

10-301/601: Introduction to Machine Learning

Lecture 8 – Optimization for Machine Learning

Geoff Gordon

with thanks to Henry Chai & Matt Gormley

HW3 out

22

- Check the Coursework tab and our HW3 FAQ post
- Due Monday Sep ~~27~~ 11:59pm (CMU time)
- Written only (no programming)
- Only 2 grace days allowed (later would go too close to the exam)

Quiz 1 logistics

- Right here, during usual class period on Friday
- First 20 minutes, followed by regular recitation
- Everyone has an assigned seat for the quiz—see Piazza
 - check **today** to make sure you know your seat
 - if you're not on the seating chart, it means we think **you** have made arrangements with ODR for them to proctor—if that's not true, contact us ASAP!
- Questions: reading and writing short Python programs related to HW1 and HW2
 - no, small syntax errors will not cause you to get the whole question wrong

Exam 1 logistics

- Note evening time (not during regular class slot)
 - 7pm Mon Sep 29
- Location & Seats: You will be split across multiple (large) rooms, and everyone will have an assigned seat (chart to be posted on Piazza)
- One letter-sized sheet of notes (both sides); no devices
- If you have exam accommodations through ODR, they will be proctoring your exam on our behalf (and you won't be in the seating chart); **you must submit the proctoring request through your student portal**

Exam 1 Topics

- Covered material: Lectures 1 – 7 (i.e., *not today*)
 - Foundations
 - Probability, Linear Algebra, Geometry, Calculus
 - Optimization
 - Important Concepts
 - Overfitting
 - Model selection / Hyperparameter optimization
 - Decision Trees
 - k -NN
 - Perceptron
 - Regression
 - Decision Tree and k -NN Regression
 - Linear Regression

Exam 1 Preparation

- Attend the midterm review OH! (recitation slot ~~this~~ Fri)
- Review the exam practice problems (to be released soon)
- Review HWs 1–3
- Consider whether you have achieved the “learning objectives” in our slides for each lecture / section
- Write your one-page cheat sheet (back and front)

Exam 1 Tips

- Solve the easy problems first
- If a problem seems extremely complicated, you might be missing something
- If you make an assumption, write it down
- Don't leave any answer blank unless out of time
 - If you look at a question and don't know the answer:
 - just start trying things
 - consider multiple approaches
 - imagine arguing for some answer and see if you like it

Recall: Gradient Descent for Linear Regression

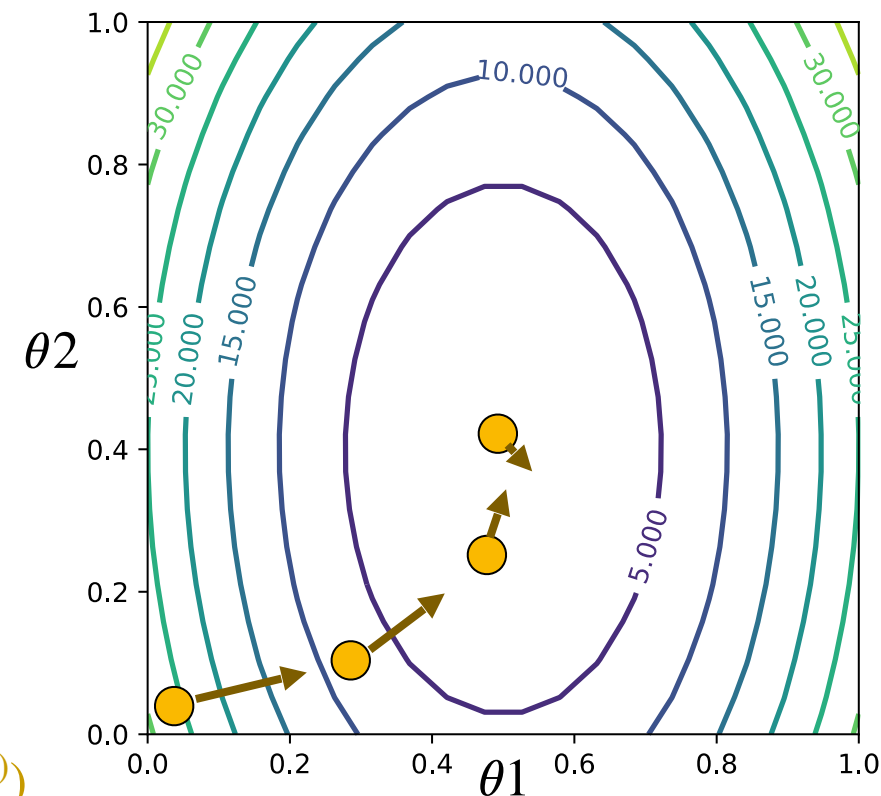
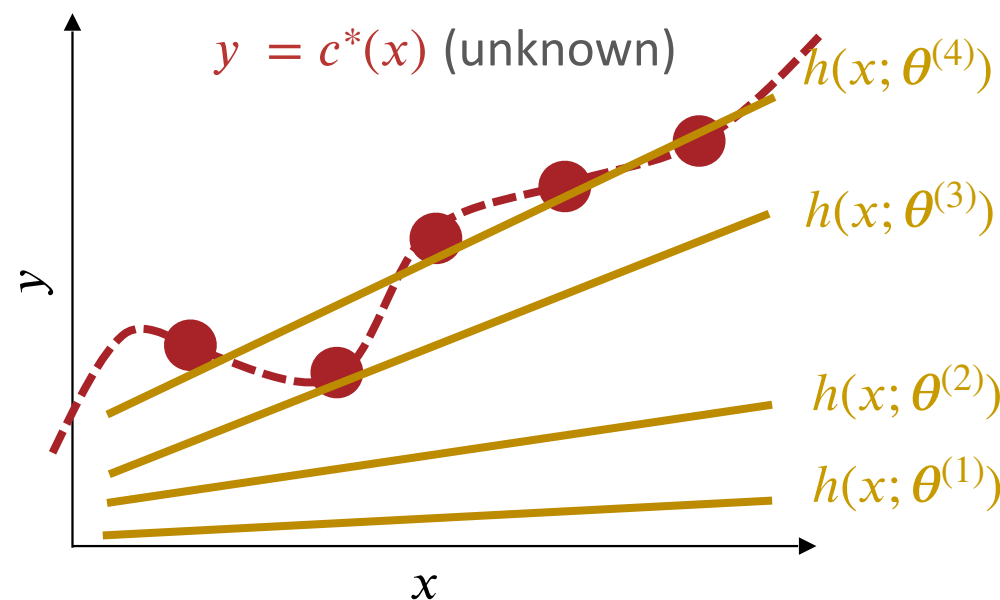
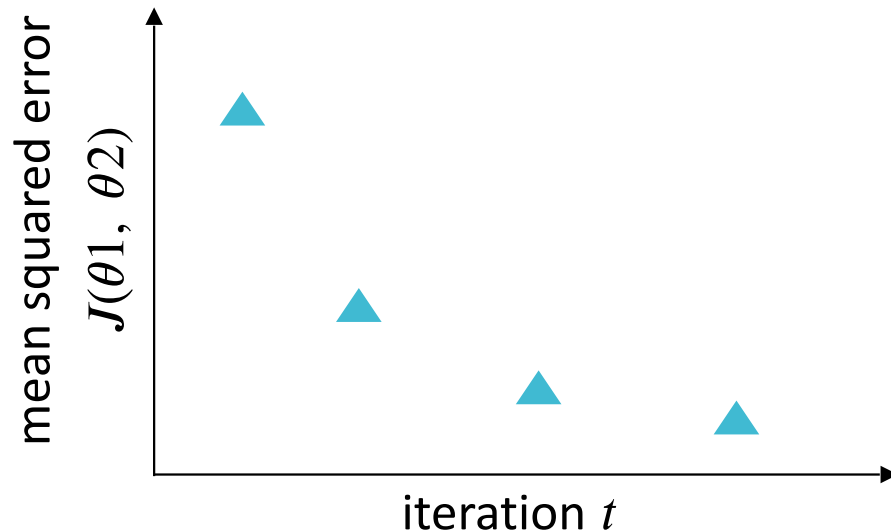
- Gradient descent for linear regression repeatedly takes steps opposite the gradient of the objective function

Algorithm 1 GD for Linear Regression

```
1: procedure GDLR( $\mathcal{D}, \theta^{(0)}$ )  
2:    $\theta \leftarrow \theta^{(0)}$  ▷ Initialize parameters  
3:   while not converged do  
4:      $\mathbf{g} \leftarrow \sum_{i=1}^N (\theta^T \mathbf{x}^{(i)} - y^{(i)}) \mathbf{x}^{(i)}$  ▷ Compute gradient  
5:      $\theta \leftarrow \theta - \gamma \mathbf{g}$  ▷ Update parameters  
6:   return  $\theta$ 
```

Recall: Gradient Descent for Linear Regression

$$J(\theta_1, \theta_2) = \frac{1}{N} \sum_{i=1}^N (y^{(i)} - \theta^T \mathbf{x}^{(i)})^2$$



t	θ_1	θ_2	$J(\theta_1, \theta_2)$
1	0.01	0.02	25.2
2	0.30	0.12	8.7
3	0.51	0.30	1.5
4	0.59	0.43	0.2

Why Gradient Descent for Linear Regression?

- Globally convergent! (see below)
- Scalable: $O(MN)$ per iteration
 - “good enough” answer = not too many iterations
- Parallelizable: distribute gradient calculation across multiple GPUs or even a cluster
- Extensions (SGD, momentum, Adam, parameter server) are even more scalable (and sometimes parallelizable)
 - much faster per iteration, somewhat more iterations
- Works for a lot of models beyond just linear regression!

Poll Question 1

- True or False: Consider two datasets

$$\mathcal{D}_1 = \left\{ \left(\mathbf{x}_1^{(1)}, y_1^{(1)} \right), \left(\mathbf{x}_1^{(2)}, y_1^{(2)} \right), \dots, \left(\mathbf{x}_1^{(N_1)}, y_1^{(N_1)} \right) \right\} \text{ and}$$

$$\mathcal{D}_2 = \left\{ \left(\mathbf{x}_2^{(1)}, y_2^{(1)} \right), \left(\mathbf{x}_2^{(2)}, y_2^{(2)} \right), \dots, \left(\mathbf{x}_2^{(N_2)}, y_2^{(N_2)} \right) \right\} \text{ where}$$

$\mathbf{x}_1^{(i)} \in \mathbb{R}^{d_1}$ and $\mathbf{x}_2^{(i)} \in \mathbb{R}^{d_2}$. Suppose $N_1 > N_2$ and $d_1 > d_2$. The maximum number of mistakes the Perceptron learning algorithm will make on $\mathcal{D}_1$ is higher than the maximum number of mistakes it will make on $\mathcal{D}_2$.

Poll Question 1

poll. m/course.org

- True or False: Consider two datasets

$$\mathcal{D}_1 = \left\{ \left(\mathbf{x}_1^{(1)}, y_1^{(1)} \right), \left(\mathbf{x}_1^{(2)}, y_1^{(2)} \right), \dots, \left(\mathbf{x}_1^{(N_1)}, y_1^{(N_1)} \right) \right\} \text{ and}$$

$$\mathcal{D}_2 = \left\{ \left(\mathbf{x}_2^{(1)}, y_2^{(1)} \right), \left(\mathbf{x}_2^{(2)}, y_2^{(2)} \right), \dots, \left(\mathbf{x}_2^{(N_2)}, y_2^{(N_2)} \right) \right\} \text{ where}$$

$\mathbf{x}_1^{(i)} \in \mathbb{R}^{d_1}$ and $\mathbf{x}_2^{(i)} \in \mathbb{R}^{d_2}$. Suppose $N_1 > N_2$ and $d_1 > d_2$. The maximum number of mistakes the Perceptron learning algorithm will make on $\mathcal{D}_1$ is higher than the maximum number of mistakes it will make on $\mathcal{D}_2$.

 $\mathbb{R} \quad \gamma$

- A. True

- B. False

- ### C. True and False (TOXIC)

Convexity

• A function $f: \mathbb{R}^D \rightarrow \mathbb{R}$ is convex if

$\forall \mathbf{x}^{(1)} \in \mathbb{R}^D, \mathbf{x}^{(2)} \in \mathbb{R}^D$ and $0 \leq c \leq 1$

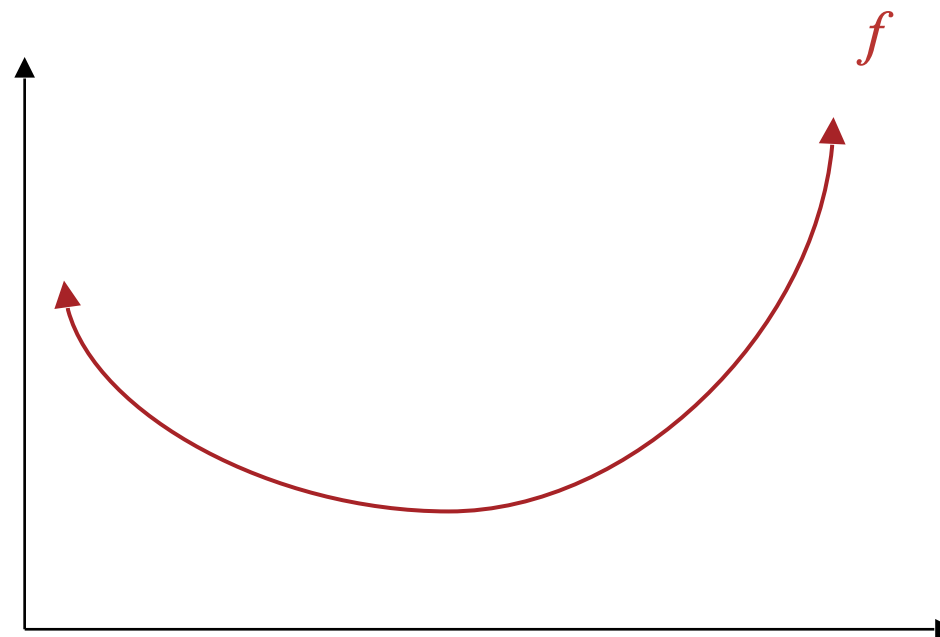
$$f(c\mathbf{x}^{(1)} + (1 - c)\mathbf{x}^{(2)}) \leq cf(\mathbf{x}^{(1)}) + (1 - c)f(\mathbf{x}^{(2)})$$

Convexity

- A function $f: \mathbb{R}^D \rightarrow \mathbb{R}$ is convex if

$$\forall \mathbf{x}^{(1)} \in \mathbb{R}^D, \mathbf{x}^{(2)} \in \mathbb{R}^D \text{ and } 0 \leq c \leq 1$$

$$f(c\mathbf{x}^{(1)} + (1 - c)\mathbf{x}^{(2)}) \leq cf(\mathbf{x}^{(1)}) + (1 - c)f(\mathbf{x}^{(2)})$$

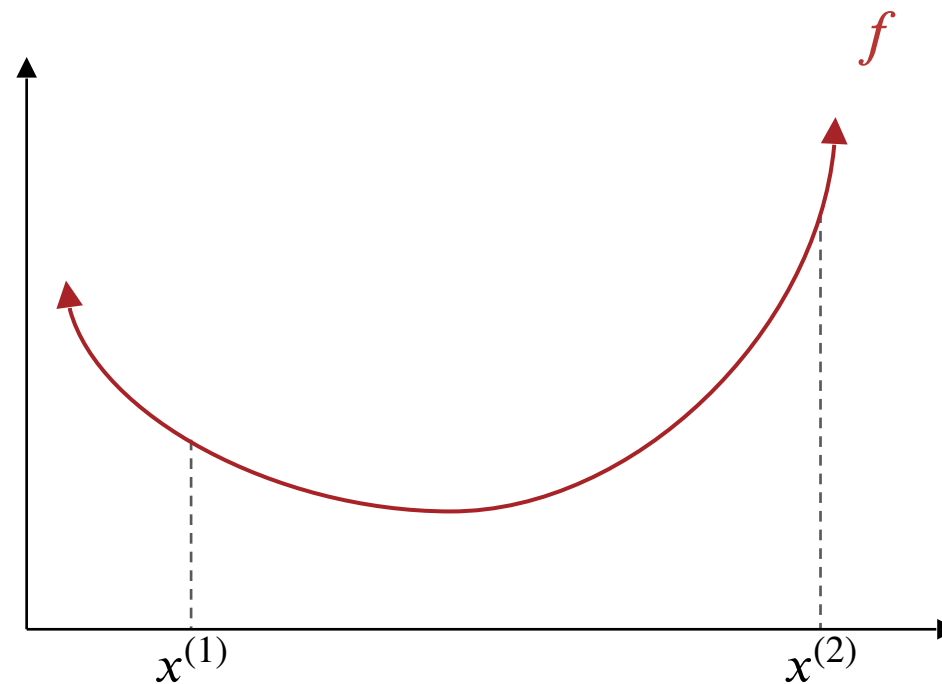


Convexity

- A function $f: \mathbb{R}^D \rightarrow \mathbb{R}$ is convex if

$$\forall \mathbf{x}^{(1)} \in \mathbb{R}^D, \mathbf{x}^{(2)} \in \mathbb{R}^D \text{ and } 0 \leq c \leq 1$$

$$f(c\mathbf{x}^{(1)} + (1 - c)\mathbf{x}^{(2)}) \leq cf(\mathbf{x}^{(1)}) + (1 - c)f(\mathbf{x}^{(2)})$$

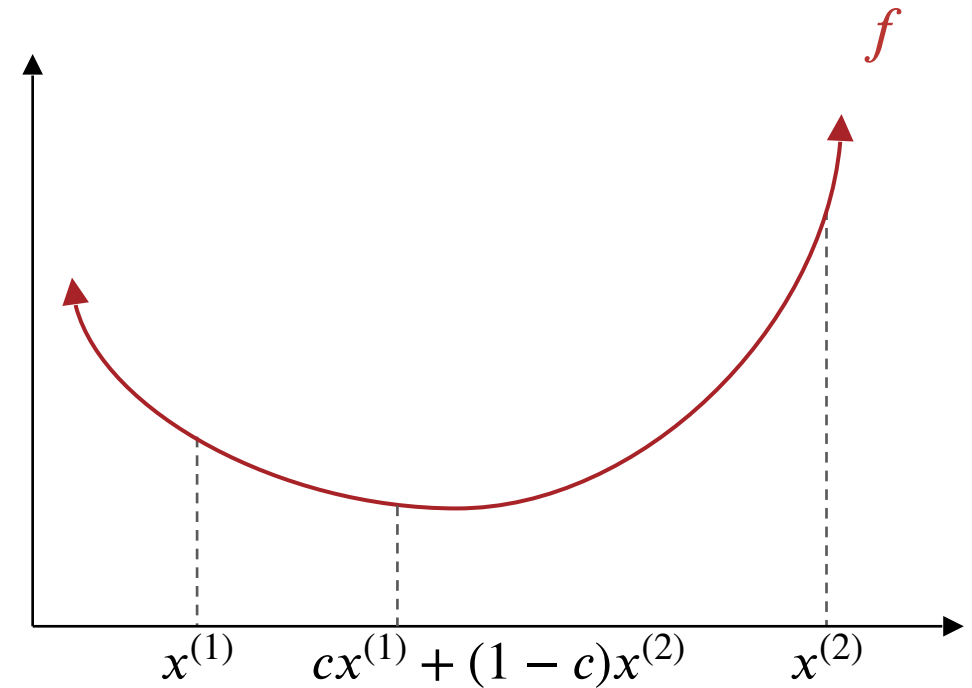


Convexity

- A function $f: \mathbb{R}^D \rightarrow \mathbb{R}$ is convex if

$$\forall \mathbf{x}^{(1)} \in \mathbb{R}^D, \mathbf{x}^{(2)} \in \mathbb{R}^D \text{ and } 0 \leq c \leq 1$$

$$f(c\mathbf{x}^{(1)} + (1 - c)\mathbf{x}^{(2)}) \leq cf(\mathbf{x}^{(1)}) + (1 - c)f(\mathbf{x}^{(2)})$$

