GENERAL ASSEMBLY

GRADIENT DESCENT

Matt Brems

Data Science Immersive, GA DC

GRADIENT DESCENT

- Recall: What is a loss function?

GRADIENT DESCENT

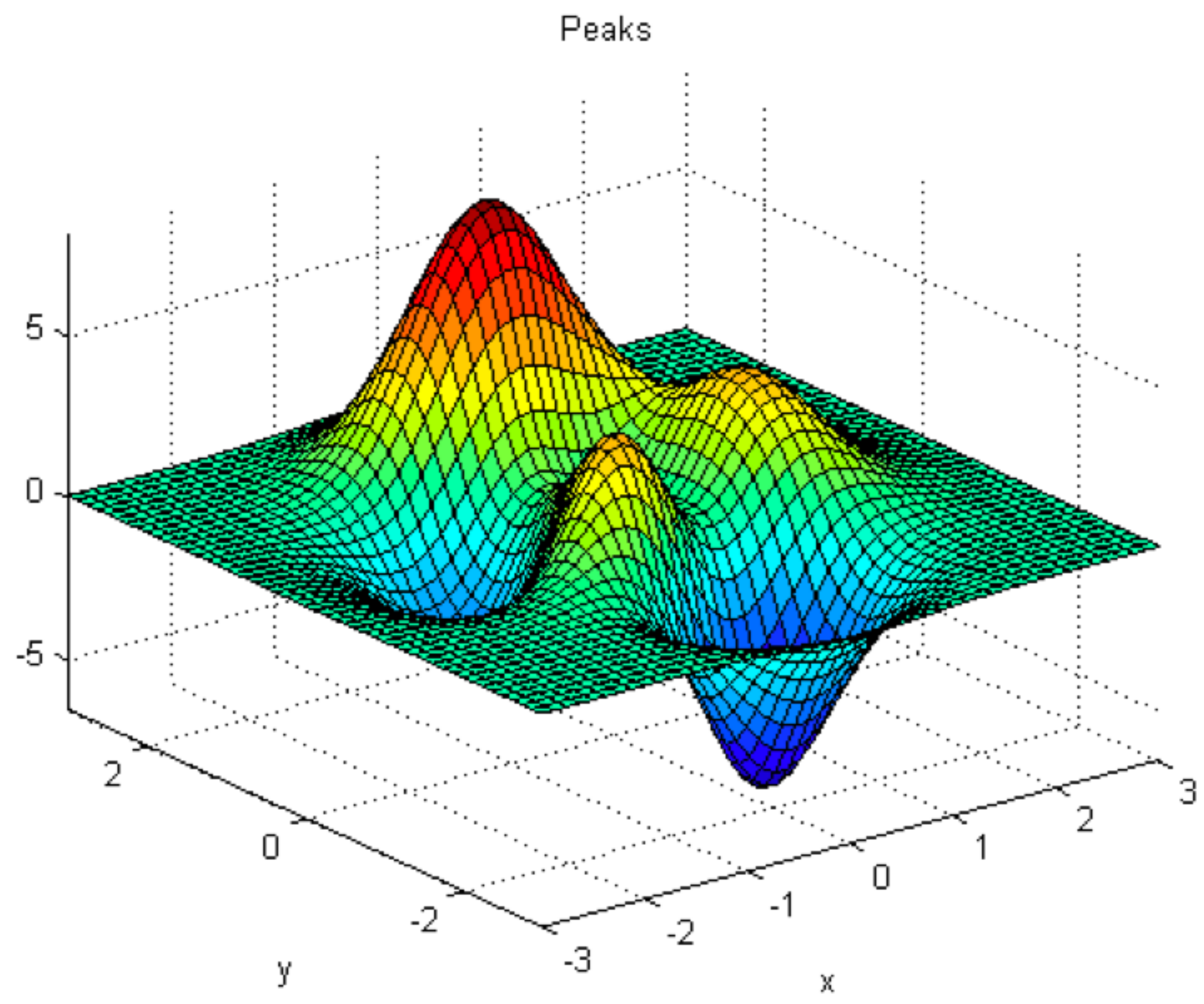
- Recall: What is a loss function?

