GENERAL ASSEMBLY

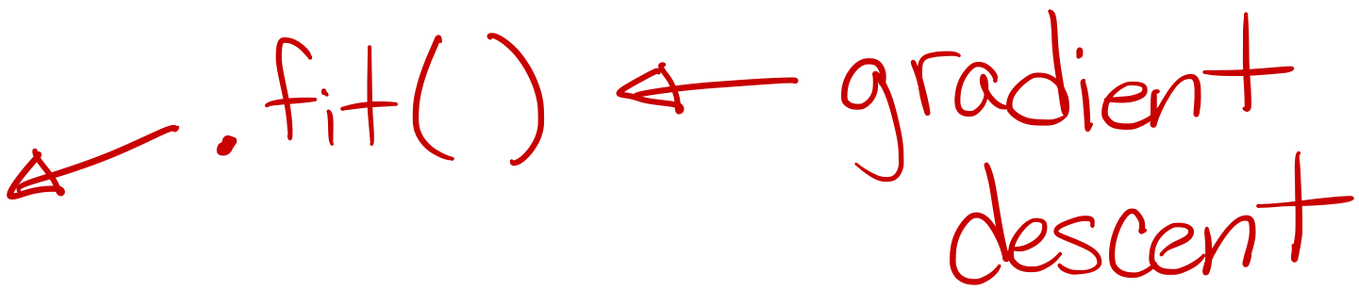
GRADIENT DESCENT

Matt Brems

DSI+

DATA SCIENCE PROCESS

1. Define the problem.
2. Obtain the data.
3. Explore the data.
4. Model with data.
5. Evaluate the model.
6. Answer the problem.



GRADIENT DESCENT

- Recall: What is a loss function?

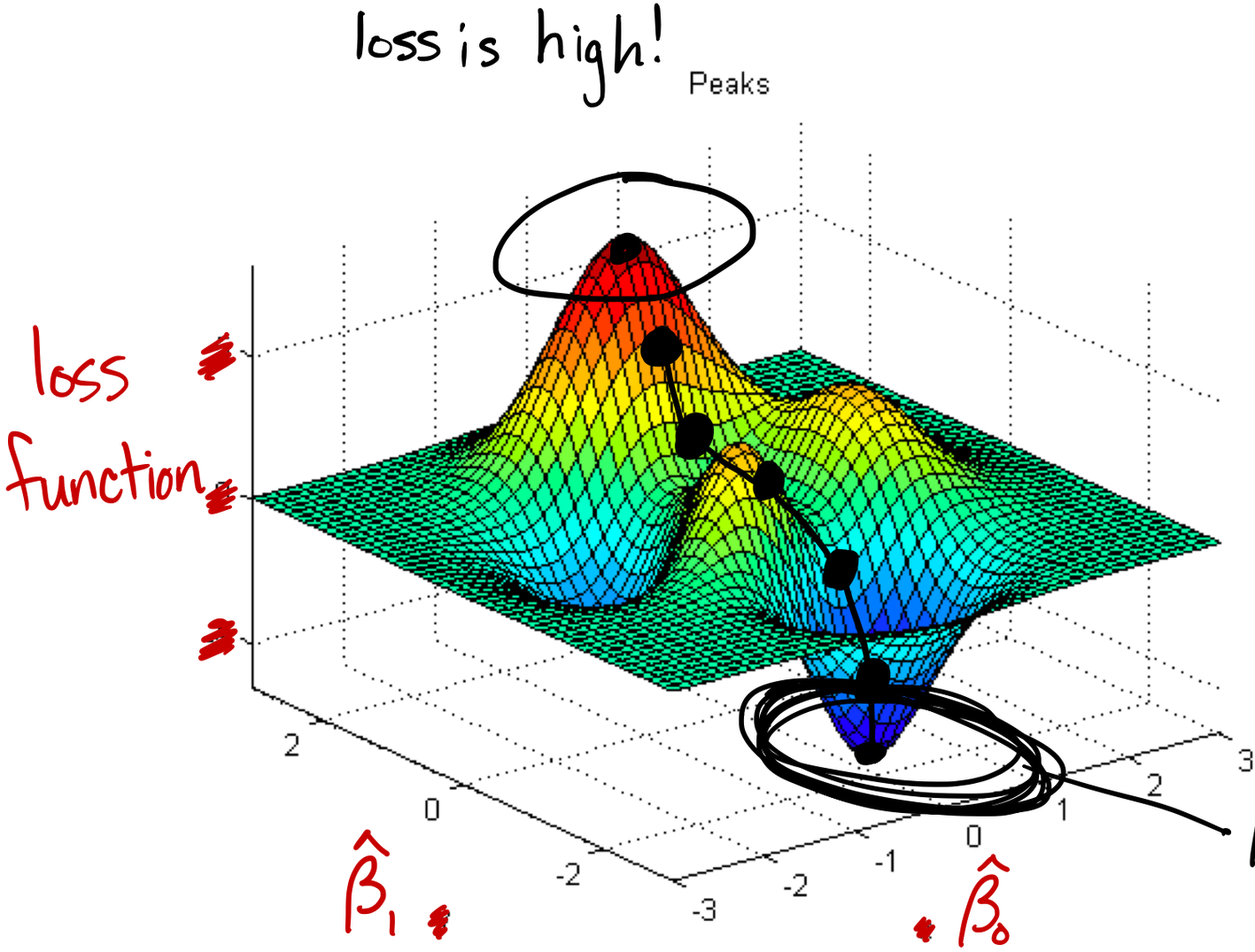
quantify/measure the goodness/badness of our model when we "fit" a model, this helps us determine the right parameters