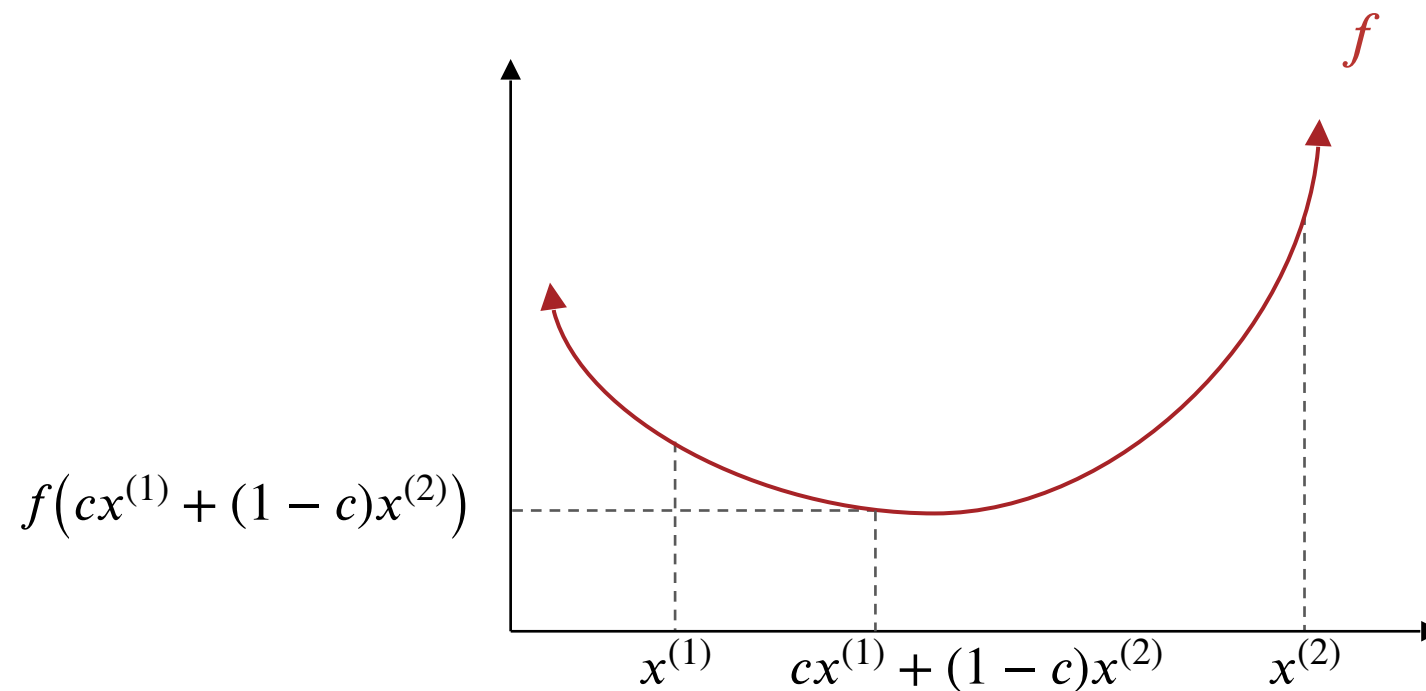


Convexity

- A function $f: \mathbb{R}^D \rightarrow \mathbb{R}$ is convex if

$$\forall \mathbf{x}^{(1)} \in \mathbb{R}^D, \mathbf{x}^{(2)} \in \mathbb{R}^D \text{ and } 0 \leq c \leq 1$$

$$f(c\mathbf{x}^{(1)} + (1 - c)\mathbf{x}^{(2)}) \leq cf(\mathbf{x}^{(1)}) + (1 - c)f(\mathbf{x}^{(2)})$$

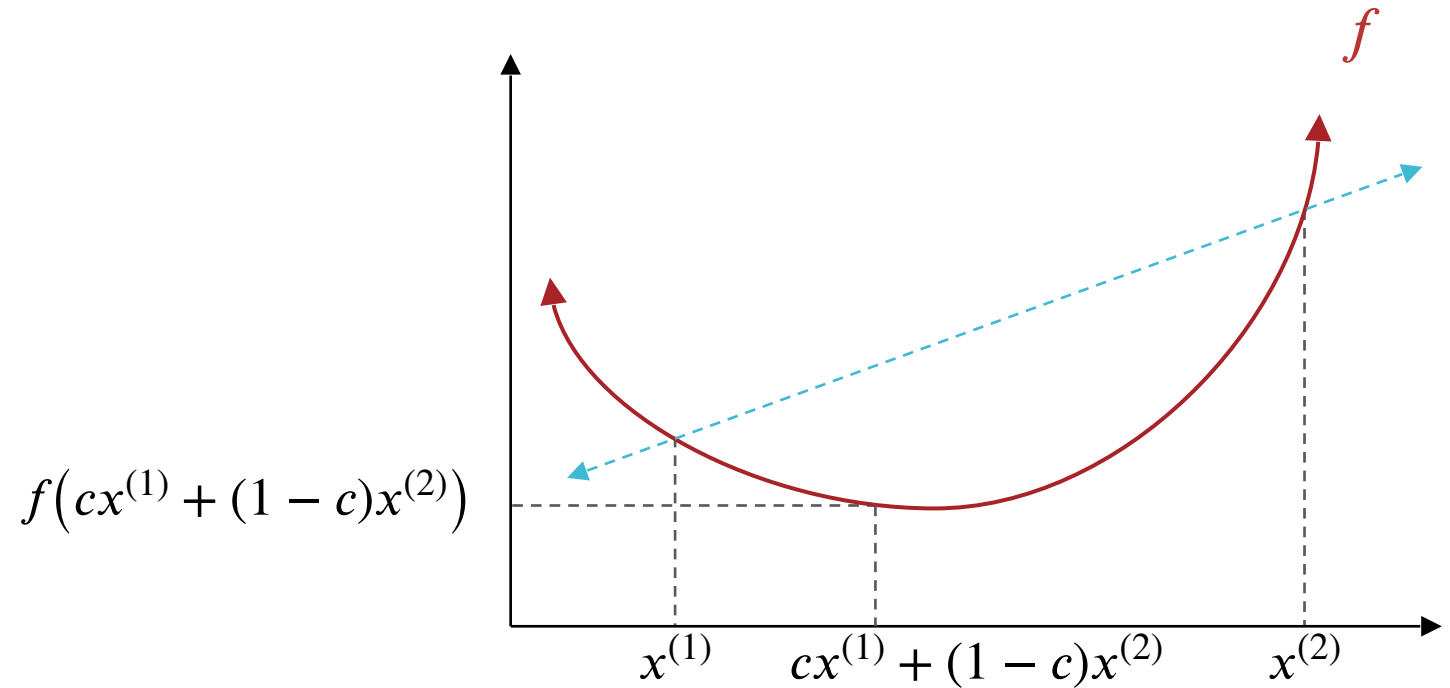


Convexity

• A function $f: \mathbb{R}^D \rightarrow \mathbb{R}$ is convex if

$$\forall \mathbf{x}^{(1)} \in \mathbb{R}^D, \mathbf{x}^{(2)} \in \mathbb{R}^D \text{ and } 0 \leq c \leq 1$$

$$f(c\mathbf{x}^{(1)} + (1 - c)\mathbf{x}^{(2)}) \leq cf(\mathbf{x}^{(1)}) + (1 - c)f(\mathbf{x}^{(2)})$$

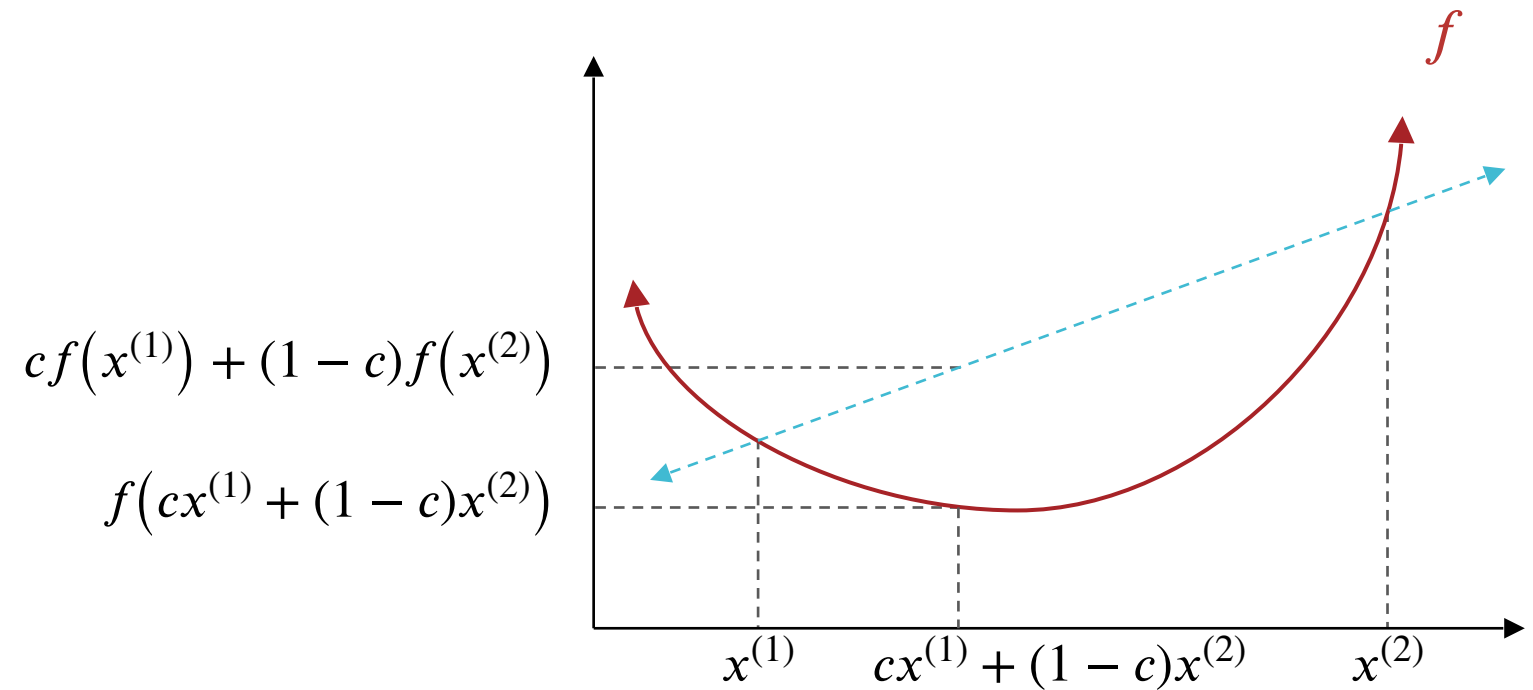


Convexity

- A function $f: \mathbb{R}^D \rightarrow \mathbb{R}$ is convex if

$$\forall \mathbf{x}^{(1)} \in \mathbb{R}^D, \mathbf{x}^{(2)} \in \mathbb{R}^D \text{ and } 0 \leq c \leq 1$$

$$f(c\mathbf{x}^{(1)} + (1-c)\mathbf{x}^{(2)}) \leq cf(\mathbf{x}^{(1)}) + (1-c)f(\mathbf{x}^{(2)})$$

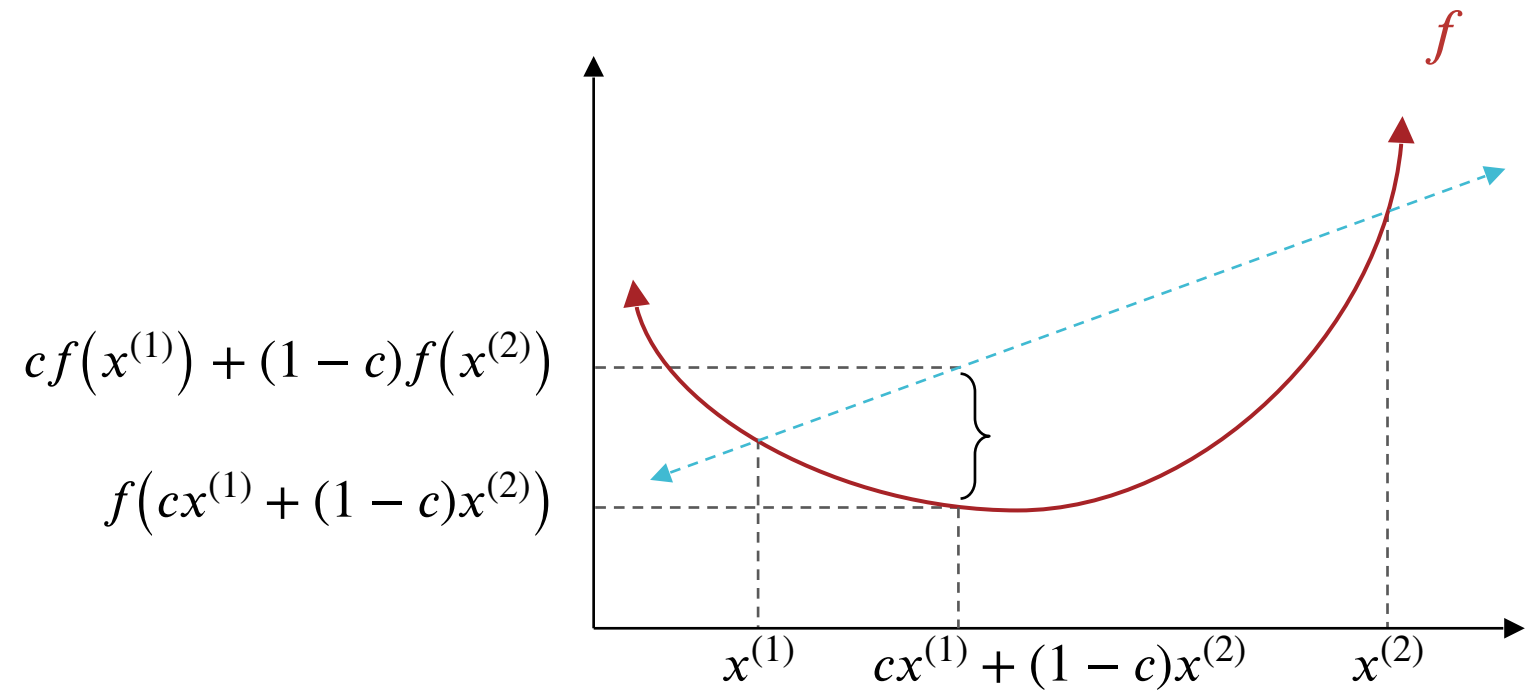


Convexity

- A function $f: \mathbb{R}^D \rightarrow \mathbb{R}$ is convex if

$$\forall \mathbf{x}^{(1)} \in \mathbb{R}^D, \mathbf{x}^{(2)} \in \mathbb{R}^D \text{ and } 0 \leq c \leq 1$$

$$f(c\mathbf{x}^{(1)} + (1 - c)\mathbf{x}^{(2)}) \leq cf(\mathbf{x}^{(1)}) + (1 - c)f(\mathbf{x}^{(2)})$$

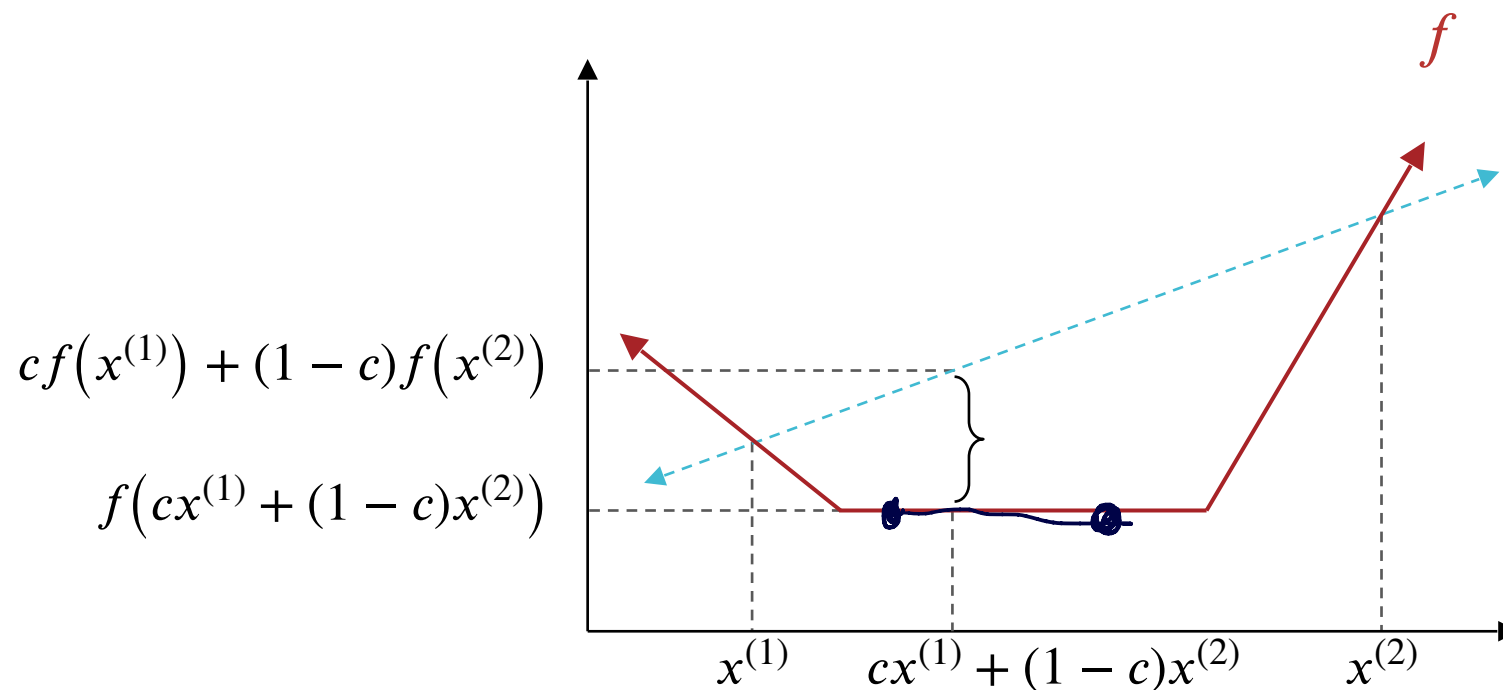


Convexity

- A function $f: \mathbb{R}^D \rightarrow \mathbb{R}$ is convex if

$$\forall \mathbf{x}^{(1)} \in \mathbb{R}^D, \mathbf{x}^{(2)} \in \mathbb{R}^D \text{ and } 0 \leq c \leq 1$$

$$f(c\mathbf{x}^{(1)} + (1 - c)\mathbf{x}^{(2)}) \leq cf(\mathbf{x}^{(1)}) + (1 - c)f(\mathbf{x}^{(2)})$$

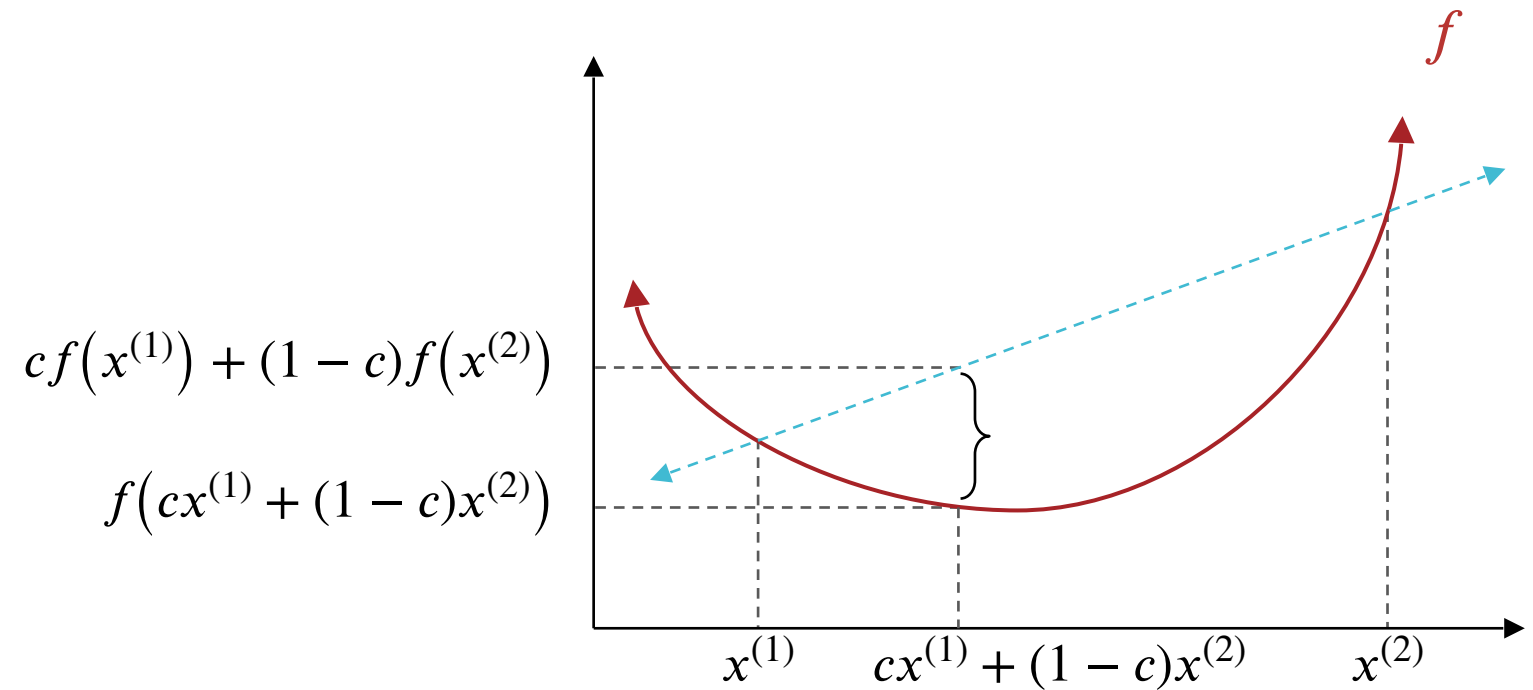


Convexity

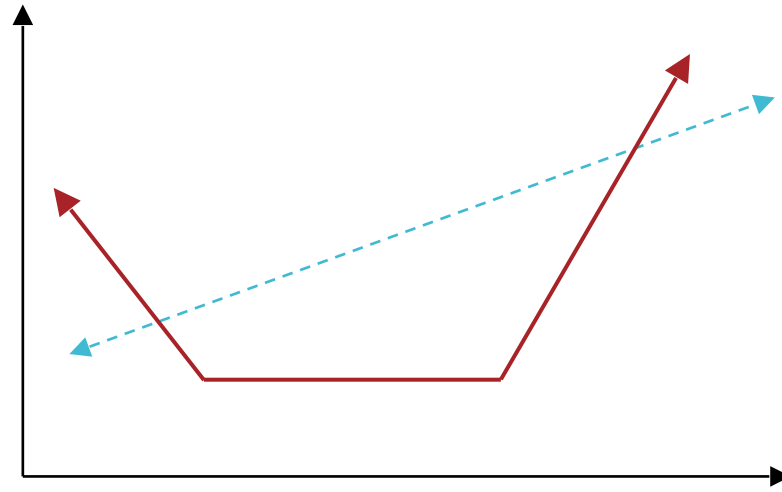
- A function $f: \mathbb{R}^D \rightarrow \mathbb{R}$ is *strictly convex* if

$$\forall \mathbf{x}^{(1)} \in \mathbb{R}^D, \mathbf{x}^{(2)} \in \mathbb{R}^D \text{ and } 0 < c < 1$$

$$f(c\mathbf{x}^{(1)} + (1 - c)\mathbf{x}^{(2)}) < cf(\mathbf{x}^{(1)}) + (1 - c)f(\mathbf{x}^{(2)})$$