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

GRADIENT DESCENT

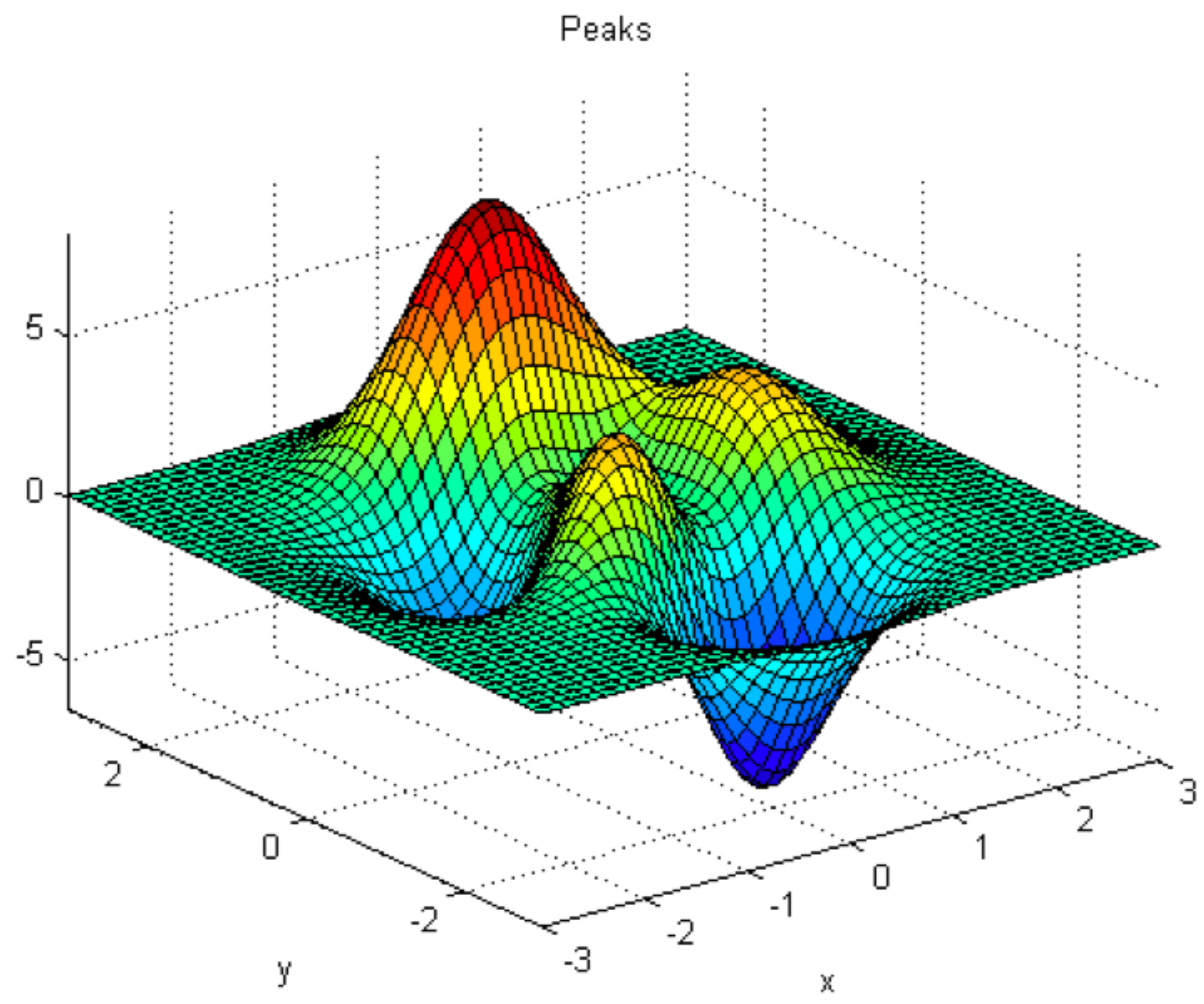
- Recall: What is a loss function?
- Recall: What are common loss functions we’ve used?
- Recall: What purposes do our loss function serve?

LOSS FUNCTION



- What do we do with this?

LOSS FUNCTION



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

GRADIENT DESCENT

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

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.”