- Recall: What are common loss functions we've used?

$$MSE = \frac{1}{n}\sum(y_i - \hat{y}_i)^2$$

$$LASSO/Ridge = \sum(y_i - \hat{y}_i)^2 + \alpha \sum |\beta_j|$$

LOSS FUNCTION



- What do we do with this?
  Minimize this
- How do we do it?
  Snowboarding

GRADIENT DESCENT

- Gradient descent is:
  - an iterative method → step by step
  - used to identify the optimal value of parameters
  - by optimizing an objective function.
    - ↳ minimize
    - ↳ loss

. fit

GRADIENT DESCENT

- Gradient descent is:
  - an iterative method
  - used to identify the optimal value of parameters
  - by optimizing an objective function.

- Algorithm Sketch:
  1. Start by making a guess for the optimal parameter value.
  2. Calculate the loss given that parameter value.
  3. Update guess to decrease loss.
  4. Keep going until loss is “sufficiently minimized.”

tol = 0.0001

MATH REPRESENTATION

- Goal: Find the best possible value for $\hat{\beta}_1$.

$$Y = \beta_1 X$$

assume $\beta_0 = 0$

MATH REPRESENTATION

- Goal: Find the best possible value for $\hat{\beta}_1$.

:= "set equal to"

iterative

$$\hat{\beta}_{1,i+1} := \hat{\beta}_{1,i} - \alpha \left[\frac{\partial L}{\partial \beta_1} \right]$$

$(i+1)^{st}$ guess for $\hat{\beta}_1$

i^{th} guess for $\hat{\beta}_1$

MATH REPRESENTATION

- Goal: Find the best possible value for $\hat{\beta}_1$.

$$\hat{\beta}_{1,i+1} := \hat{\beta}_{1,i} - \alpha\left[\frac{\partial L}{\partial \beta_1}\right]$$

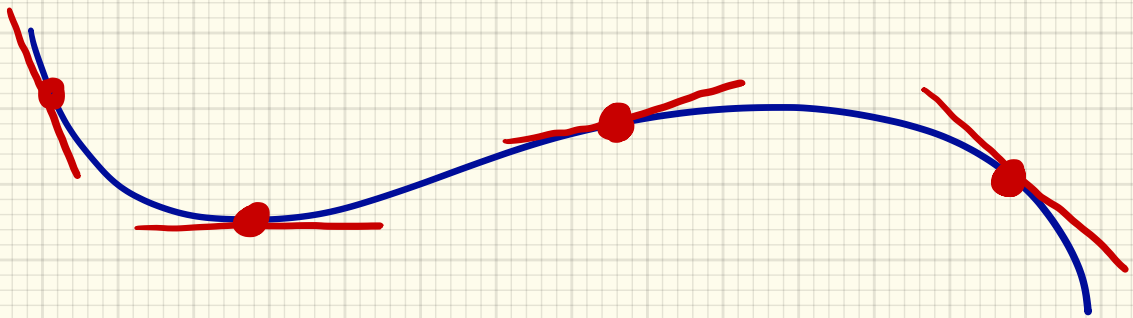
- α: Learning Rate
  - Controls how fast we move with each step.