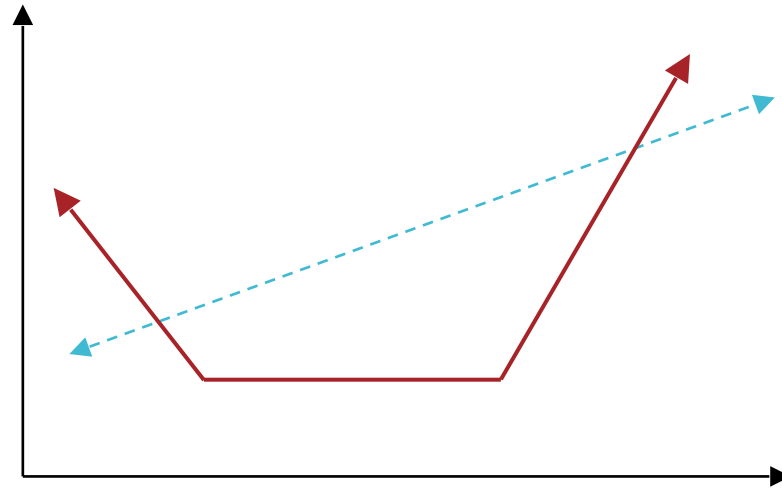


Convexity: local and global minima

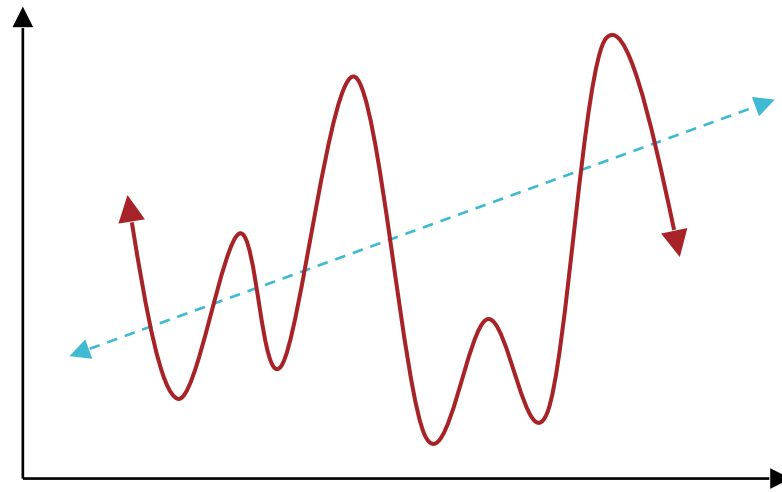


Convex functions

Convexity: local and global minima

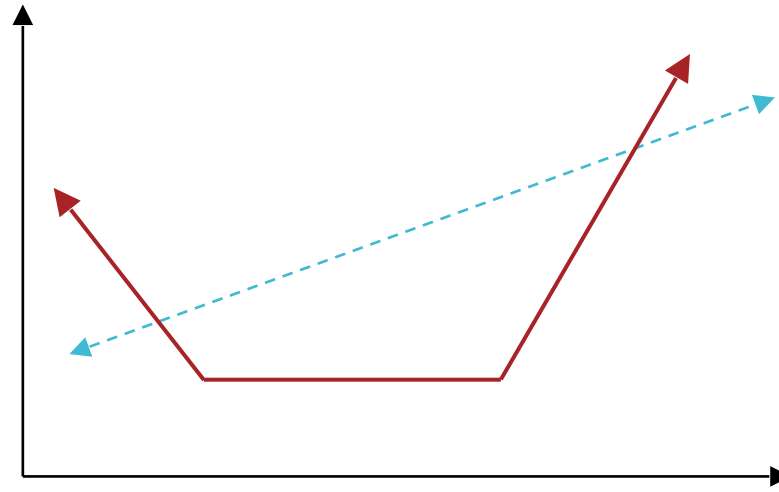


Convex functions

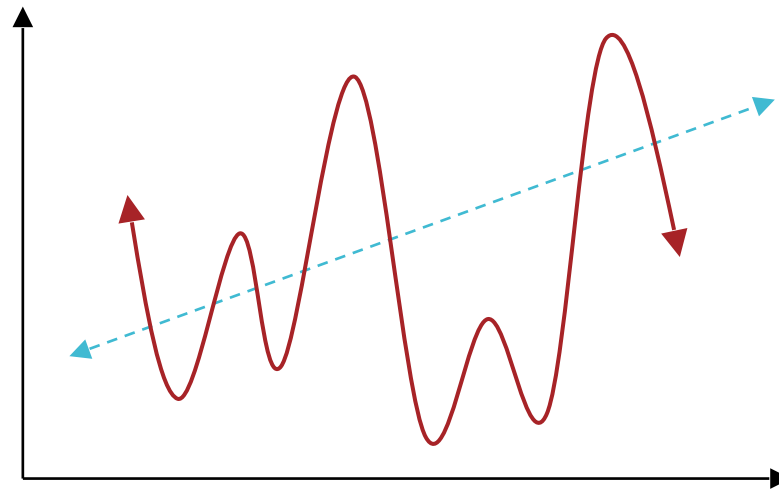


Non-convex functions

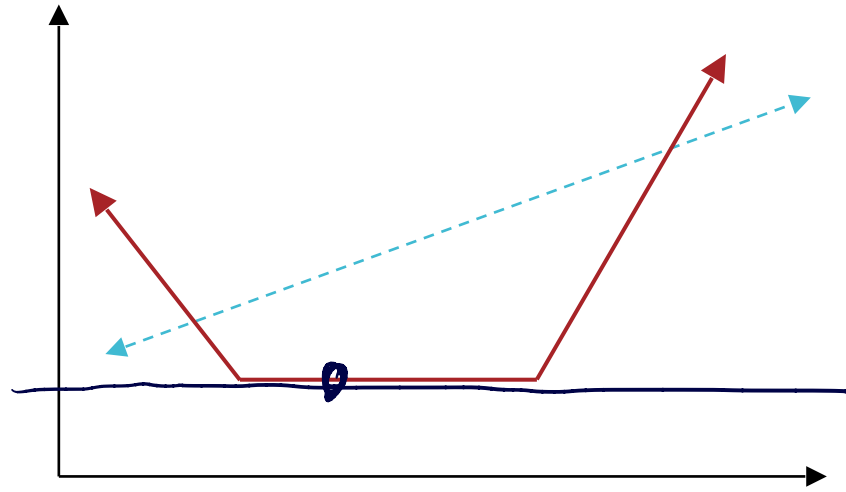
Convexity: local and global minima



Given a function $f: \mathbb{R}^D \rightarrow \mathbb{R}$

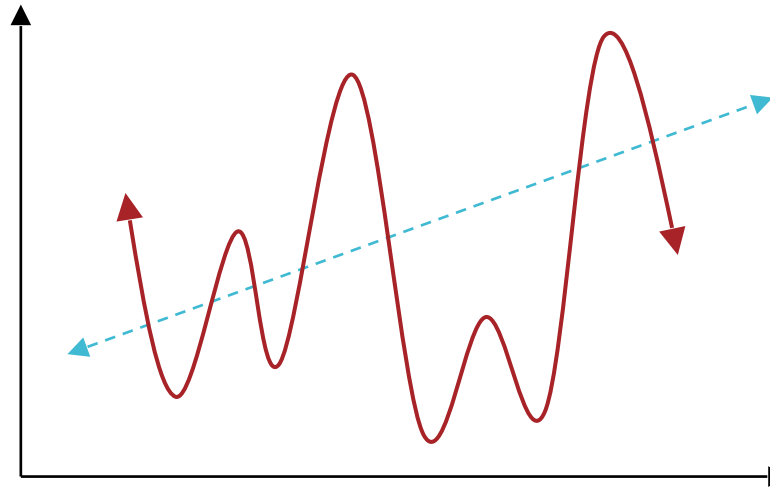


Convexity: local and global minima

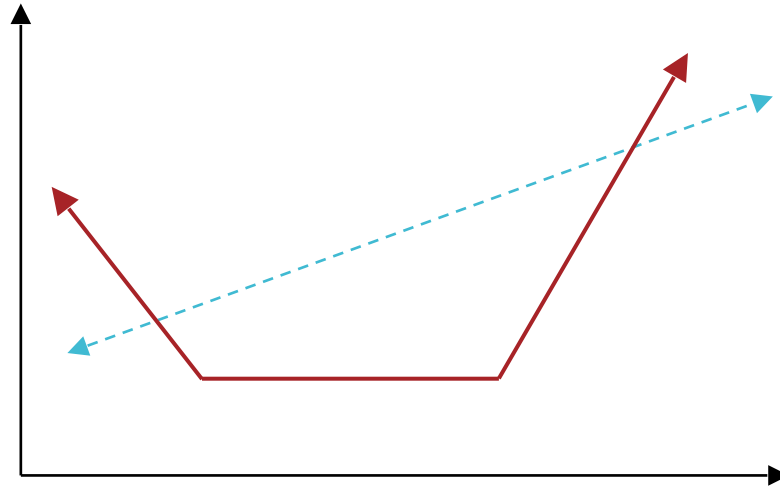


Given a function $f: \mathbb{R}^D \rightarrow \mathbb{R}$

- $\mathbf{x}^*$ is a global minimum iff
$$f(\mathbf{x}^*) \leq f(\mathbf{x}) \quad \forall \mathbf{x} \in \mathbb{R}^D$$

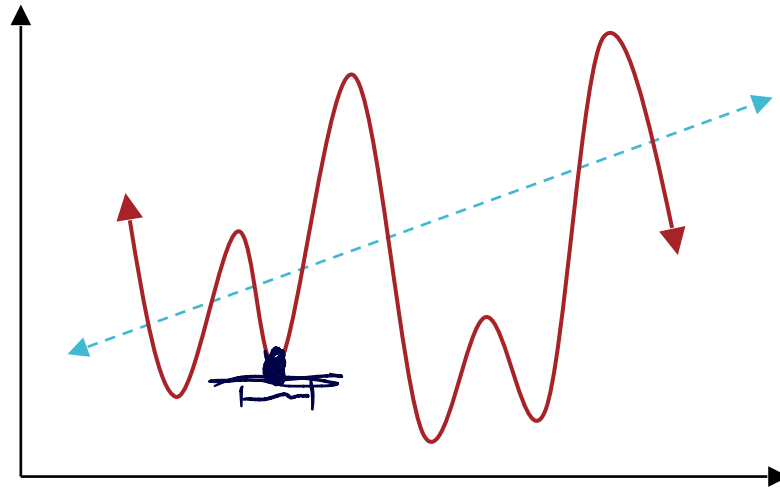


Convexity: local and global minima



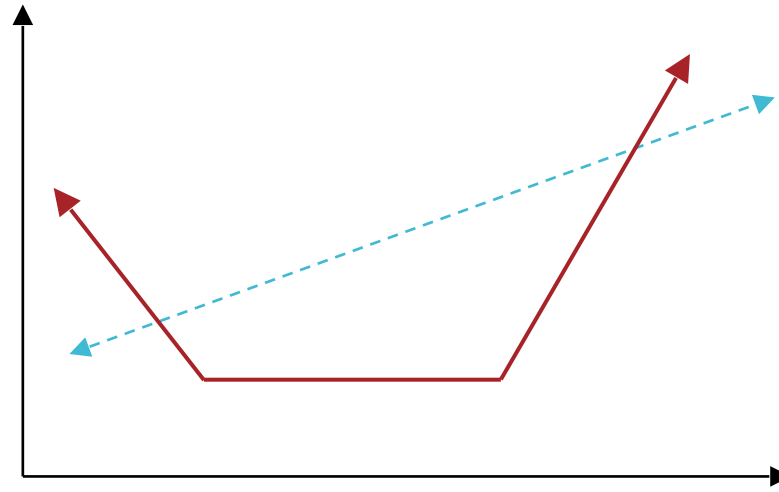
Given a function $f: \mathbb{R}^D \rightarrow \mathbb{R}$

- $\mathbf{x}^*$ is a global minimum iff
$$f(\mathbf{x}^*) \leq f(\mathbf{x}) \quad \forall \mathbf{x} \in \mathbb{R}^D$$

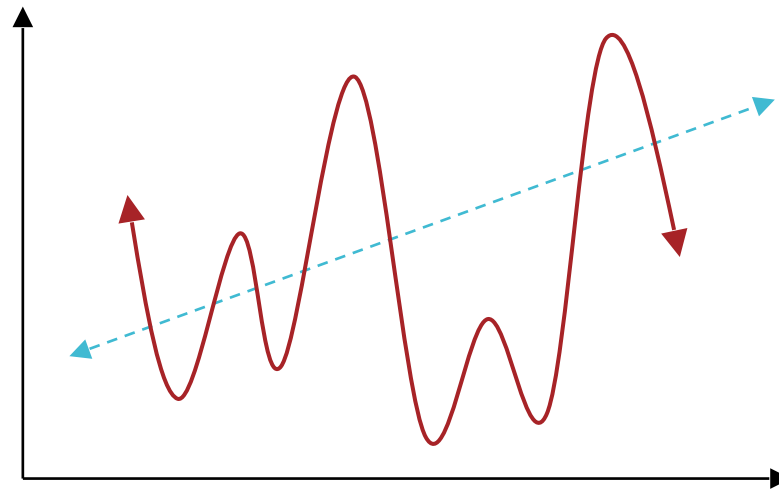


- $\mathbf{x}^*$ is a local minimum iff
$$\exists \epsilon \text{ s.t. } f(\mathbf{x}^*) \leq f(\mathbf{x}) \quad \forall \mathbf{x} \text{ s.t. } \|\mathbf{x} - \mathbf{x}^*\|_2 < \epsilon$$

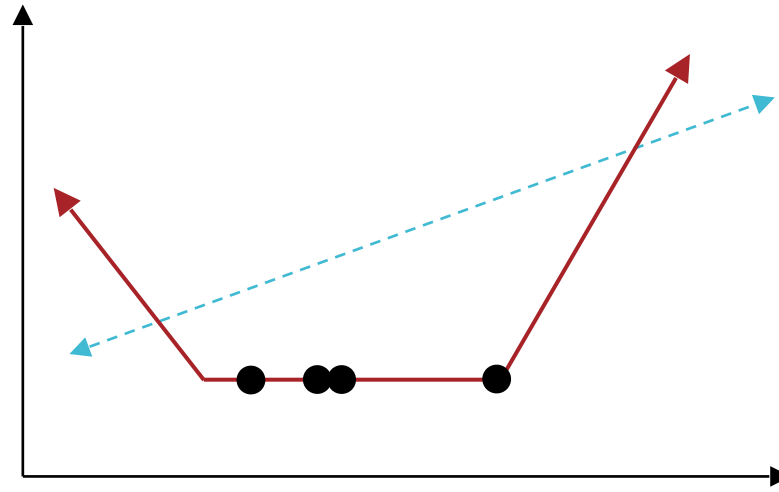
Convexity: local and global minima



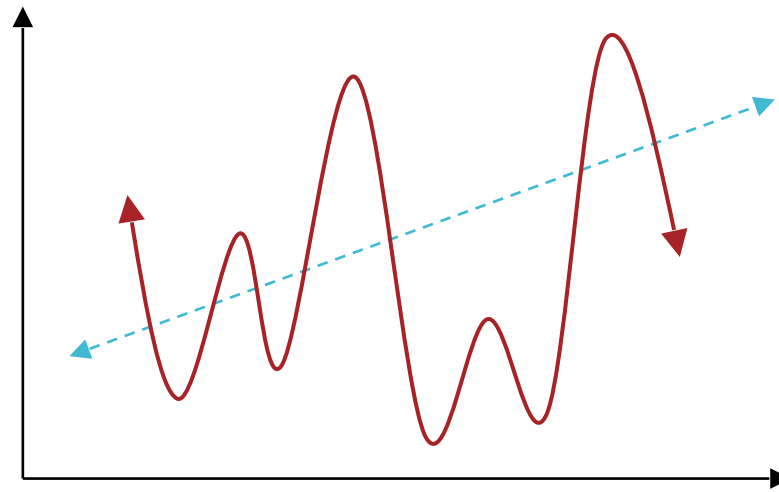
Convex functions:
Each local minimum is a
global minimum!



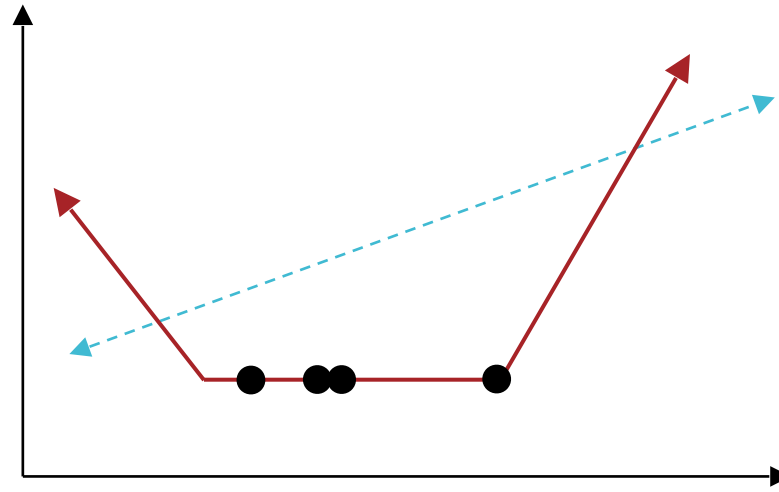
Convexity: local and global minima



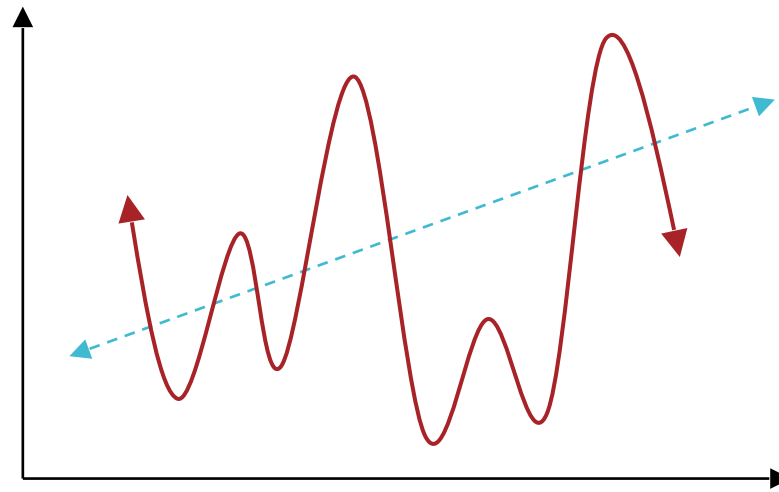
Convex functions:
Each local minimum is a
global minimum!



Convexity: local and global minima

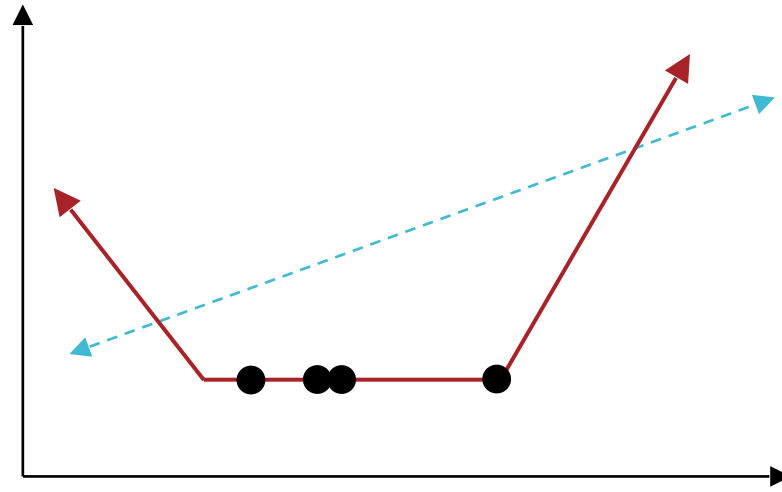


Convex functions:
Each local minimum is a
global minimum!