MATH REPRESENTATION

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

MATH REPRESENTATION

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

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

MATH REPRESENTATION

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

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

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

MATH REPRESENTATION

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

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

- 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 $\hat{\beta}_{1,i+1} - \hat{\beta}_{1,i}$ is sufficiently small. (Why?)

GRADIENT DESCENT ALGORITHM

- Step 0: Instantiate model.

GRADIENT DESCENT ALGORITHM

- Step 0: Instantiate model.
- Step 1: Select a “learning rate” α.

GRADIENT DESCENT ALGORITHM

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

GRADIENT DESCENT ALGORITHM

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

GRADIENT DESCENT ALGORITHM

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

GRADIENT DESCENT ALGORITHM

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

GRADIENT DESCENT ALGORITHM

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

GRADIENT DESCENT ALGORITHM

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

LOGISTIC REGRESSION

- Let’s walk through an example of using gradient descent to optimize parameters for logistic regression.

LOGISTIC REGRESSION LOSS FUNCTION

- What do we want our loss function to look like in logistic regression?

LOGISTIC REGRESSION LOSS FUNCTION

- The loss function used in logistic regression is called the "cross-entropy."

$$L(y_i, \hat{y}_i) = y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i)$$

- Why would this be a good choice of loss function?

LOGISTIC REGRESSION MODEL

- Suppose I fit a model with two slopes and a y-intercept: $\beta_0 + \beta_1 x_1 + \beta_2 x_2$

GRADIENT DESCENT ALGORITHM

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

LOGISTIC REGRESSION MODEL

- Suppose I fit a model with two slopes and a y-intercept: $\beta_0 + \beta_1 x_1 + \beta_2 x_2$

- Instantiate the model. ($\hat{\beta}_0$, $\hat{\beta}_1$, and $\hat{\beta}_2$ are empty placeholders.)

GRADIENT DESCENT ALGORITHM

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

LOGISTIC REGRESSION MODEL

- Suppose I fit a model with two slopes and a y-intercept: $\beta_0 + \beta_1 x_1 + \beta_2 x_2$
- Instantiate the model. ($\hat{\beta}_0$, $\hat{\beta}_1$, and $\hat{\beta}_2$ are empty placeholders.)
- Pick α.

GRADIENT DESCENT ALGORITHM

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

LOGISTIC REGRESSION MODEL

- Suppose I fit a model with two slopes and a y-intercept: $\beta_0 + \beta_1 x_1 + \beta_2 x_2$
- Instantiate the model. ($\hat{\beta}_0$, $\hat{\beta}_1$, and $\hat{\beta}_2$ are empty placeholders.)
- Pick α.
- Pick starting guesses for $\hat{\beta}_0$, $\hat{\beta}_1$, and $\hat{\beta}_2$.

GRADIENT DESCENT ALGORITHM

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

LOGISTIC REGRESSION MODEL

- Pick starting guesses for $\hat{\beta}_0$, $\hat{\beta}_1$, and $\hat{\beta}_2$.
- Calculate gradients:
  - $\frac{\partial L}{\partial \beta_0} = (y_i - \hat{y}_i)$
  - $\frac{\partial L}{\partial \beta_1} = x_1(y_i - \hat{y}_i)$
  - $\frac{\partial L}{\partial \beta_2} = x_2(y_i - \hat{y}_i)$

GRADIENT DESCENT ALGORITHM

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

LOGISTIC REGRESSION MODEL

- Calculate gradients:
  - $\frac{\partial L}{\partial \beta_0} = (y_i - \hat{y}_i)$
  - $\frac{\partial L}{\partial \beta_1} = x_1(y_i - \hat{y}_i)$
  - $\frac{\partial L}{\partial \beta_2} = x_2(y_i - \hat{y}_i)$

- Update guesses!
  - $\hat{\beta}_{0,1} = \hat{\beta}_{0,0} - \alpha(y_i - \hat{y}_i)$
  - $\hat{\beta}_{1,1} = \hat{\beta}_{1,0} - \alpha x_1(y_i - \hat{y}_i)$
  - $\hat{\beta}_{2,1} = \hat{\beta}_{2,0} - \alpha x_2(y_i - \hat{y}_i)$