Constant

Hyperparameter

slope of a line

derivative: the slope of the curve at a given point



MATH REPRESENTATION

- Goal: Find the best possible value for $\hat{\beta}_1$.

$$\hat{\beta}_{1,i+1} := \hat{\beta}_{1,i} - \alpha \left[\frac{\partial L}{\partial \beta_1}\right]$$

$(i+1)^{st}$ guess for β_1 ← $\hat{\beta}_{1,i+1}$

step of size α → in this direction

i^{th} guess for $\hat{\beta}_1$

- α: Learning Rate
  - Controls how fast we move with each step.

- $\frac{\partial L}{\partial \beta_1}$: Gradient of Loss Function with respect to β_1.
  - Tells us direction of steepest slope. (Gravity!)

MATH REPRESENTATION

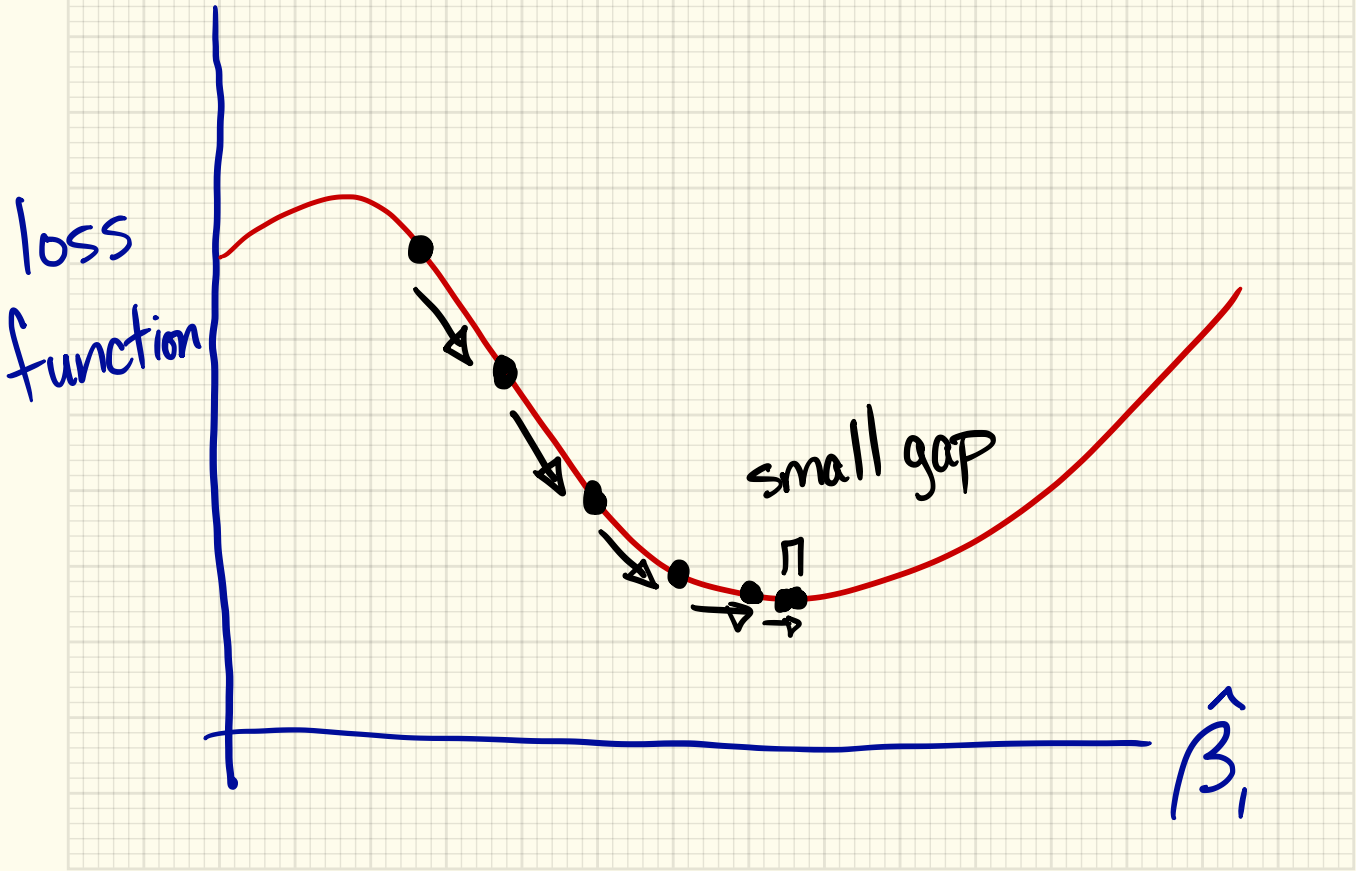
- Goal: Find the best possible value for $\hat{\beta}_1$.