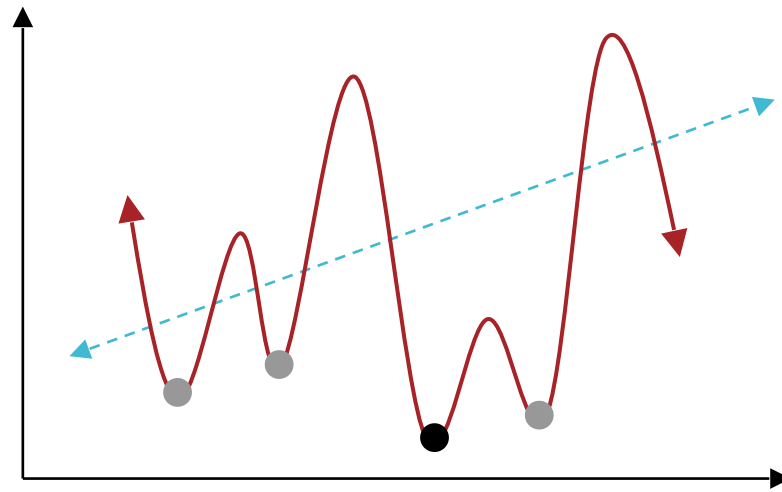


Non-convex functions:
A local minimum may or may
not be a global minimum...

Convexity: local and global minima

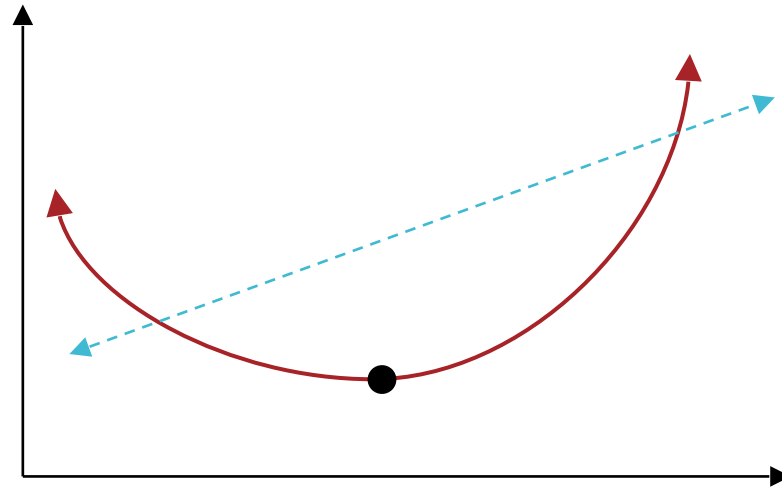


Convex functions:
Each local minimum is a
global minimum!

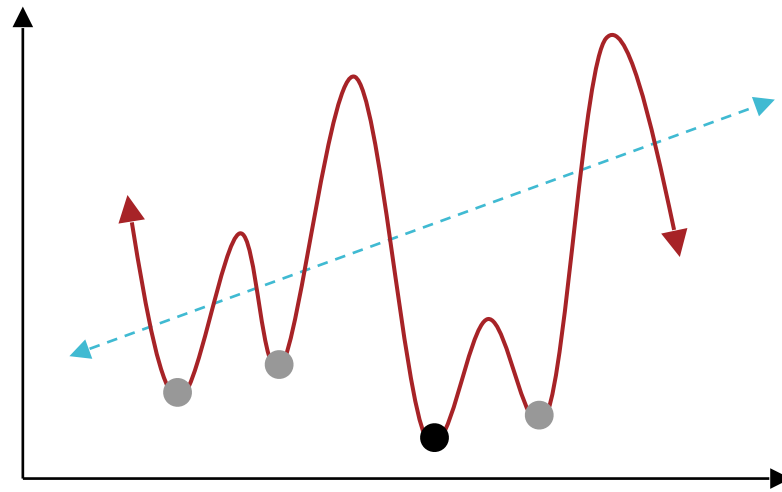


Non-convex functions:
A local minimum may or may
not be a global minimum...

Convexity: local and global minima



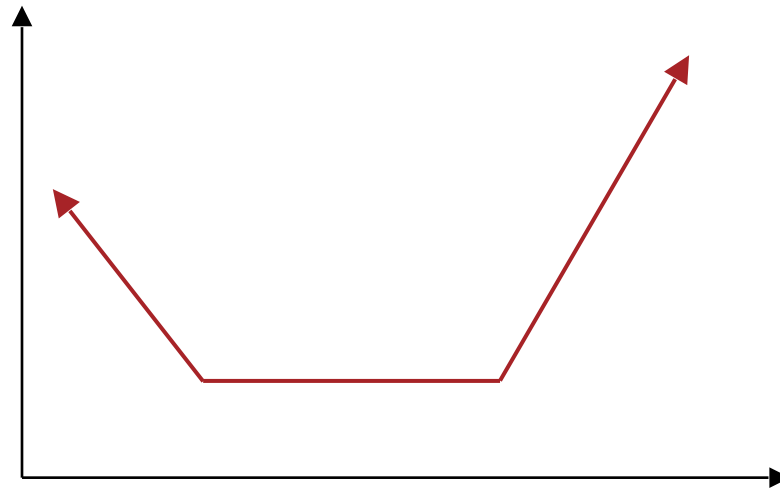
Strictly convex functions:
There exists a unique global
minimum!



Non-convex functions:
A local minimum may or may
not be a global minimum...

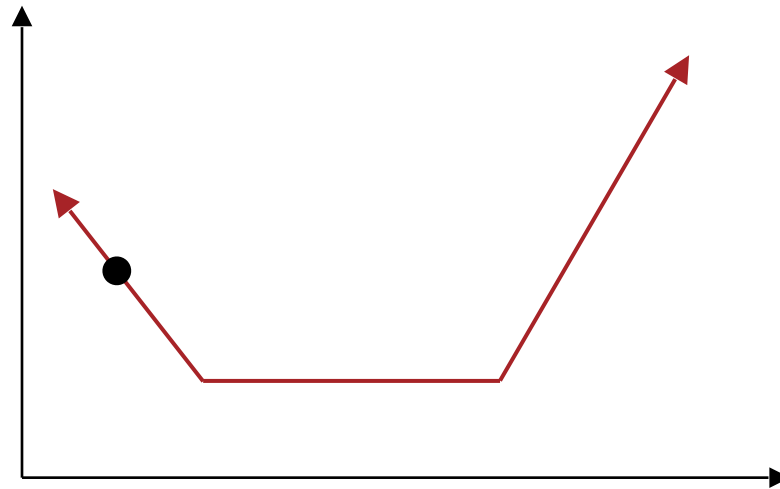
Gradient Descent & Convexity

- Gradient descent is a local optimization algorithm – it will converge to a local minimum (if it converges)
 - Works great if the objective function is convex!



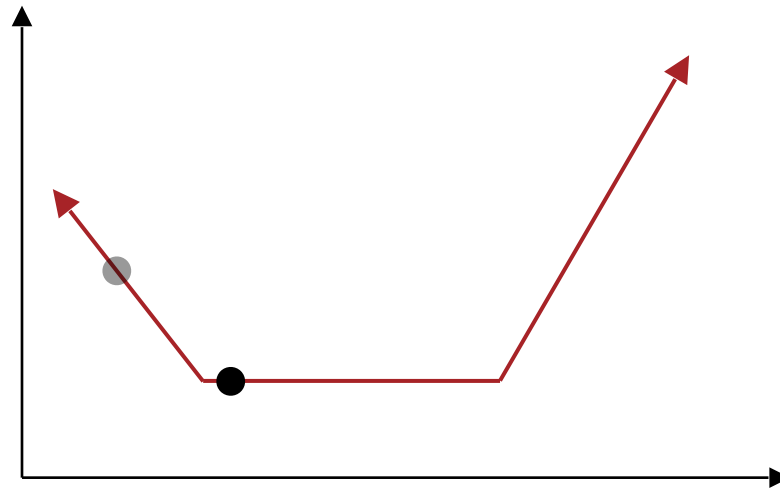
Gradient Descent & Convexity

- Gradient descent is a local optimization algorithm – it will converge to a local minimum (if it converges)
 - Works great if the objective function is convex!



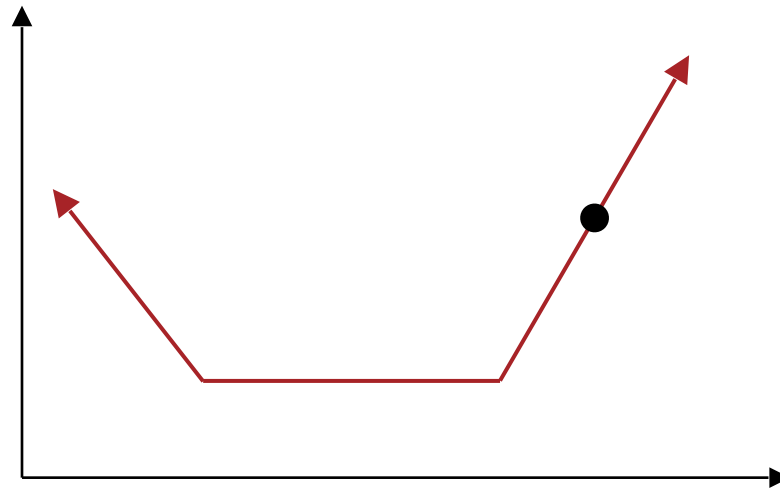
Gradient Descent & Convexity

- Gradient descent is a local optimization algorithm – it will converge to a local minimum (if it converges)
 - Works great if the objective function is convex!



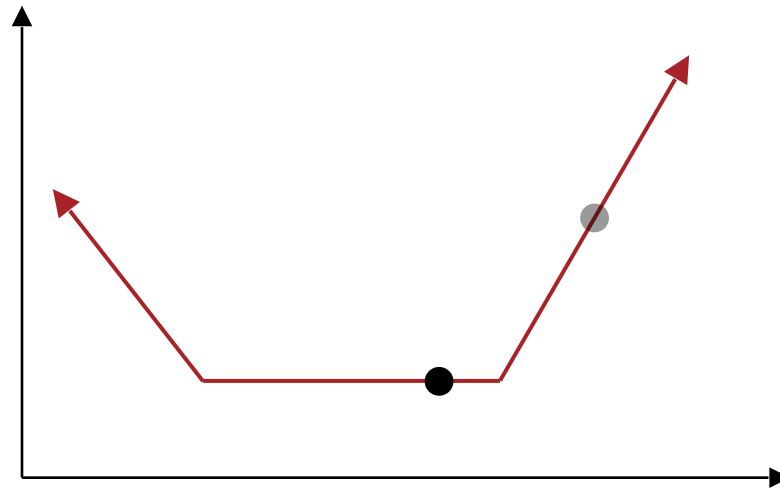
Gradient Descent & Convexity

- Gradient descent is a local optimization algorithm – it will converge to a local minimum (if it converges)
 - Works great if the objective function is convex!



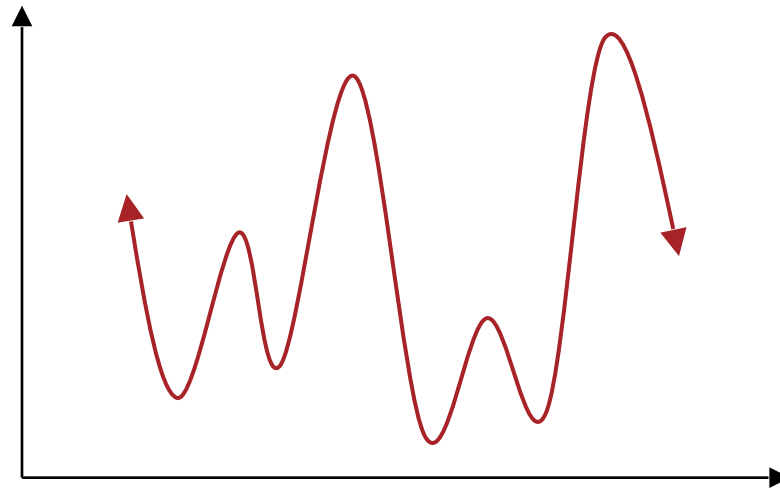
Gradient Descent & Convexity

- Gradient descent is a local optimization algorithm – it will converge to a local minimum (if it converges)
 - Works great if the objective function is convex!



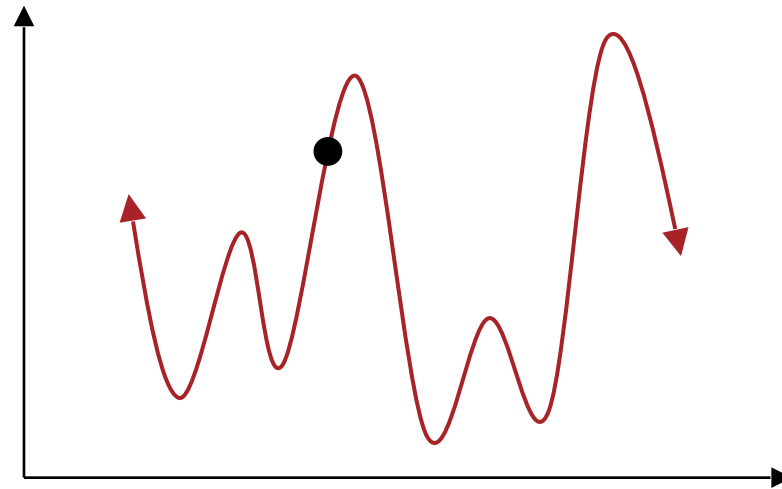
Gradient Descent & Convexity

- Gradient descent is a local optimization algorithm – it will converge to a local minimum (if it converges)
 - Not ideal if the objective function is non-convex...



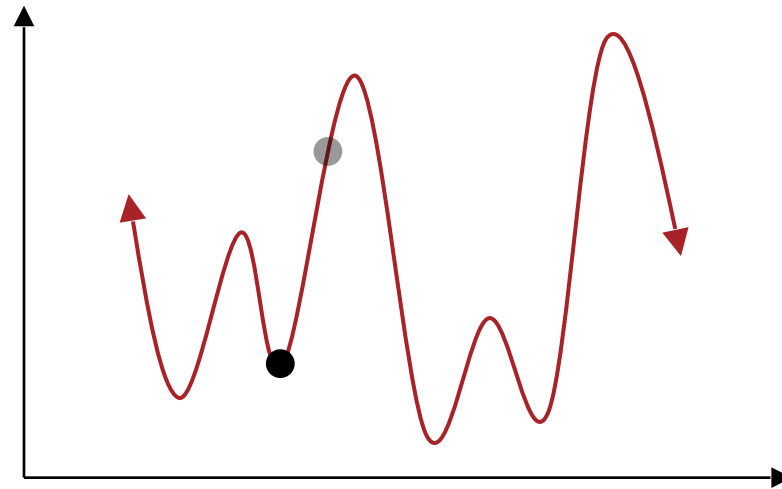
Gradient Descent & Convexity

- Gradient descent is a local optimization algorithm – it will converge to a local minimum (if it converges)
 - Not ideal if the objective function is non-convex...



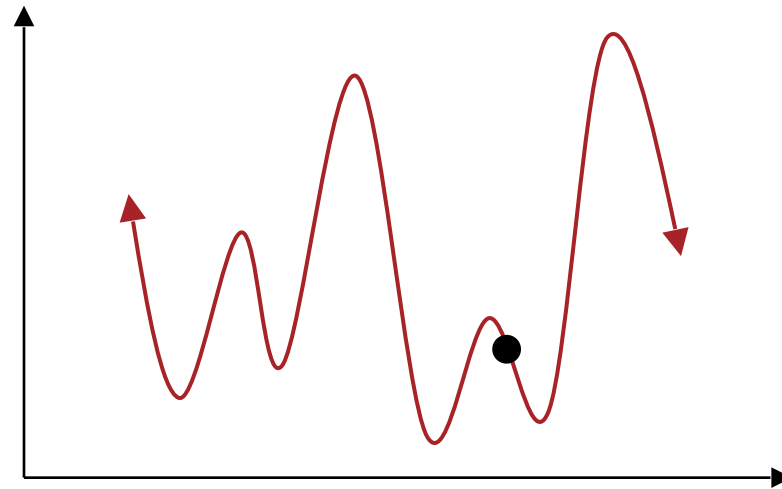
Gradient Descent & Convexity

- Gradient descent is a local optimization algorithm – it will converge to a local minimum (if it converges)
 - Not ideal if the objective function is non-convex...



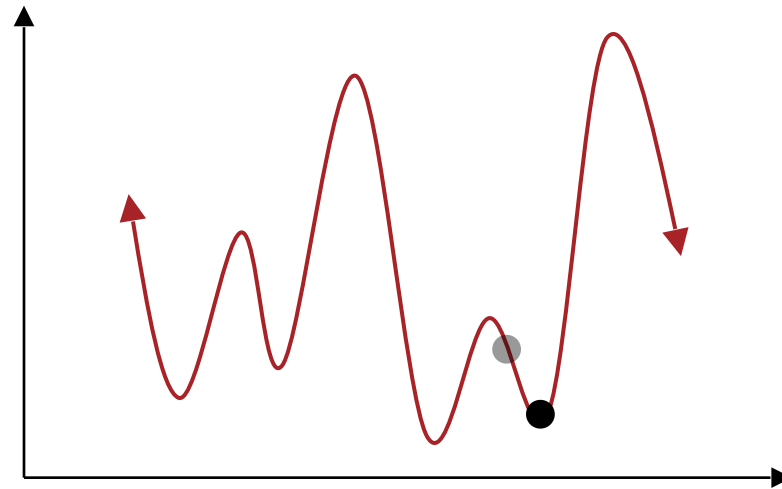
Gradient Descent & Convexity

- Gradient descent is a local optimization algorithm – it will converge to a local minimum (if it converges)
 - Not ideal if the objective function is non-convex...



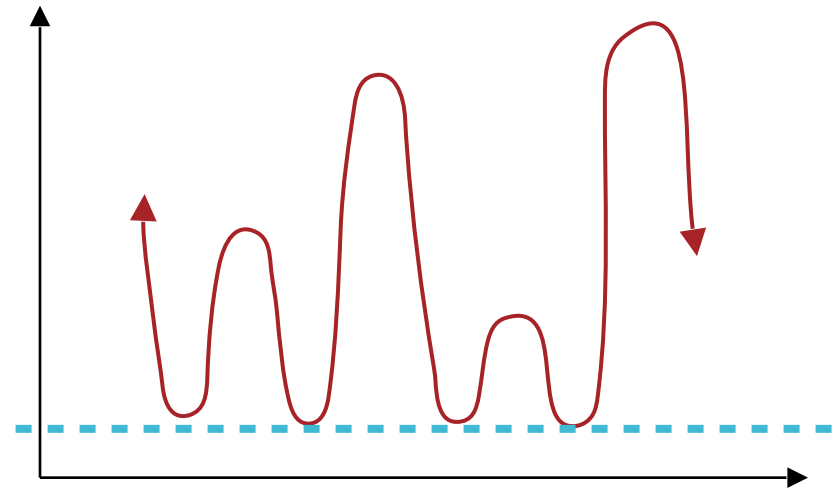
Gradient Descent & Convexity

- Gradient descent is a local optimization algorithm – it will converge to a local minimum (if it converges)
 - Not ideal if the objective function is non-convex...