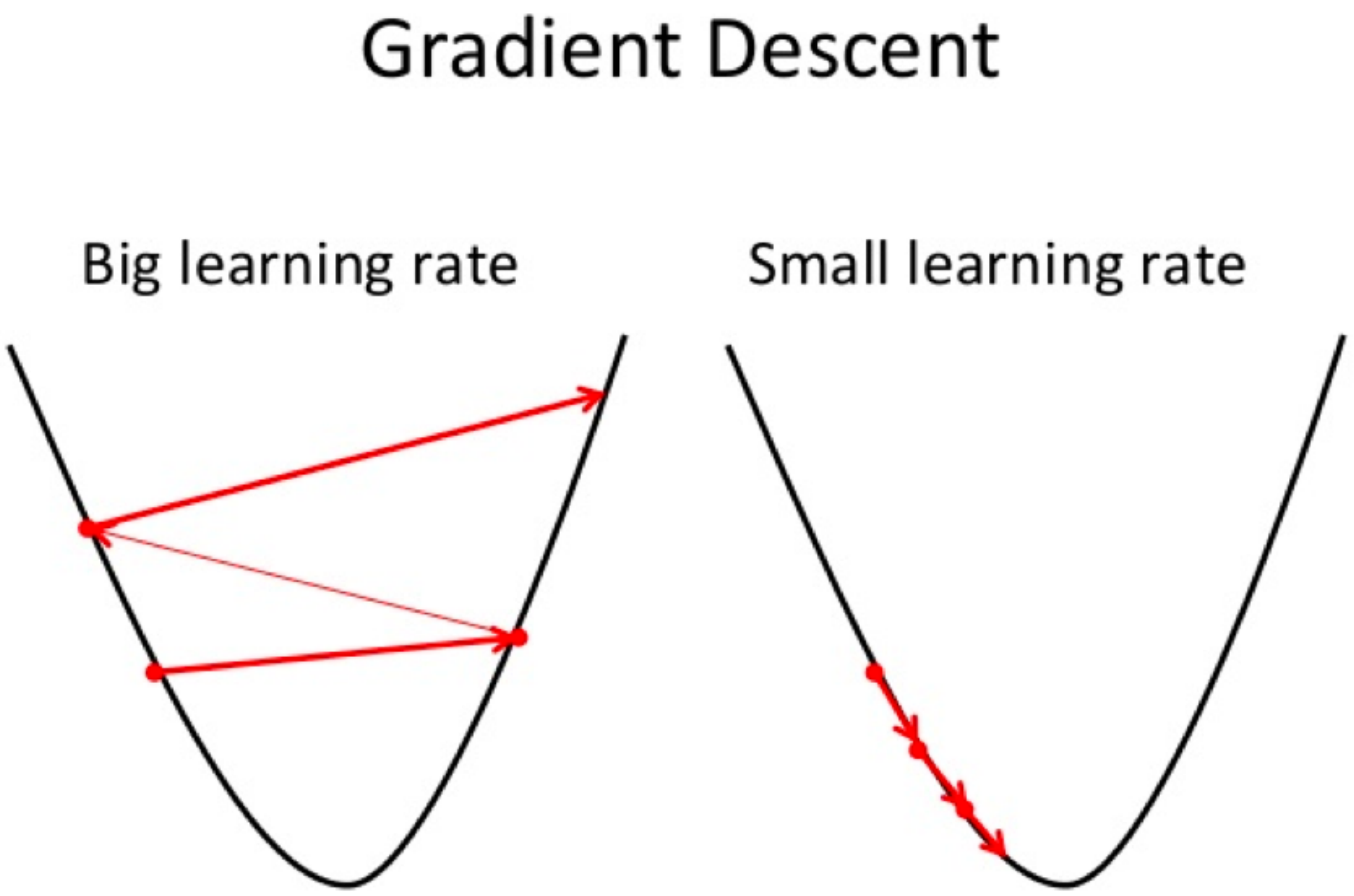
GRADIENT DESCENT ALGORITHM

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

INTUITIVE RECAP

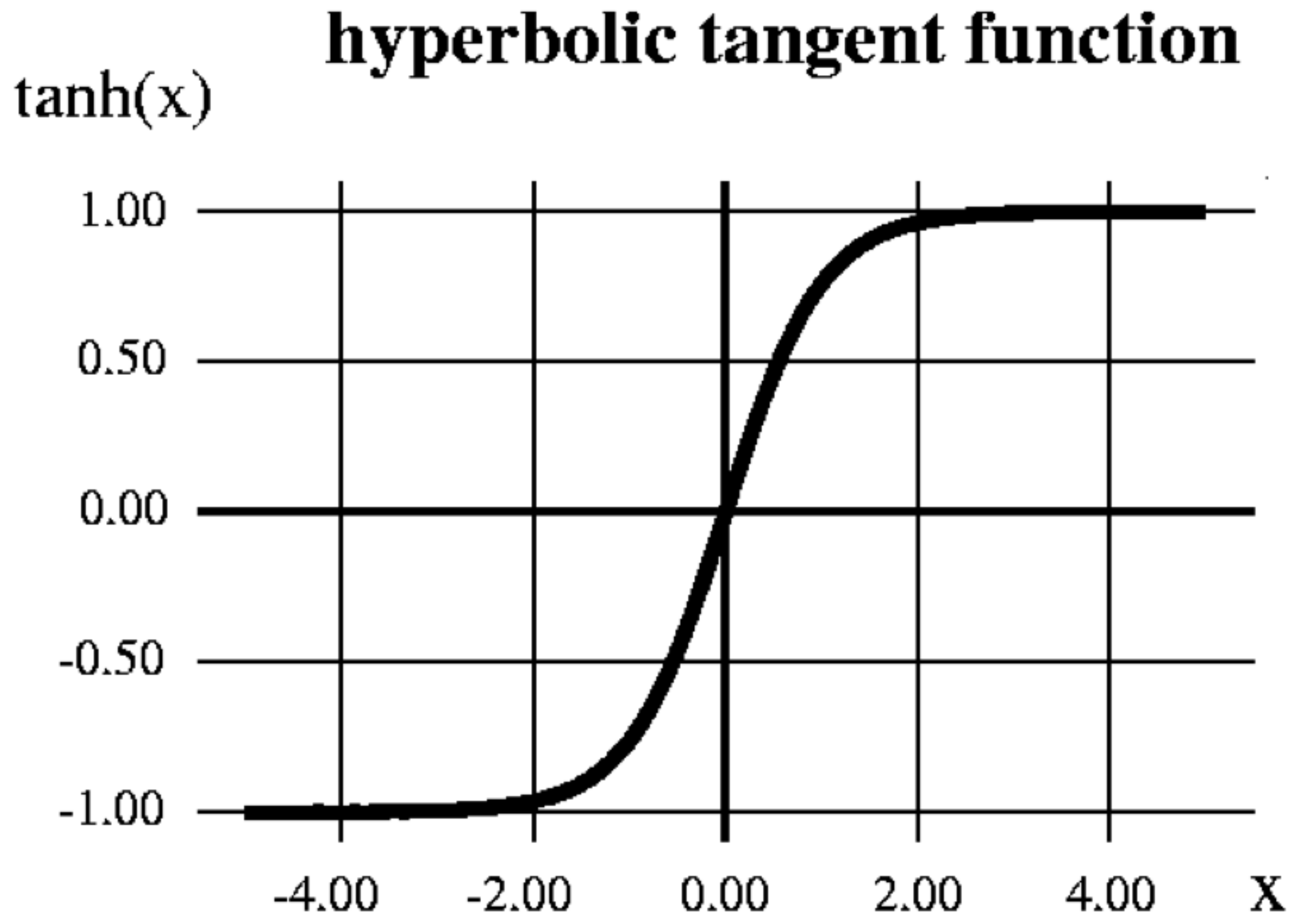
- When fitting a model, we:
    - Identify some loss function to optimize.
    - Pick a first guess.
    - Take a step of fixed size in the "best" direction.
    - Keep going until we've found the minimum of our loss function!