$$\hat{\beta}_{1,i+1} \coloneqq \hat{\beta}_{1,i} - \alpha \left[\frac{\partial L}{\partial \beta_1}\right]$$

- Keep going until $\left|\hat{\beta}_{1,i+1} - \hat{\beta}_{1,i}\right|$ is sufficiently small. (Why?)

– Slope is approximately zero.

– We hope we are at the global min.


loss function
small gap
$\hat{\beta}_1$

GRADIENT DESCENT ALGORITHM

- Step 1: Instantiate model.

GRADIENT DESCENT ALGORITHM

- Step 1: Instantiate model.
- Step 2: Select a “learning rate” α.

GRADIENT DESCENT ALGORITHM

- Step 1: Instantiate model.
- Step 2: Select a "learning rate" α.
- Step 3: Select a "starting point" $\hat{\beta}_{1,0}$.

GRADIENT DESCENT ALGORITHM

- Step 1: Instantiate model.
- Step 2: Select a "learning rate" α.
- Step 3: Select a "starting point" $\hat{\beta}_{1,0}$.
- Step 4: Calculate the gradient of the loss function w.r.t. parameter $\frac{\partial L}{\partial \beta_1}$.

↳ direction of steepest slope

GRADIENT DESCENT ALGORITHM

- Step 1: Instantiate model.
- Step 2: Select a "learning rate" α.
- Step 3: Select a "starting point" $\hat{\beta}_{1,0}$.
- Step 4: Calculate the gradient of the loss function w.r.t. parameter $\frac{\partial L}{\partial \beta_1}$.
- Step 5: Calculate $\hat{\beta}_{1,i+1} = \hat{\beta}_{1,i} - \alpha \left[\frac{\partial L}{\partial \beta_1}\right]$.

GRADIENT DESCENT ALGORITHM

- Step 1: Instantiate model.
- Step 2: Select a "learning rate" α.
- Step 3: Select a "starting point" $\hat{\beta}_{1,0}$.
- Step 4: Calculate the gradient of the loss function w.r.t. parameter $\frac{\partial L}{\partial \beta_1}$.
- Step 5: Calculate $\hat{\beta}_{1,i+1} = \hat{\beta}_{1,i} - \alpha \left[\frac{\partial L}{\partial \beta_1}\right]$.
- Step 6: Check value of $\left|\hat{\beta}_{1,i+1} - \hat{\beta}_{1,i}\right|$.

GRADIENT DESCENT ALGORITHM

- Step 1: Instantiate model.
- Step 2: Select a "learning rate" α.
- Step 3: Select a "starting point" $\hat{\beta}_{1,0}$.
- Step 4: Calculate the gradient of the loss function w.r.t. parameter $\frac{\partial L}{\partial \beta_1}$.
- Step 5: Calculate $\hat{\beta}_{1,i+1} = \hat{\beta}_{1,i} - \alpha \left[\frac{\partial L}{\partial \beta_1}\right]$.
- Step 6: Check value of $\left|\hat{\beta}_{1,i+1} - \hat{\beta}_{1,i}\right| < tol$
- Step 7: Repeat steps 4 through 6 until "stopping condition" is met.