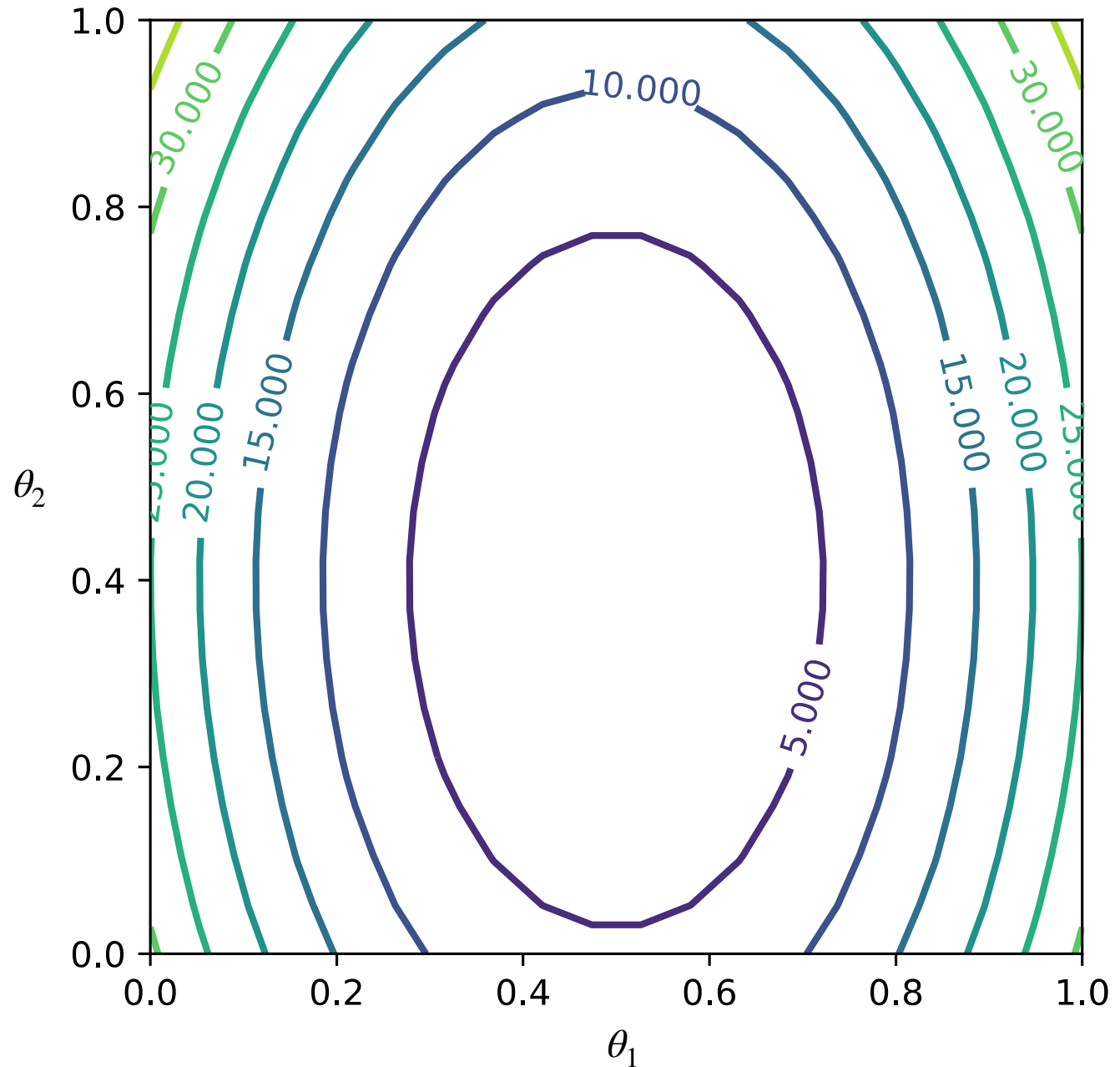


Gradient Descent & Convexity

- Gradient descent is a local optimization algorithm – it will converge to a local minimum (if it converges)
 - But can be OK if $\exists$ lots of pretty good local optima

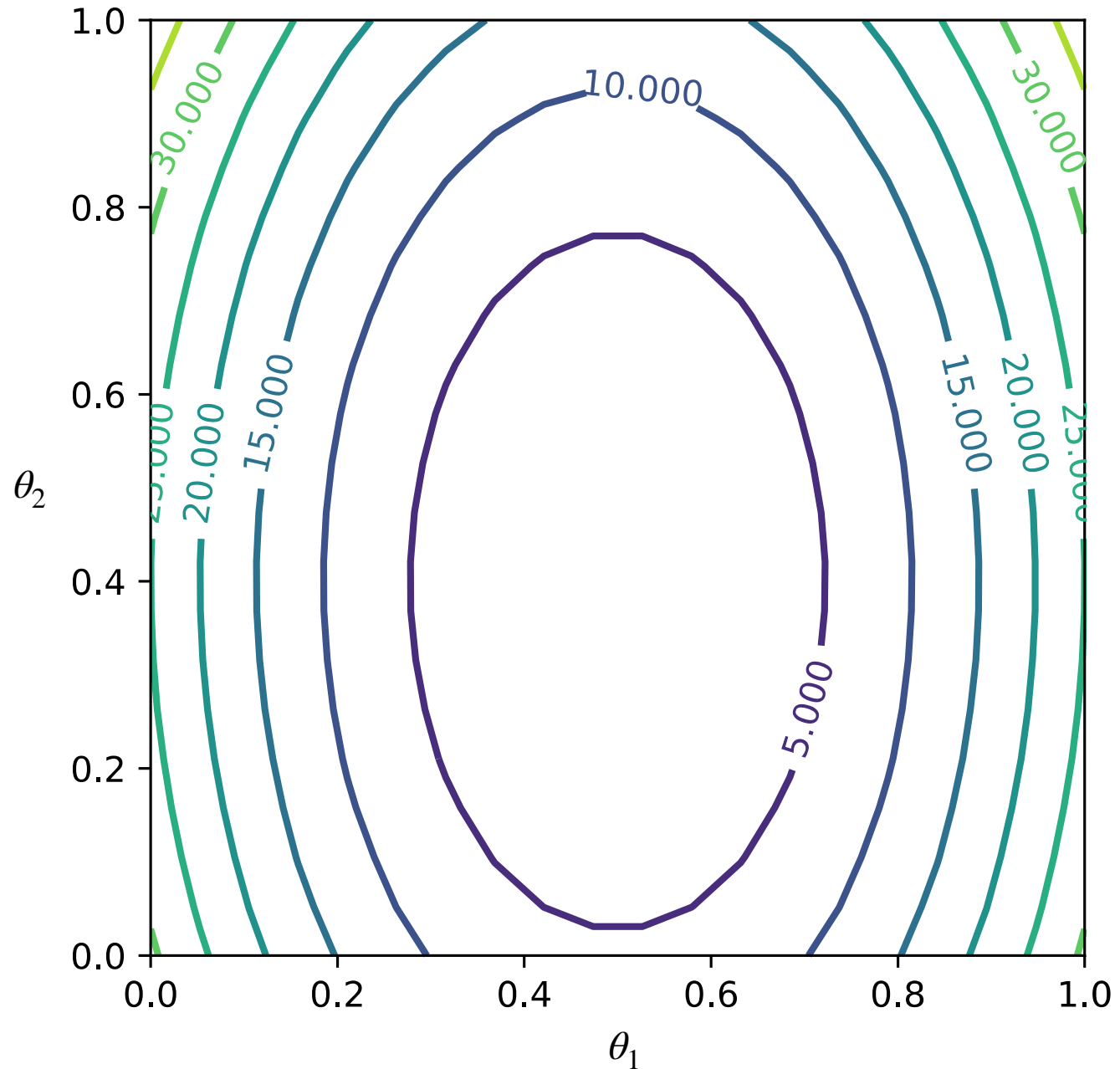


Gradient
descent for
linear
regression:
MSE is
convex!



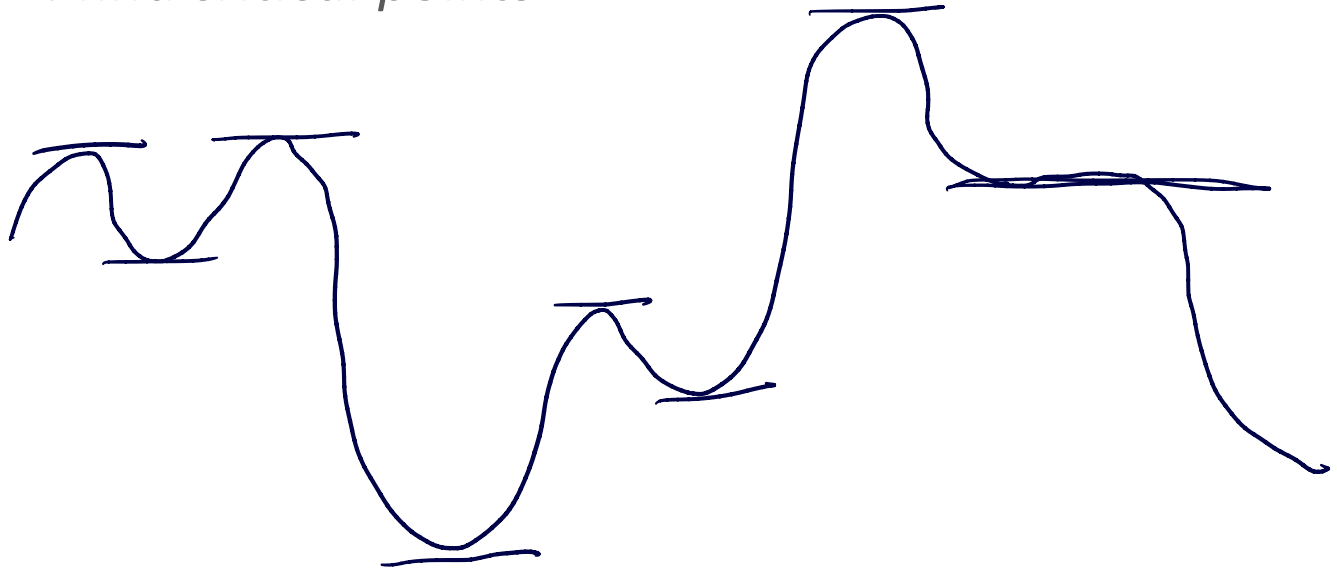
Gradient descent for linear regression: MSE is convex!

*but not always strictly convex
(see below for example)*



OK, but calculus class told me a different way to find optima

- Set $\nabla J(\theta) = 0$ and solve for θ
 - 1D: find *critical points*



- higher D, same idea: critical point $\leftrightarrow$ tangent is flat
- Not all critical points are local optima in general, but in our case J is convex!

Closed Form Optimization

$$= \begin{pmatrix} \mathbf{x}^{(1)T} \cdot \boldsymbol{\theta} \\ \vdots \\ \mathbf{x}^{(N)T} \cdot \boldsymbol{\theta} \end{pmatrix} \stackrel{N}{\sim} \begin{matrix} \text{D+1} \\ \boxed{} \\ \text{D+1} \end{matrix} \quad \begin{matrix} \hat{\mathbf{y}}^{(1)} \\ \vdots \\ \hat{\mathbf{y}}^{(N)} \end{matrix}$$

$$(\hat{\mathbf{y}} - \mathbf{y})^T (\hat{\mathbf{y}} - \mathbf{y}) \rightarrow \mathbf{X} \boldsymbol{\theta}$$

- Notation: given training data $\mathcal{D} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^N$

$$\mathbf{X} = \begin{bmatrix} 1 & \mathbf{x}^{(1)T} \\ 1 & \mathbf{x}^{(2)T} \\ \vdots & \vdots \\ 1 & \mathbf{x}^{(N)T} \end{bmatrix} = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_D^{(1)} \\ 1 & x_1^{(2)} & \dots & x_D^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(N)} & \dots & x_D^{(N)} \end{bmatrix} \in \mathbb{R}^{N \times D+1}$$

is the *design matrix*

$\mathbf{y} = [y^{(1)}, \dots, y^{(N)}]^T \in \mathbb{R}^N$ is the *target vector*

Minimizing the Mean Squared Error

$$J(\theta) = \frac{1}{N} \sum_{i=1}^N \frac{1}{2} (y^{(i)} - \theta^T x^{(i)})^2 = \frac{1}{2N} \sum_{i=1}^N \left(x^{(i)T} \theta - y^{(i)} \right)^2$$

$$\begin{aligned} J(\theta) &= \frac{1}{2N} (X\theta - y)^T (X\theta - y) \\ &= \frac{1}{2N} \left((X\theta)^T (X\theta) - \underbrace{y^T X\theta}_{= y^T X\theta} - (X\theta)^T y + y^T y \right) \\ &= \frac{1}{2N} \left(\theta^T X^T X \theta - 2 y^T X \theta + y^T y \right) \end{aligned}$$

$$\nabla J(\theta) = \frac{1}{N} \left(\underbrace{2 X^T X \theta}_{(D+1) \times 1} - \underbrace{2 X^T y}_{(D+1) \times 1} + 0 \right)$$

$$\begin{matrix} X^T X \theta & = & X^T y \\ (D+1) \times (D+1) & & (D+1) \end{matrix}$$

Finding the optimal θ

$$\underbrace{X^T X \theta = X^T y}_{((D+1) \times N) \cdot (N \times (D+1))} \leftarrow \text{the normal equations}$$

- Does a solution even exist? *yes!*
- Is it unique? *maybe not*
- If not, which one?
min norm sol'n arg min $\|\theta\|$ st
- How expensive is it to find desired solution?

$$O(D^3) + \underline{\underline{O(D^2 N)}}$$

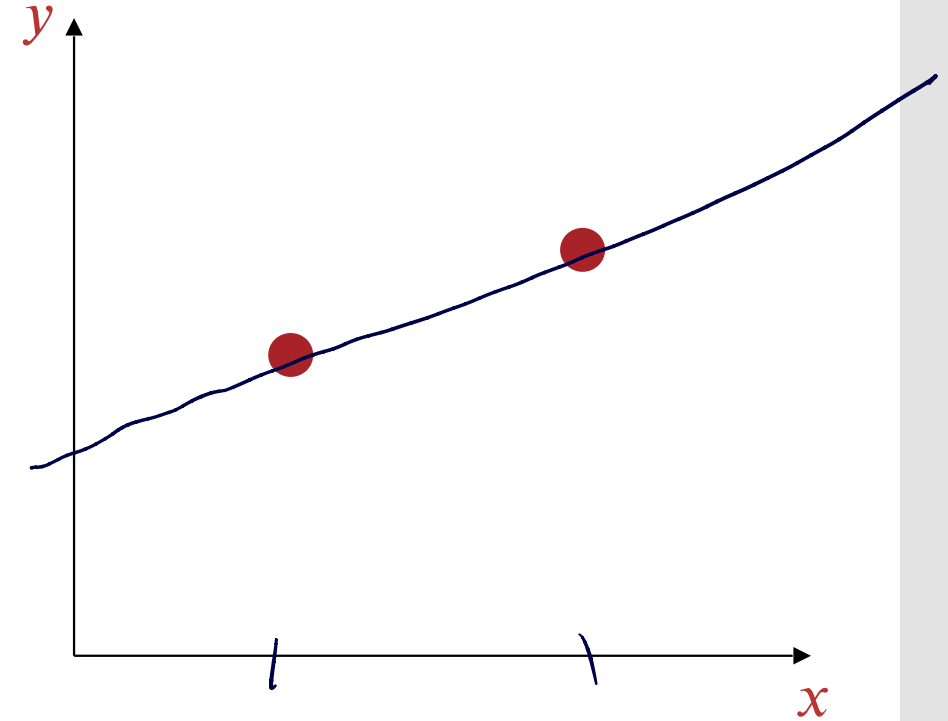
Finding the optimal θ

$$X^T X \theta = X^T y \quad \leftarrow \text{the } \textit{normal equations}$$

- I just want to find the solution, what should I call?
 - Options (all in `numpy.linalg`):
 - ~~• `inv(X.T @ X) @ (X.T @ y)`~~
 - ~~• `pinv(X.T @ X) @ (X.T @ y)`~~
 - ✓ • `solve(X.T @ X, X.T @ y)`
 - ✓ • `lstsq(X.T @ X, X.T @ y)`
 - ✓✓ • `lstsq(X, y)`

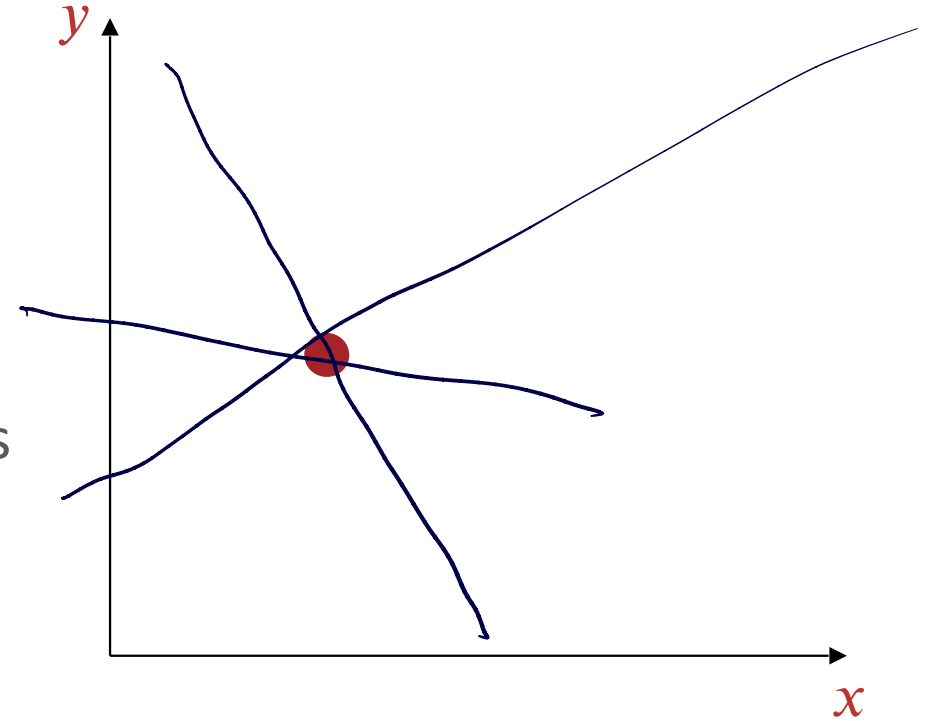
Linear Regression: Uniqueness

- Consider a 1D linear regression model trained to minimize the mean squared error: how many optimal solutions (i.e., sets of parameters θ) are there for the given dataset?



Linear Regression: Uniqueness

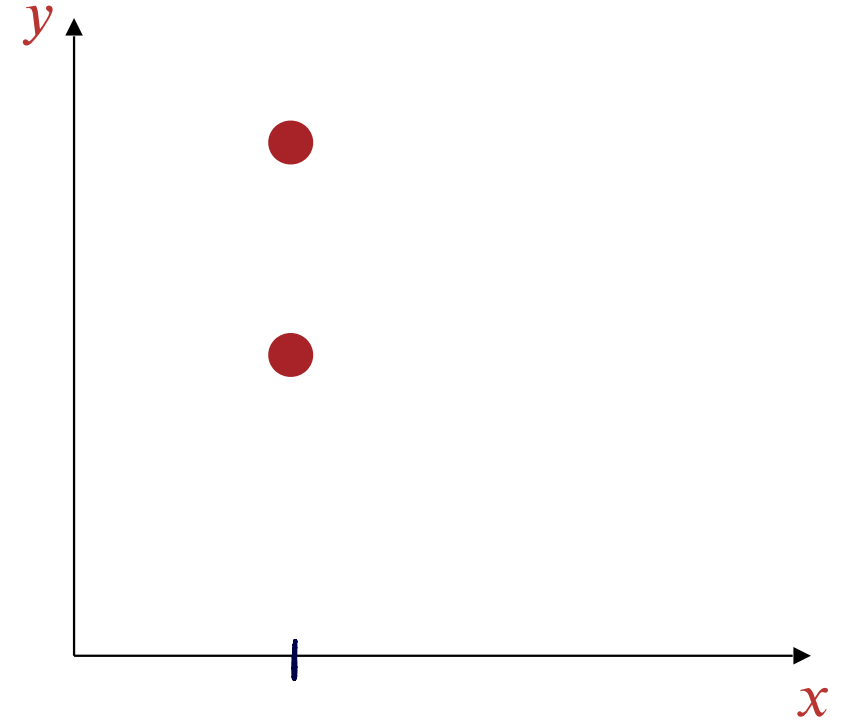
- Consider a 1D linear regression model trained to minimize the mean squared error: how many optimal solutions (i.e., sets of parameters θ) are there for the given dataset?



Poll Question 3

*[no poll 2
today!]*

- Consider a 1D linear regression model trained to minimize the mean squared error: how many optimal solutions (i.e., sets of parameters θ) are there for the given dataset?



Poll Question 3

[no poll 2 today!]

- Consider a 1D linear regression model trained to minimize the mean squared error: how many optimal solutions (i.e., sets of parameters θ) are there for the given dataset?