POTENTIAL PITFALLS



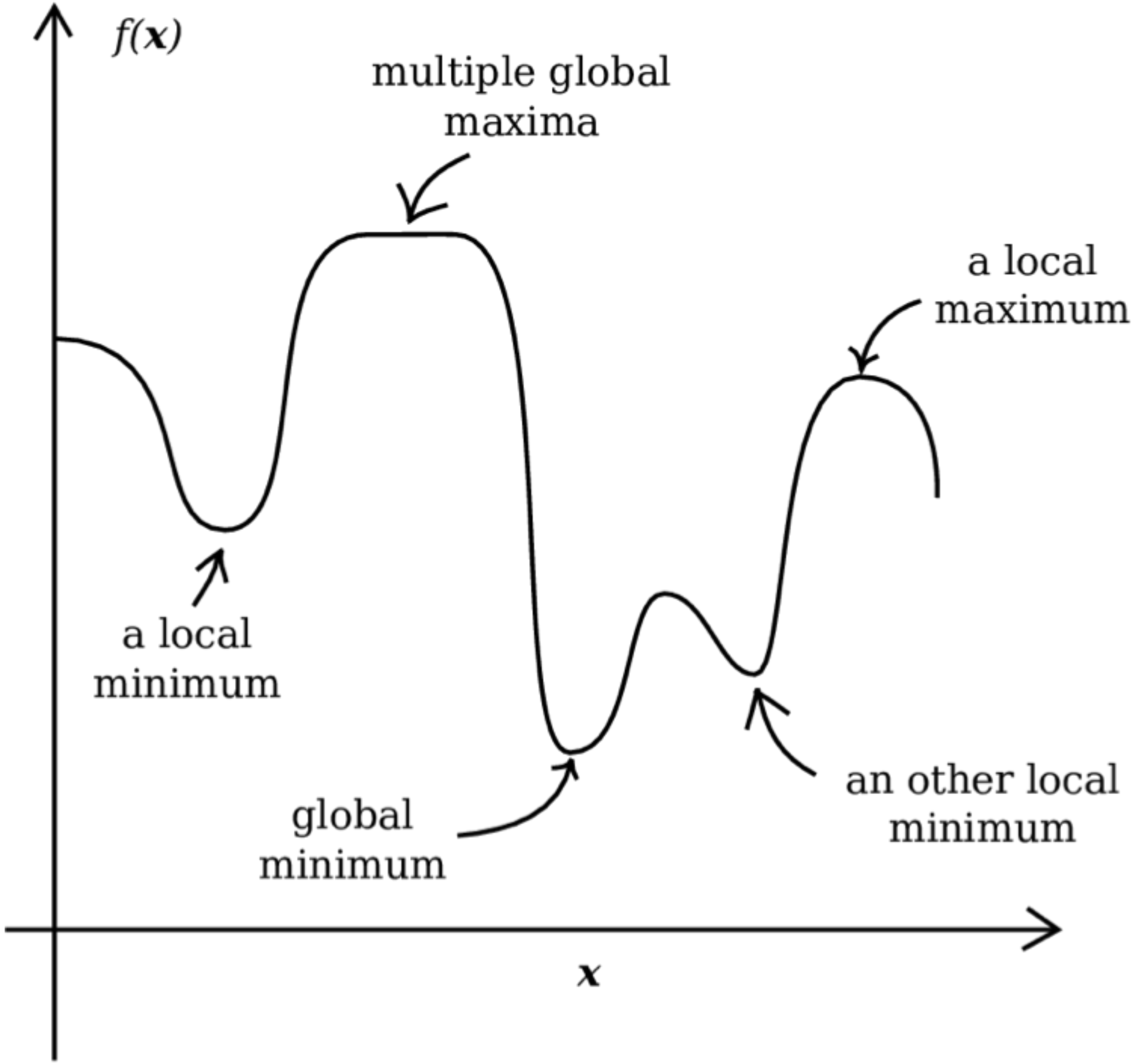
- If our step size is too big, we may never converge!
- If our step size is too small, it may take a very long time for us to converge!

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.