$tol = 0.0001$

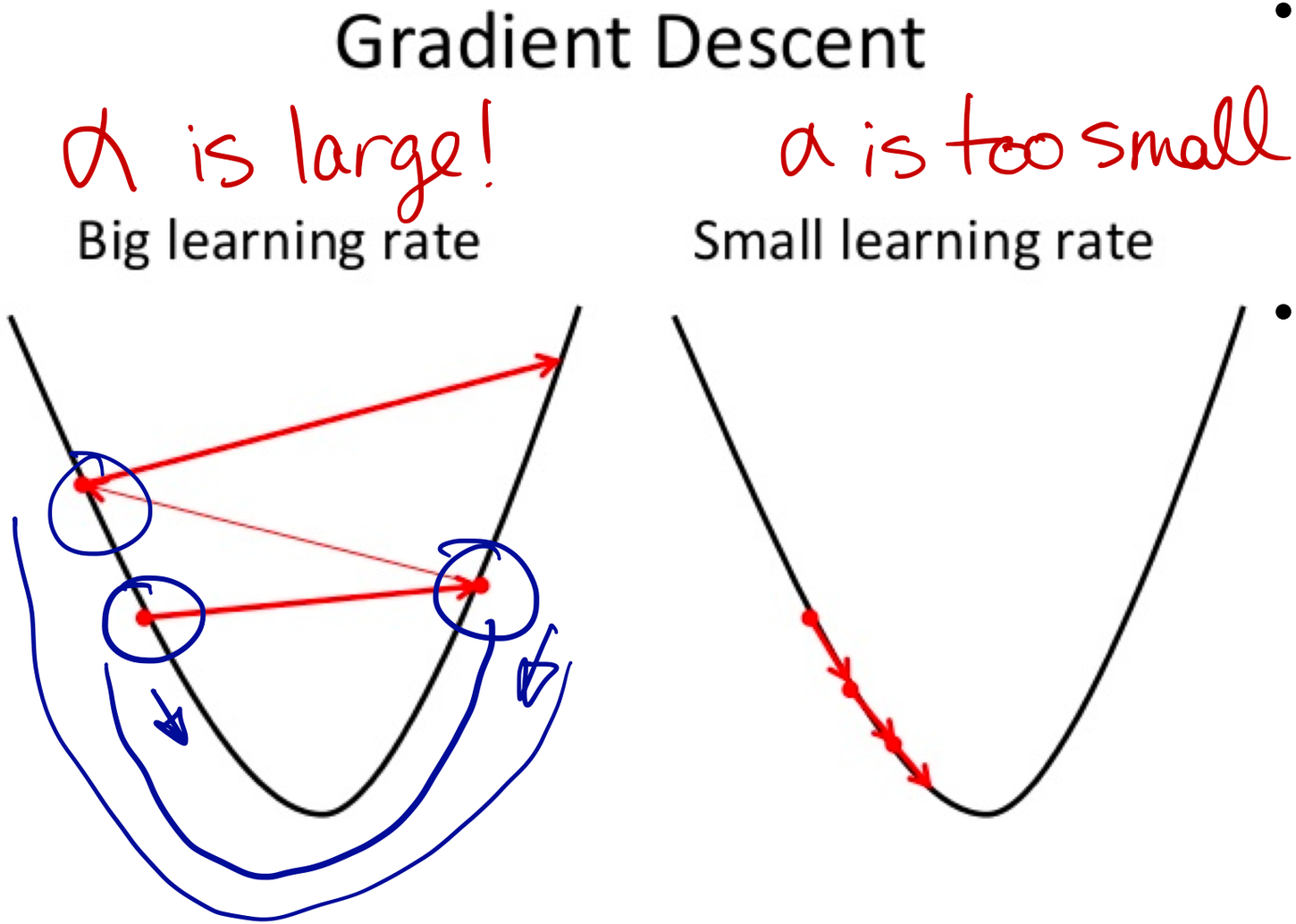
GRADIENT DESCENT ALGORITHM

- Step 1: Instantiate model.
- Step 2: Select a "learning rate" α.
- Step 3: Select a "starting point" $\hat{\beta}_{1,0}$.
- Step 4: Calculate the gradient of the loss function w.r.t. parameter $\frac{\partial L}{\partial \beta_1}$.
- Step 5: Calculate $\hat{\beta}_{1,i+1} = \hat{\beta}_{1,i} - \alpha \left[\frac{\partial L}{\partial \beta_1}\right]$.
- Step 6: Check value of $\left|\hat{\beta}_{1,i+1} - \hat{\beta}_{1,i}\right|$.
- Step 7: Repeat steps 4 through 6 until "stopping condition" is met.
- Step 8: $\hat{\beta}_1 := \hat{\beta}_{1,n}$, where n is the number of iterations of gradient descent.