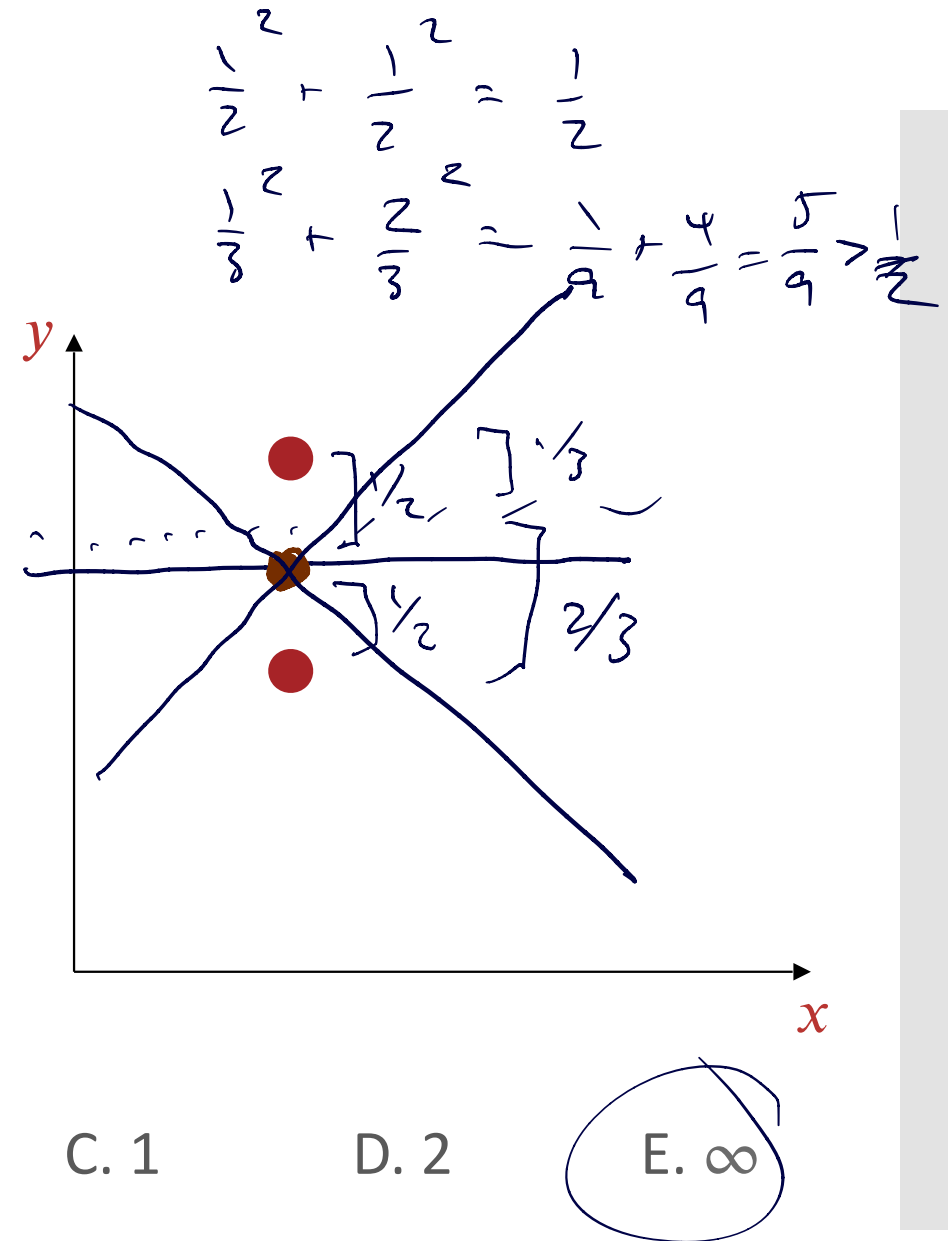
A. -1 (TOXIC)

B. 0

C. 1

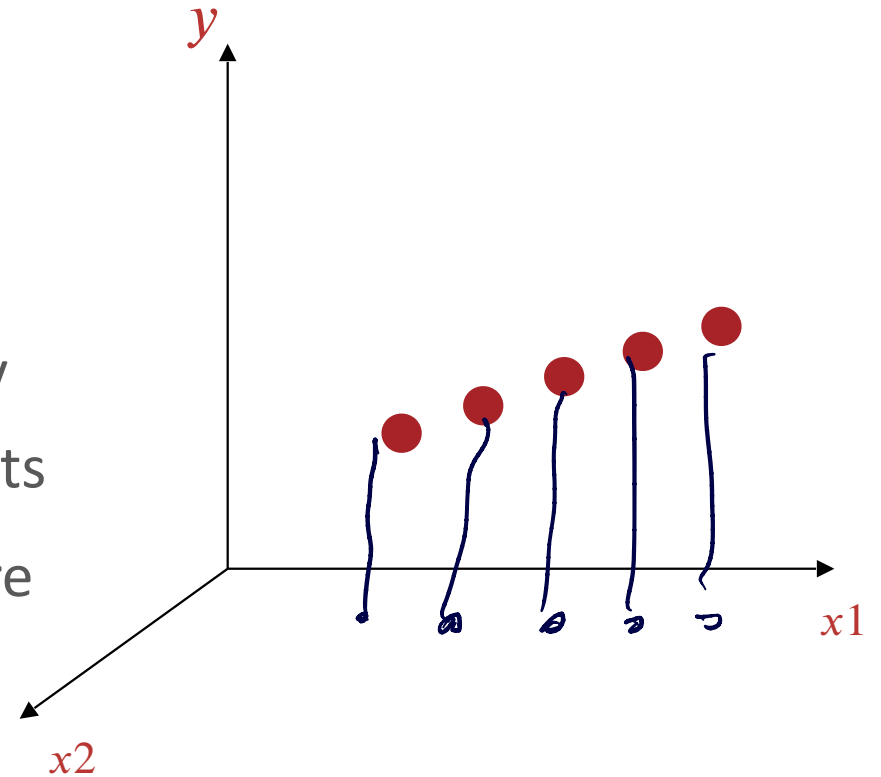
D. 2

E. ∞



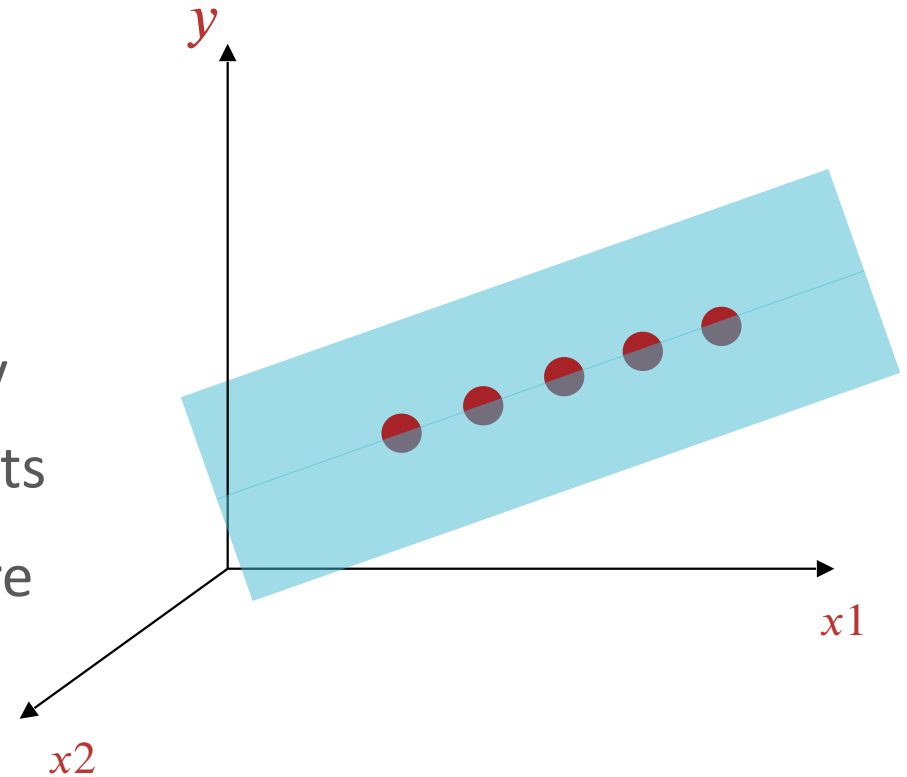
Linear Regression: Uniqueness

- Consider a 2D linear regression model trained to minimize the mean squared error: how many optimal solutions (i.e., sets of parameters θ) are there for the given dataset?



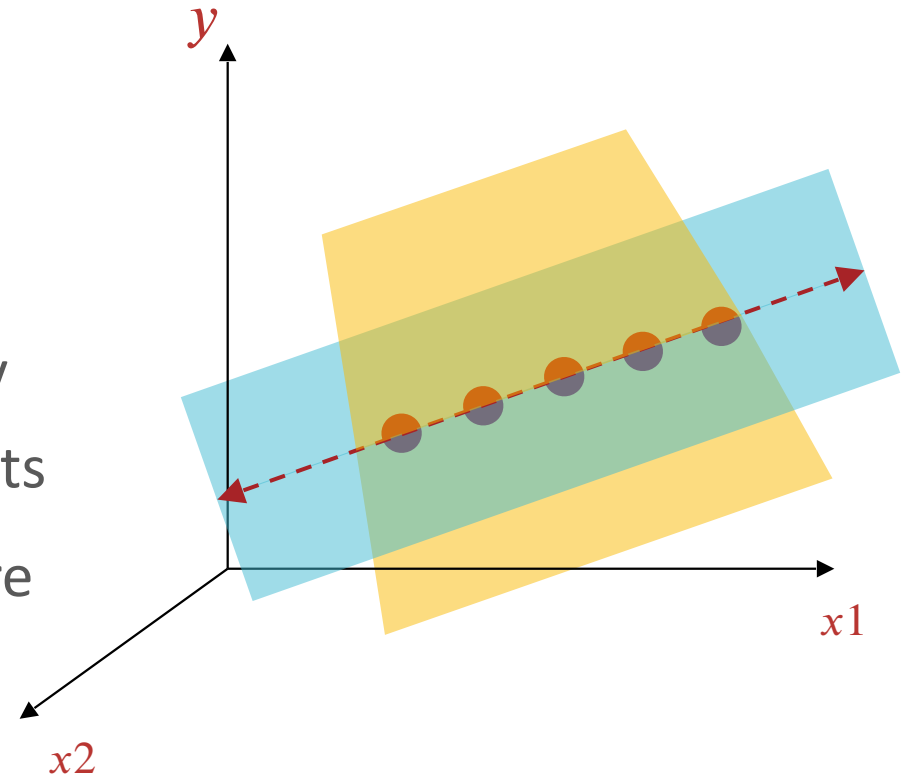
Linear Regression: Uniqueness

- Consider a 2D linear regression model trained to minimize the mean squared error: how many optimal solutions (i.e., sets of parameters θ) are there for the given dataset?



Linear Regression: Uniqueness

- Consider a 2D linear regression model trained to minimize the mean squared error: how many optimal solutions (i.e., sets of parameters θ) are there for the given dataset?



Linearly independent features

- Solution is unique if number of examples N is *at least as large as* number of *linearly independent* features
 - independent = not a linear function of other features
 - e.g., $x_1^{(i)}$ = length (mm) and $x_2^{(i)}$ = length (inches): *not* linearly independent
 - e.g., if examples are on a line in 2d, number of linearly independent features = 1 (not 2)
 - e.g., on a line or plane in 3d: 1 or 2, not 3
 - etc.
- Can drop features to get independence, or resolve ambiguity by picking minimum norm solution

Recap: exact vs. gradient descent

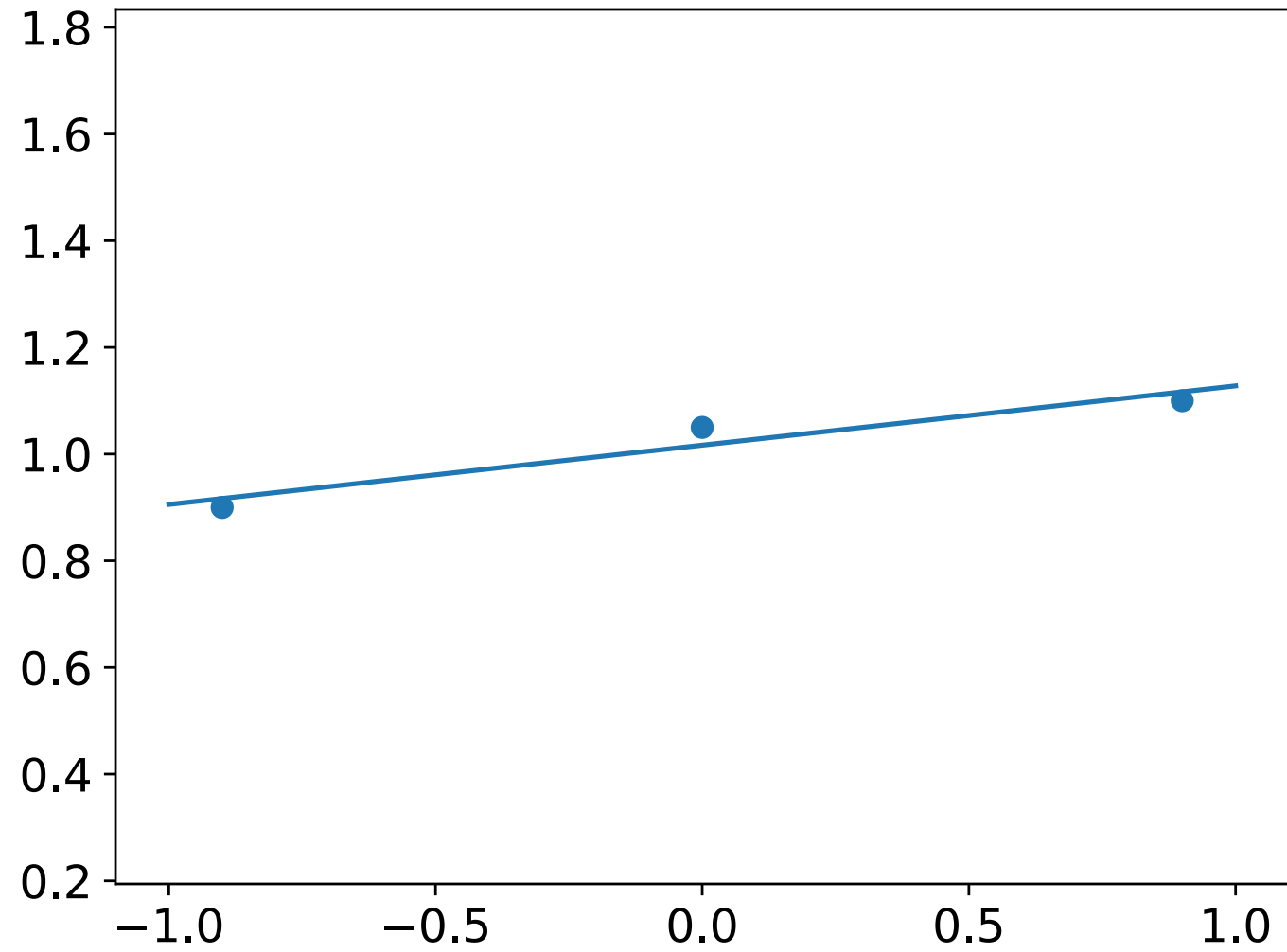
- Use *exact solution* if ~~M~~ is small and N is large
 - fast enough since ~~M~~ is small, accurate solution is worth it since N is large
- Use (variants of) *gradient descent* if ~~M~~ is large
- If ~~M~~ and N both small, both methods are OK

Linear Regression Learning Objectives

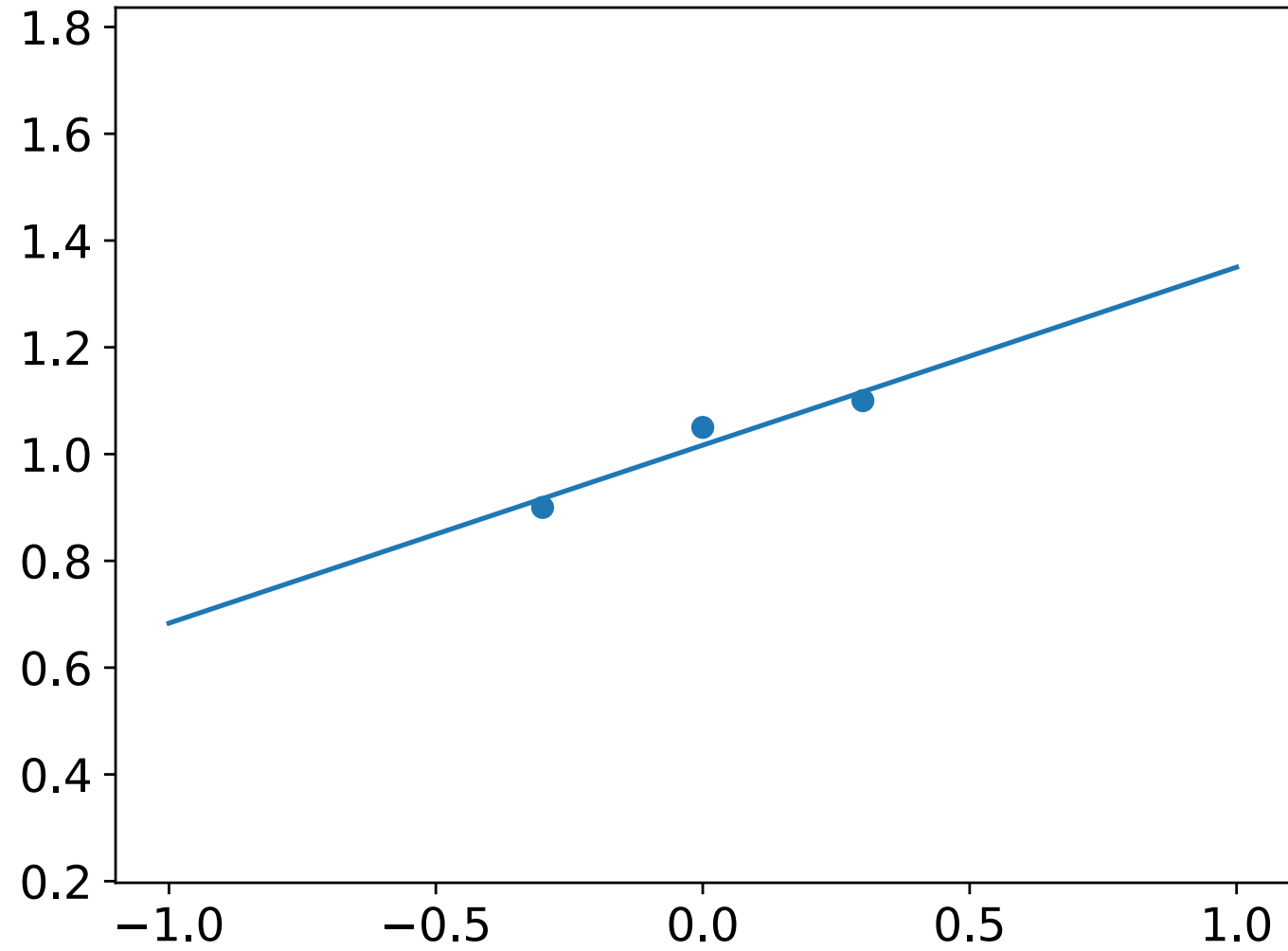
You should be able to...

- Design k-NN Regression and Decision Tree Regression
- Choose a Linear Regression optimization technique that is appropriate for a particular dataset by analyzing the tradeoff of computational complexity vs. convergence speed
- Implement learning for Linear Regression using gradient descent or closed form optimization (with well chosen NumPy calls!)
- Identify situations where least squares regression has exactly one solution or infinitely many solutions

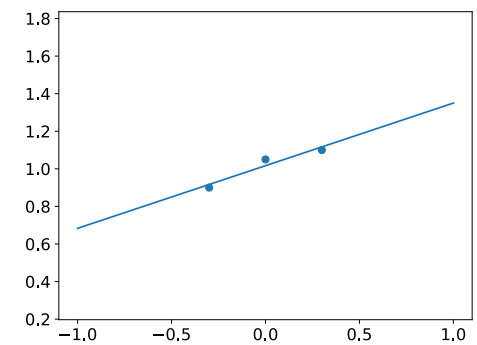
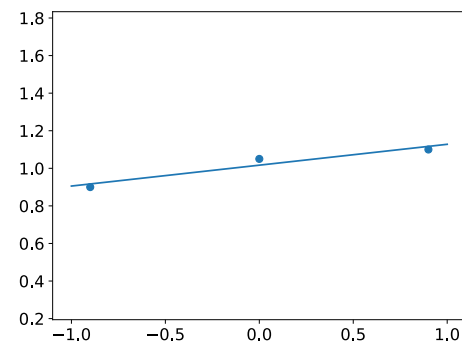
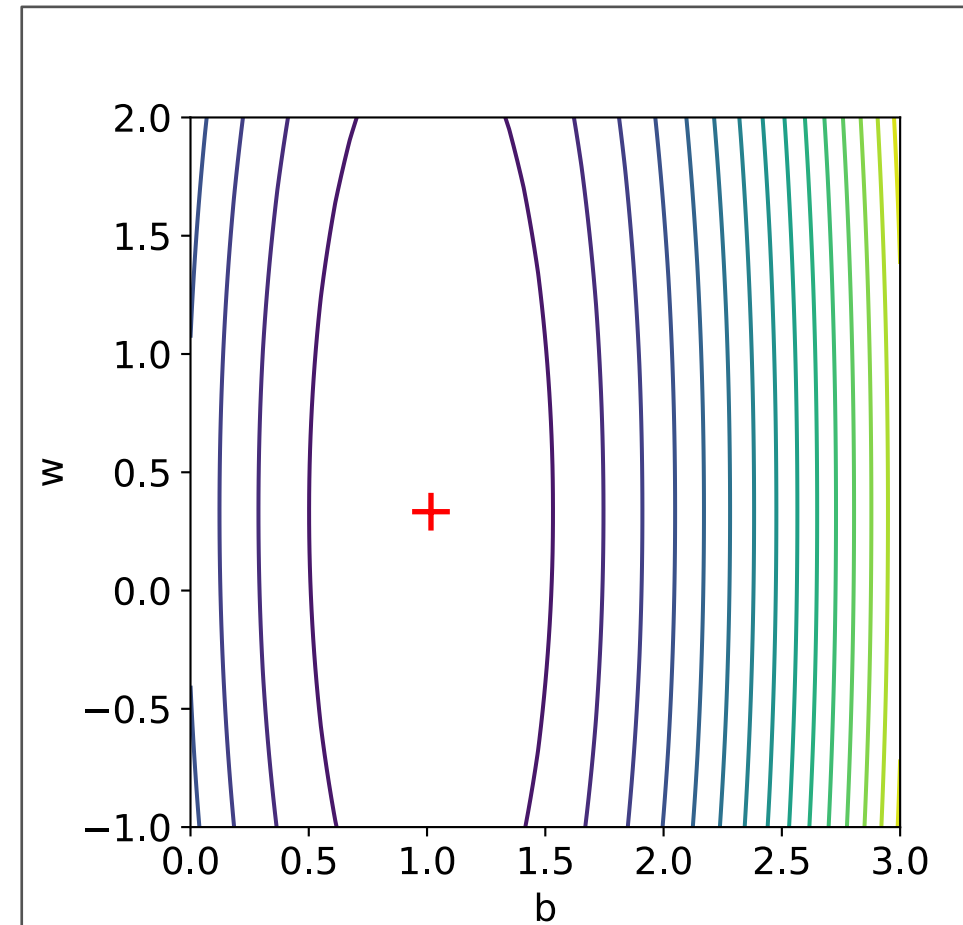
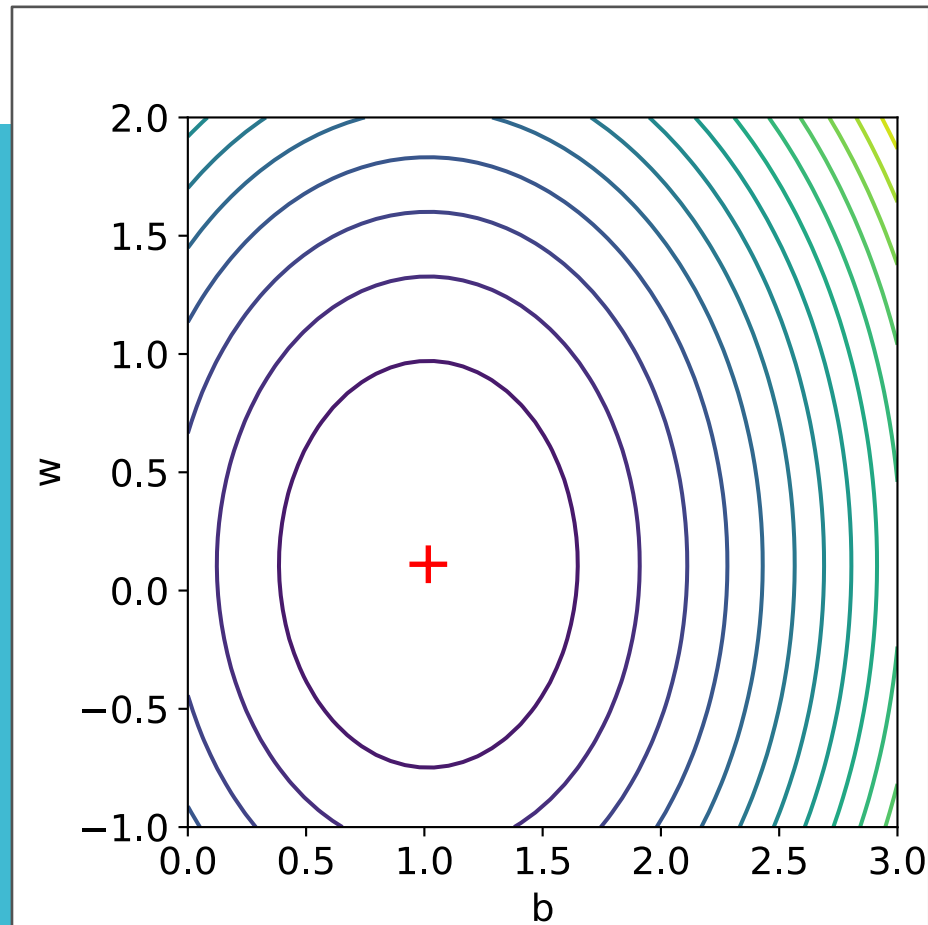
A tale of two datasets