INTUITIVE RECAP

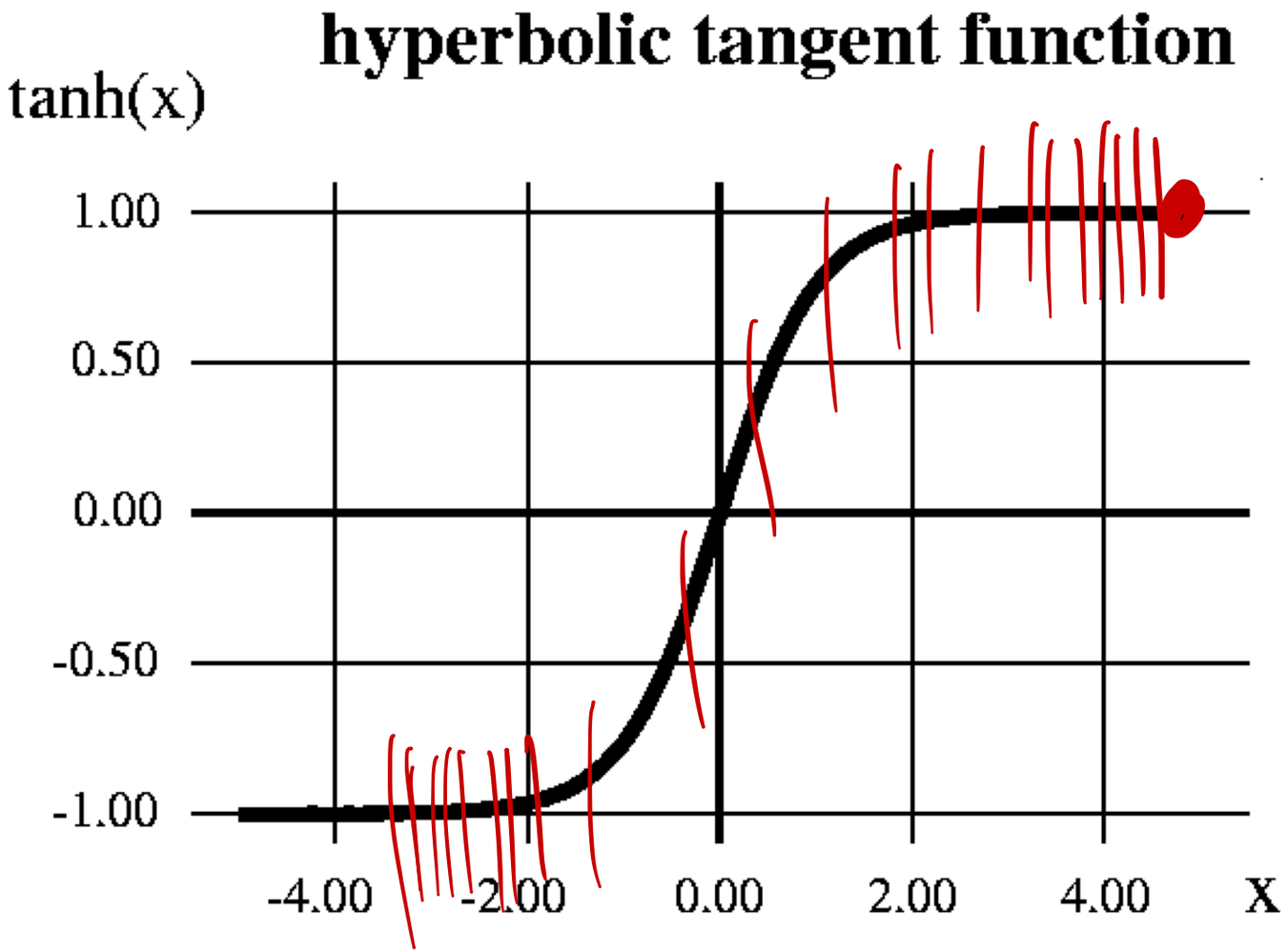
- When fitting a model, we:
  - Identify some loss function to optimize.
  - Pick a first guess.
  - Take a step of fixed size in the “best” direction.
  - Keep going until we’ve found the minimum of our loss function!

POTENTIAL PITFALLS



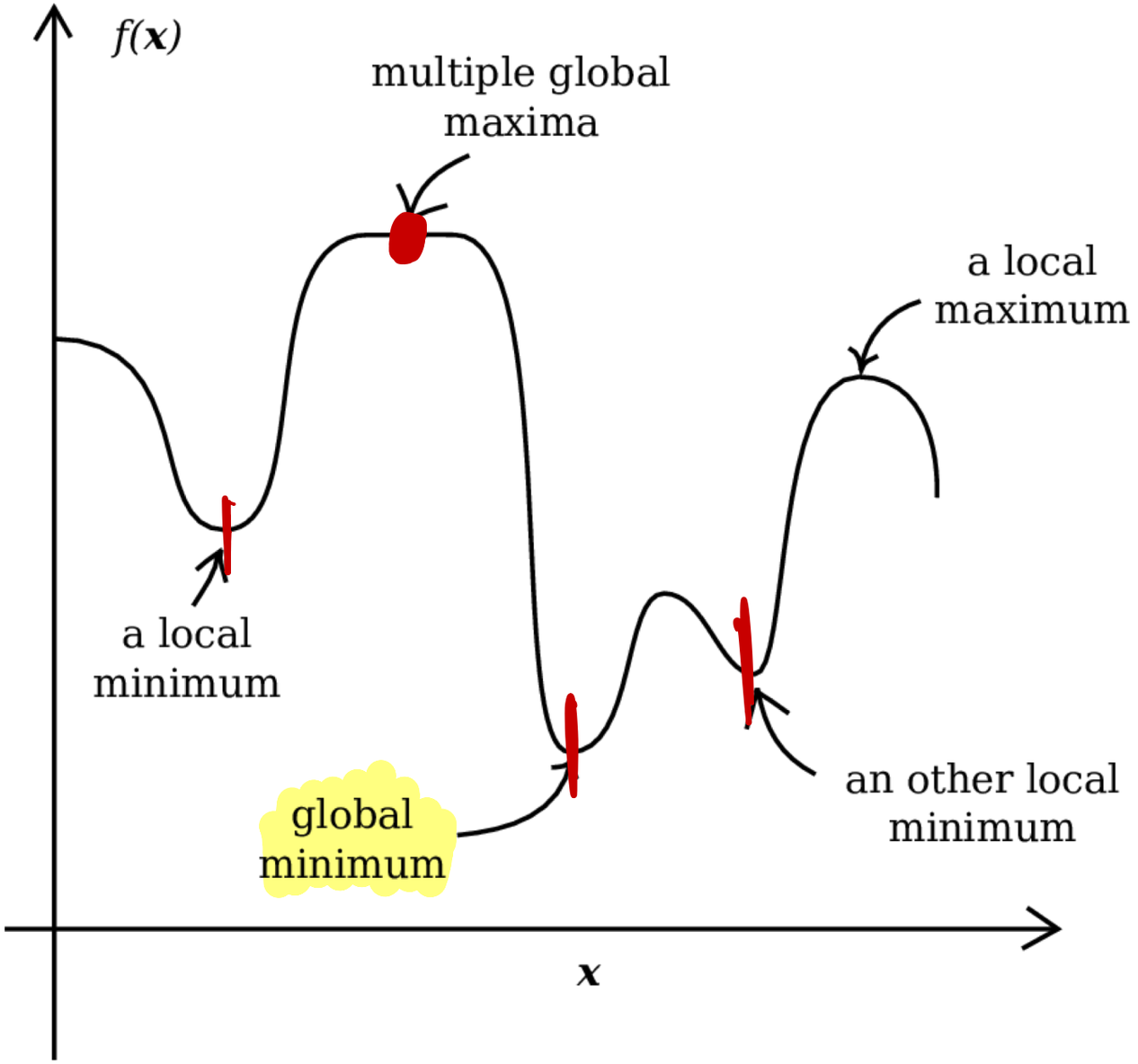
- If our step size is too big, we may never converge!
- If our step size is too small, it may take a very long time for us to converge!