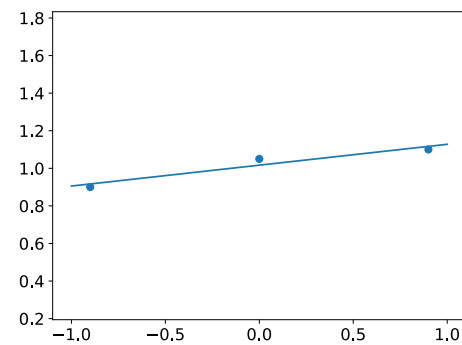
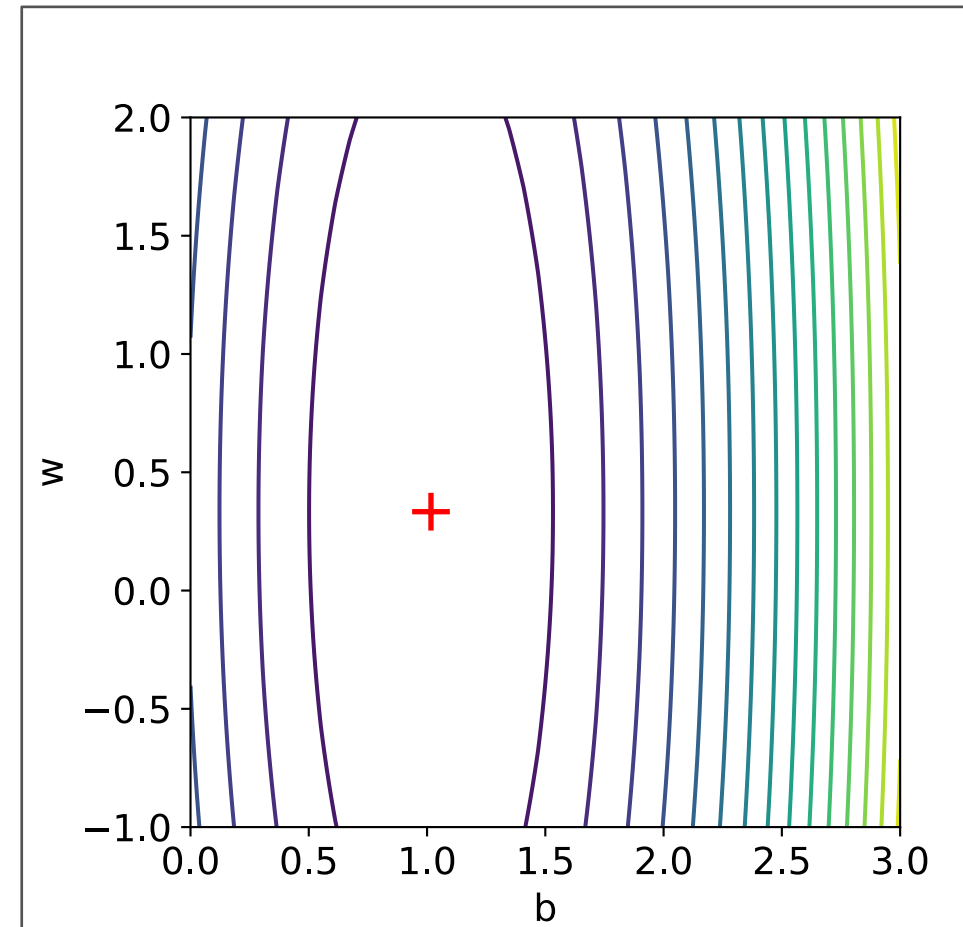
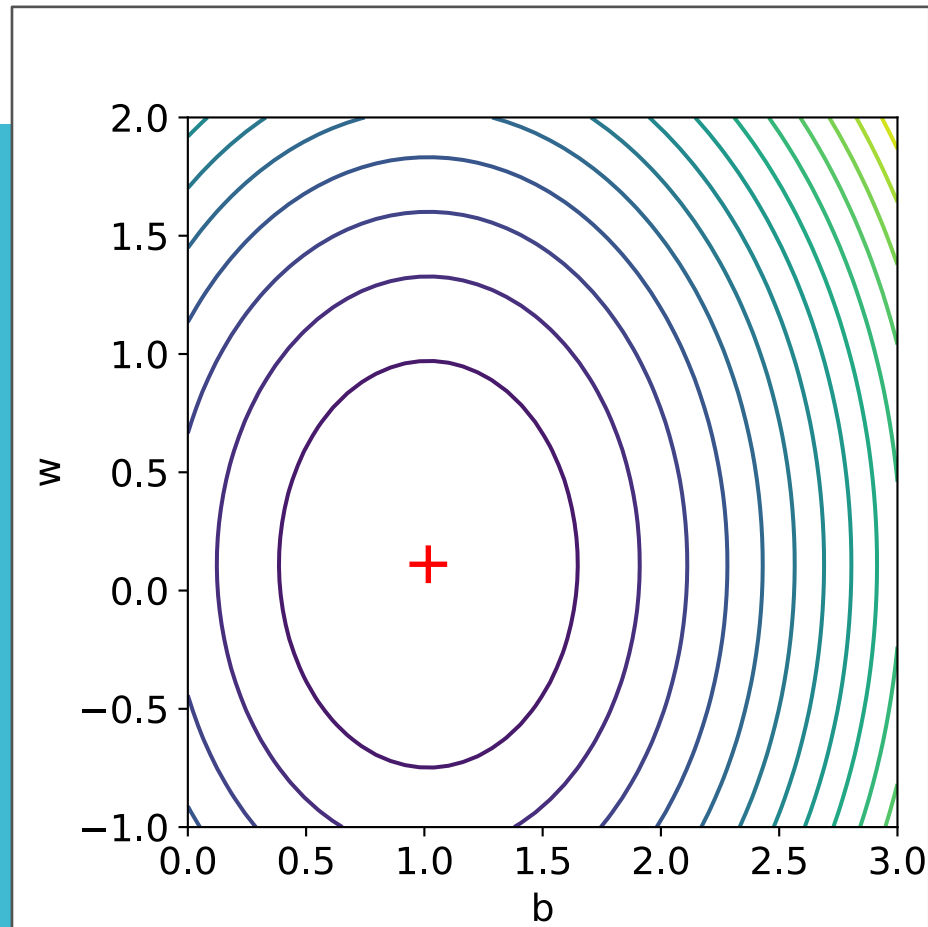
A tale of two datasets



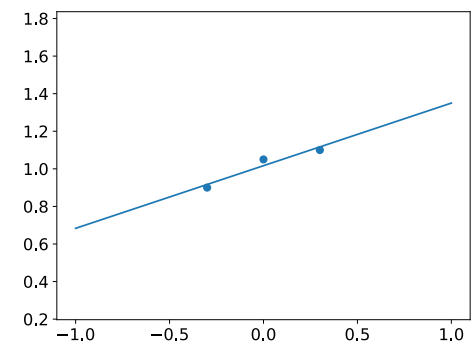
MSE loss



MSE loss

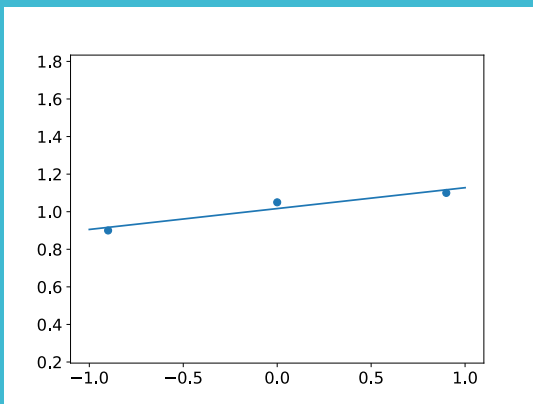


well conditioned

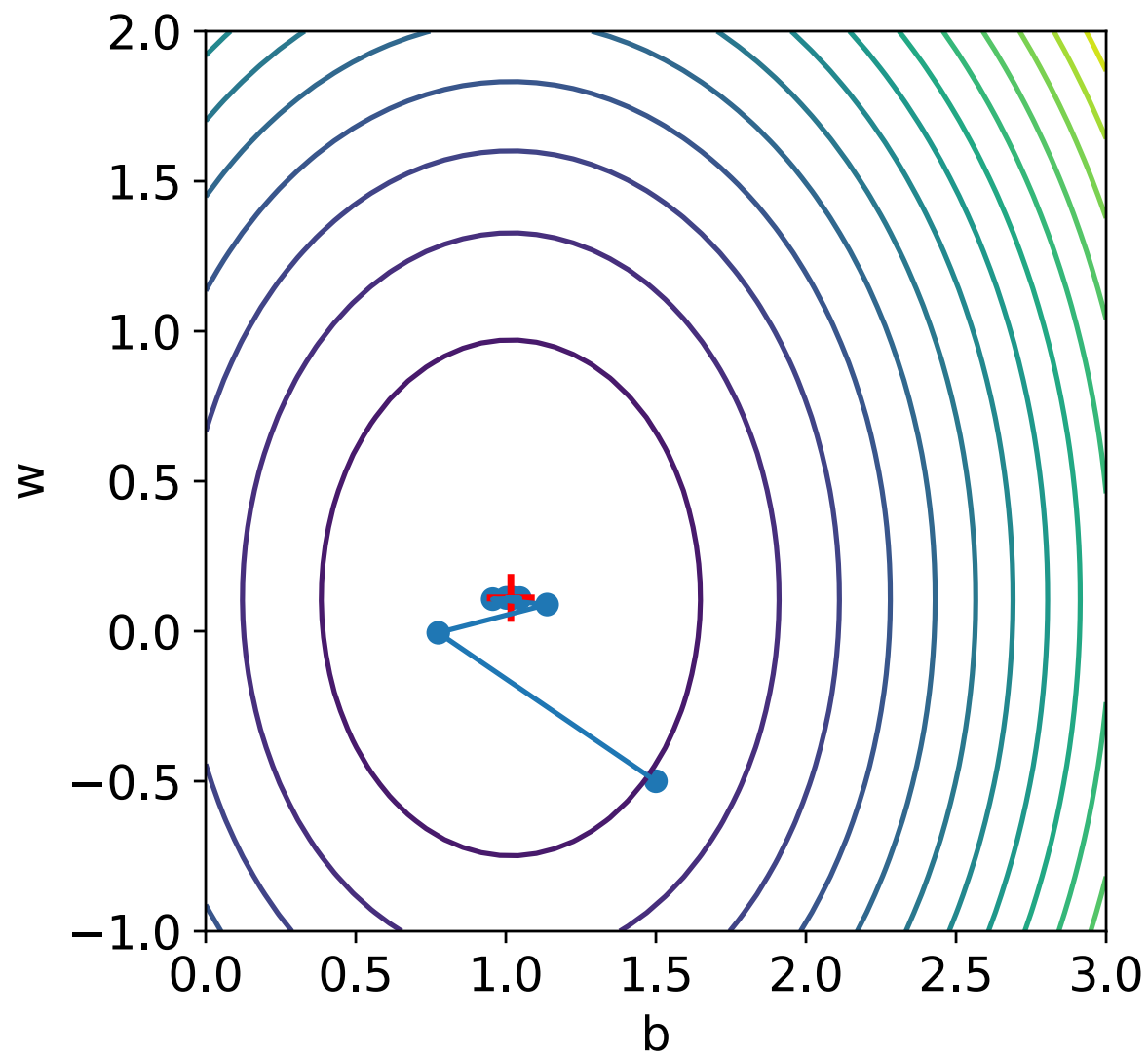


poorly conditioned

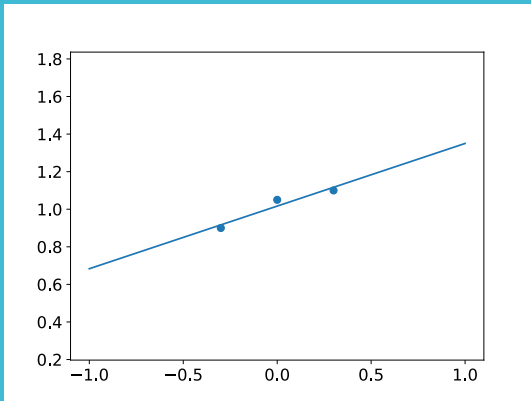
Gradient descent (good conditioning)



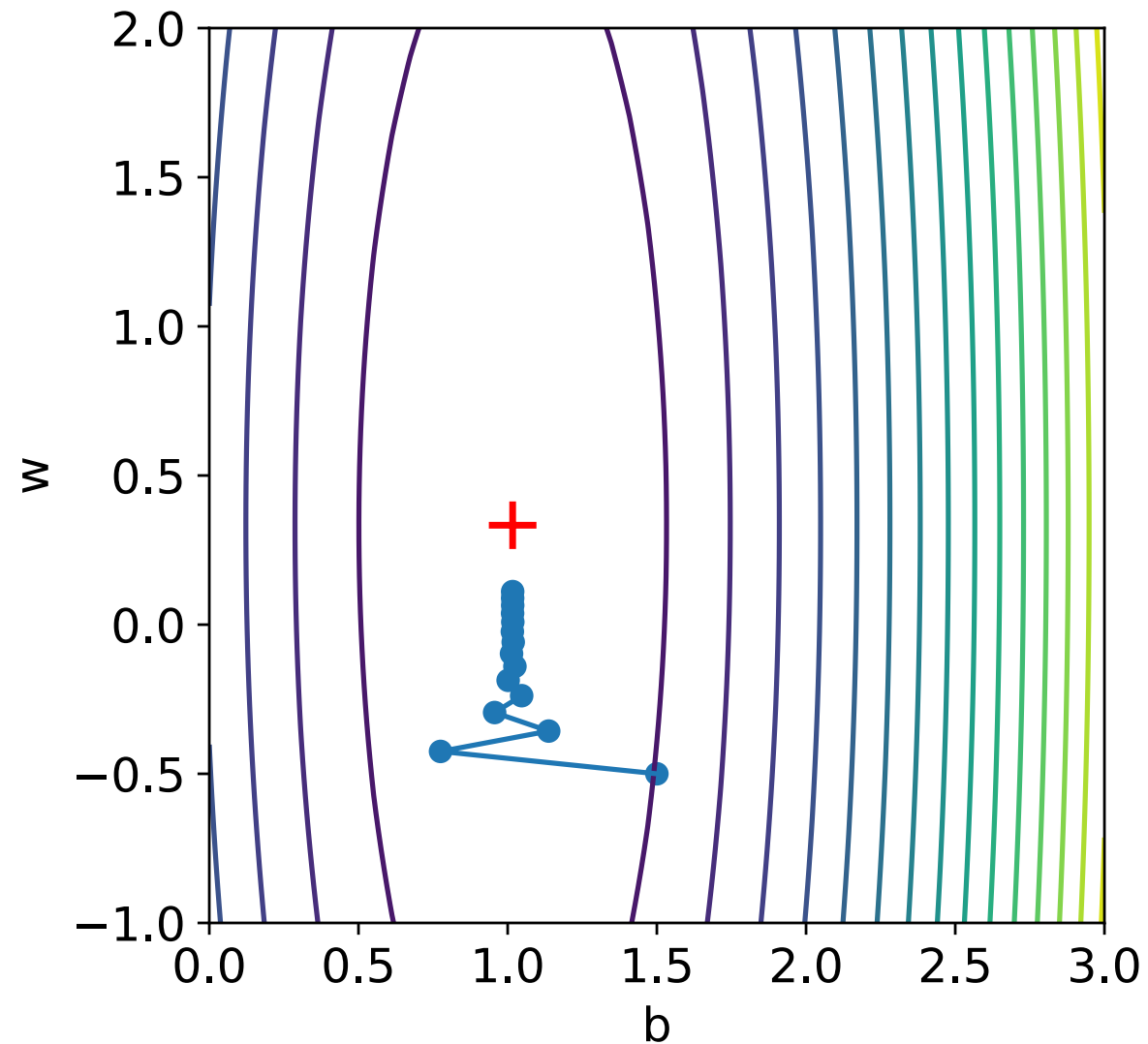
learning rate 0.5



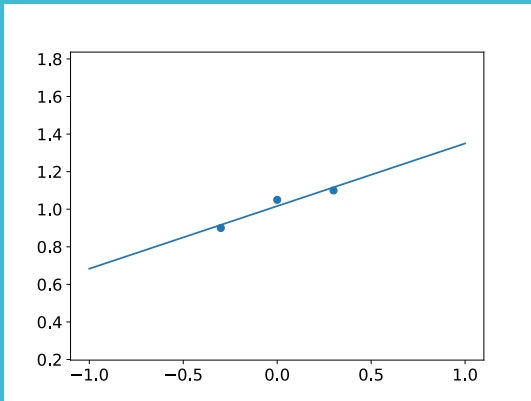
Gradient descent (poor conditioning)



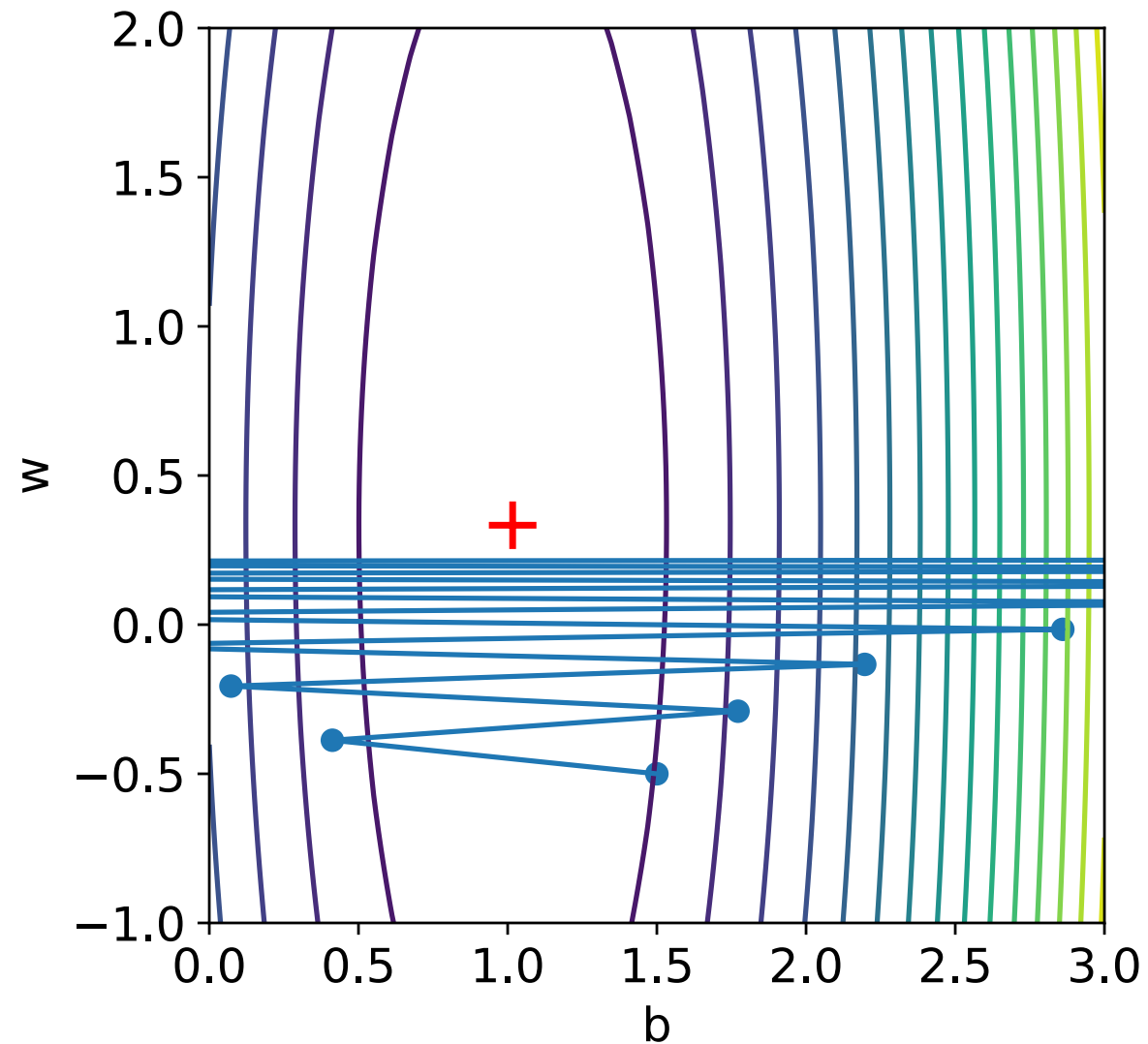
learning rate 0.5



Gradient descent (poor conditioning)



learning rate 0.75



What could we do?