POTENTIAL PITFALLS

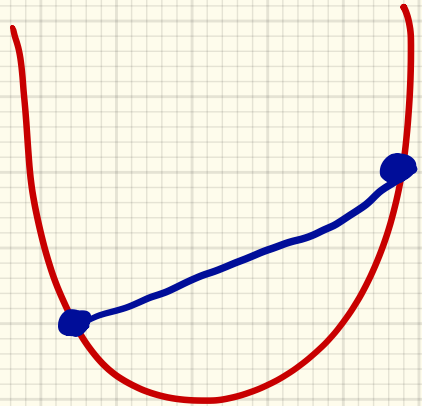


- Depending on the shape of the loss function, gradient (slope of the curve) may be close to zero, meaning that learning occurs very slowly.

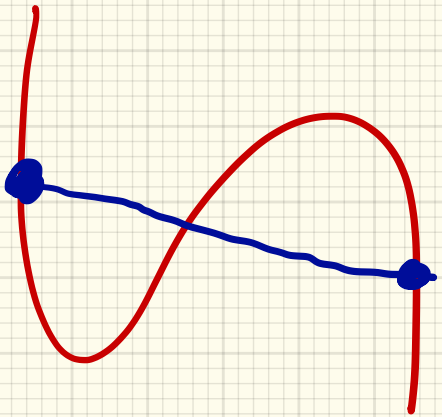
POTENTIAL PITFALLS



- Depending on the shape of the loss function, we may converge to a local optimum.
- This is one reason we attempt to choose convex loss functions.
- Shockingly, this isn't a major problem.



Convex



Concave

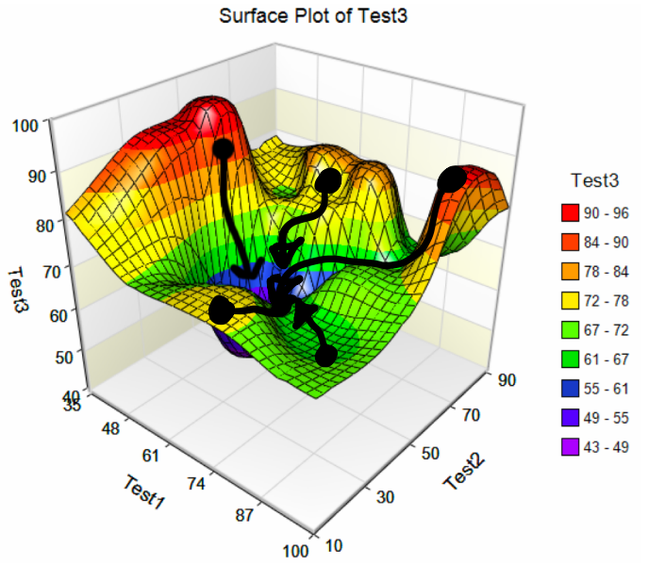
SOLUTION 1: ADAPTIVE GRADIENT DESCENT

- One way to attempt to protect against some of these pitfalls is to use **adaptive gradient descent**, which means that, at each step, we draw α from some distribution so that we aren't taking a step of fixed size.
  - There are many different flavors of "adaptive" gradient descent algorithms for various purposes.

$$\alpha \sim N(1, 0.3)$$

SOLUTION 2: CHANGE STARTING POINTS

- Another way to attempt to protect against some of these pitfalls is to change the starting points of the algorithm.



- Intuitively, if we get nearly identical results a few times, we can be more confident that we've arrived at the global optimum.

MACHINE LEARNING = GRADIENT DESCENT

Tim Hopper
@tdhopper

Machine learning path to fame:
1) Find learning algorithm requiring optimization
2) Apply gradient descent
3) Give this algorithm cool name