- For horizontal direction (b), want a small learning rate to prevent oscillation
- For vertical direction (w), want a large learning rate to make fast enough progress
- Ideas?

pre scale data — or scale gradient
adapt step size

What if we
could discover
a good scaling
automatically?

- Wouldn't it be great if we could automatically scale down the learning rate in directions where we start to oscillate?
 - then we'd be free to set learning rate high enough to make progress in other dimensions

Momentum

- Exponential moving average

$$\bar{g} \leftarrow \beta \bar{g} + (1 - \beta)g$$

$\bar{g}$ EMA gradient

g gradient

if gradients always point $\uparrow$

if gradients alternate $\uparrow \downarrow$

- Modified gradient descent update

$$\theta \leftarrow \theta - \gamma \bar{g}$$

Gradient descent with momentum

procedure GDLR_M($\mathcal{D}, \boldsymbol{\theta}^{(0)}, \mathbf{g}^{(0)}$)

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}, \bar{\mathbf{g}} \leftarrow \mathbf{g}^{(0)}$

while not converged do

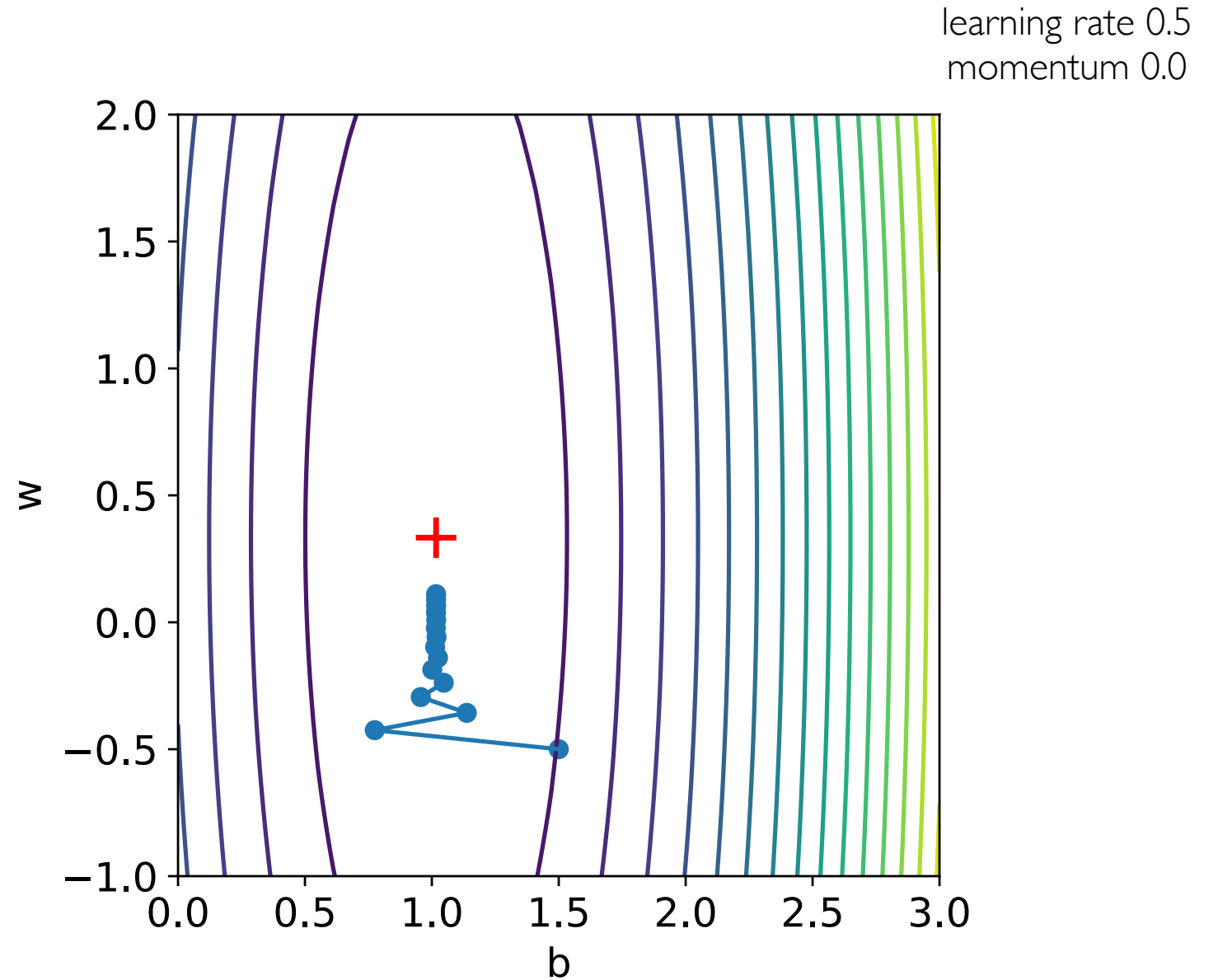
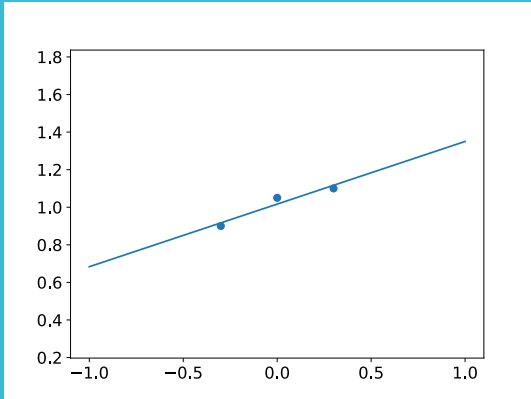
$$\mathbf{g} \leftarrow \sum_{i=1}^N (\boldsymbol{\theta}^\top \mathbf{x}^{(i)} - \mathbf{y}^{(i)}) \mathbf{x}^{(i)}$$

$$\bar{\mathbf{g}} \leftarrow \beta \bar{\mathbf{g}} + (1 - \beta) \mathbf{g}$$

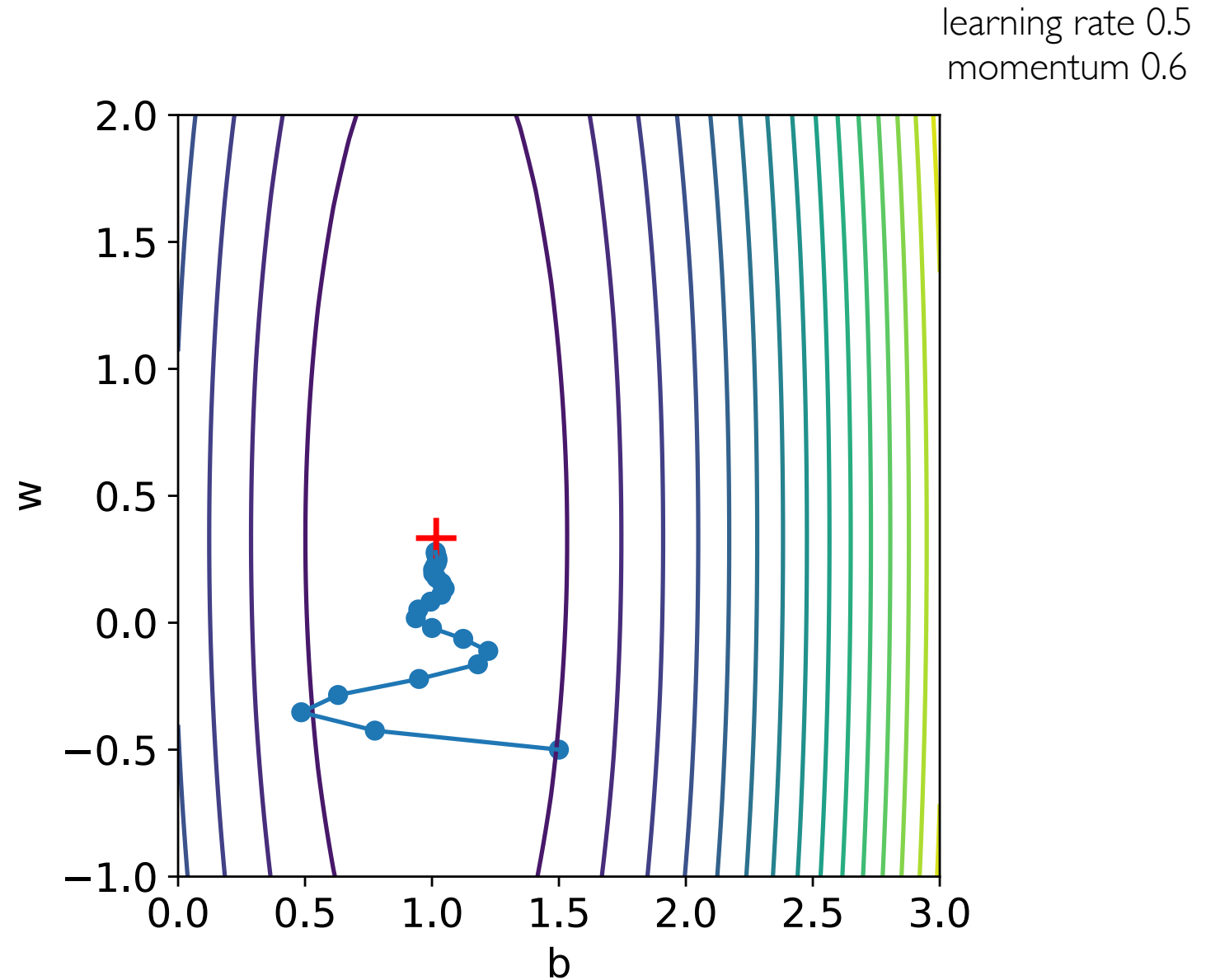
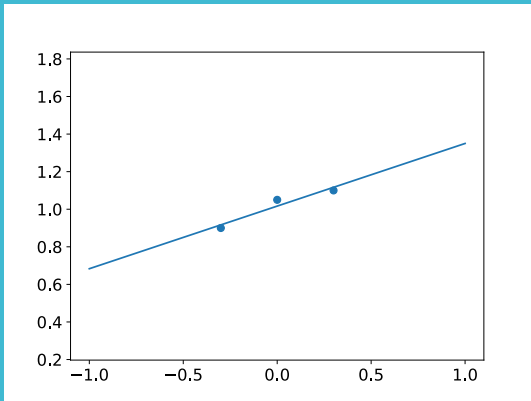
$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \gamma \bar{\mathbf{g}}$$

return $\boldsymbol{\theta}$

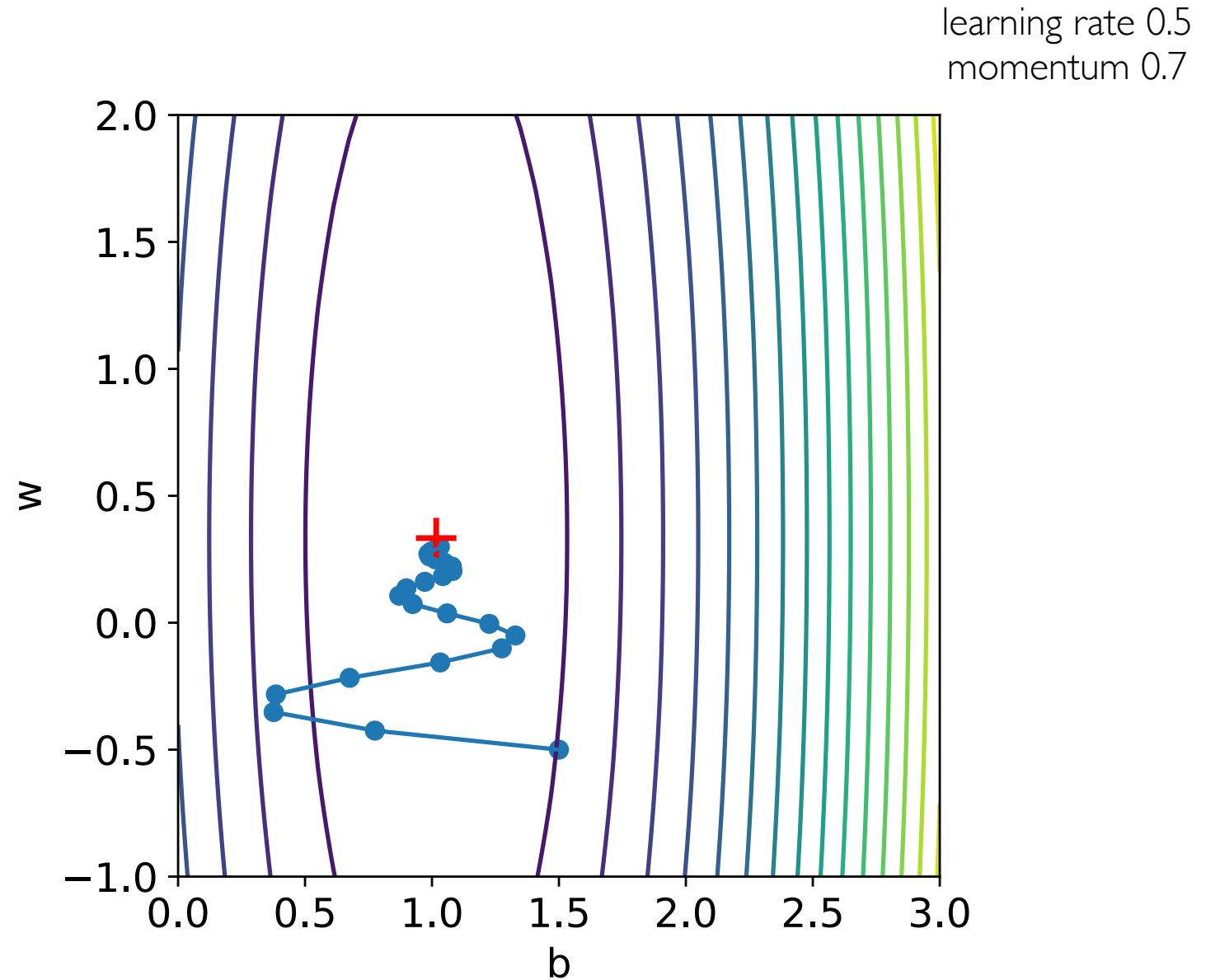
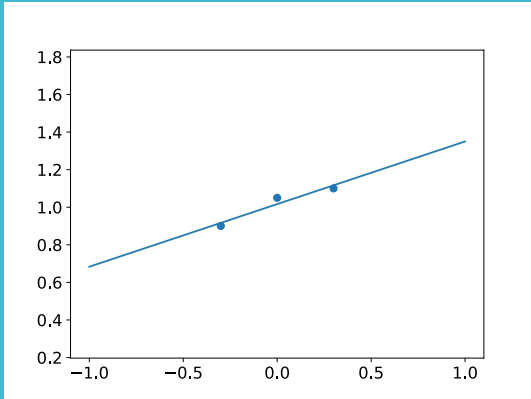
Gradient descent + momentum (poor conditioning)



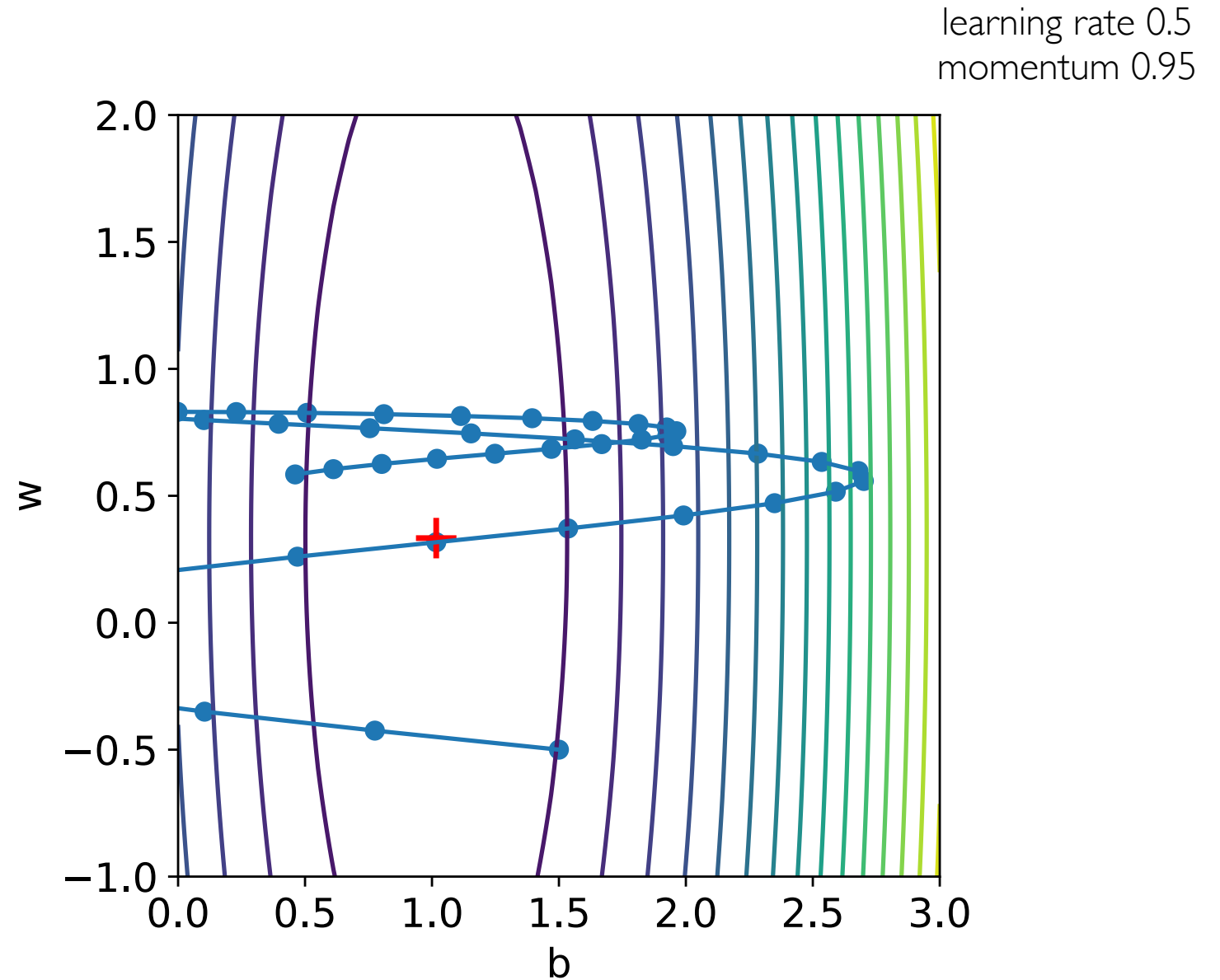
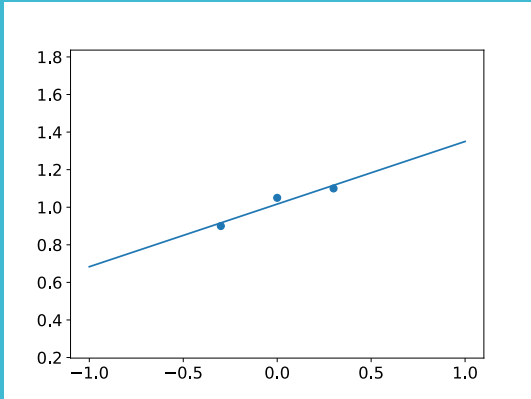
Gradient descent + momentum (poor conditioning)



Gradient descent + momentum (poor conditioning)



Gradient descent + momentum (poor conditioning)



Preview:
nonconvex