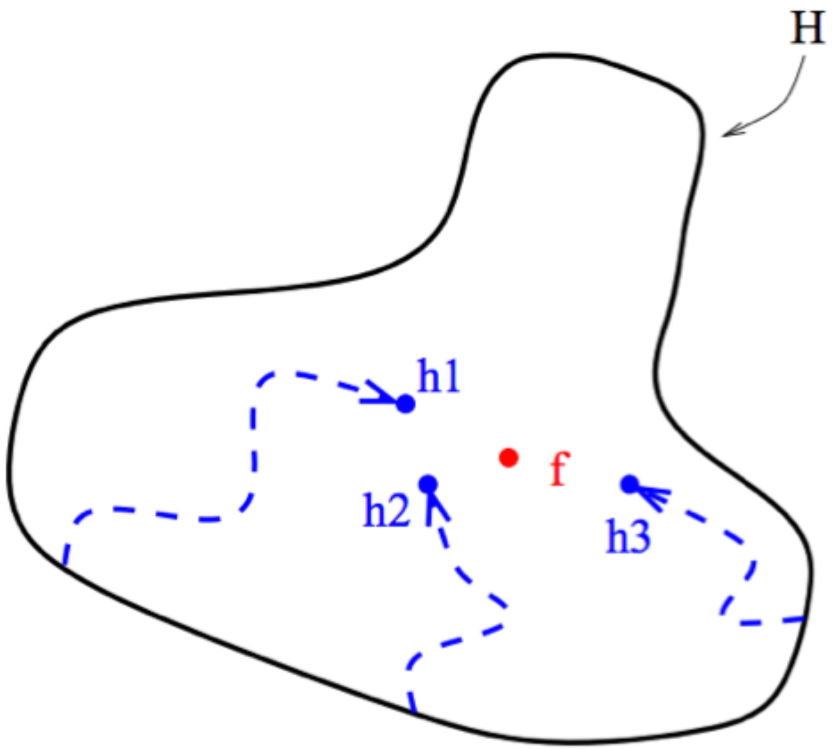
SOLUTION 1: STOCHASTIC GRADIENT DESCENT

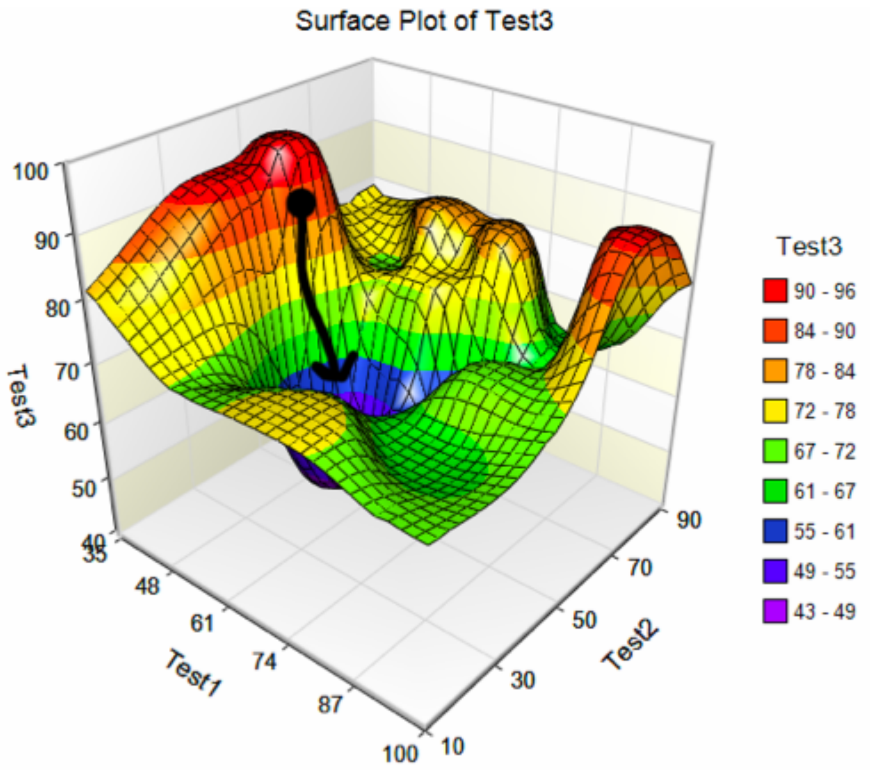
- One way to attempt to protect against some of these pitfalls is to use **stochastic gradient descent**, which means that, at each step, we draw α from some distribution so that we aren't taking a step of fixed size.

- Practically, we could fit a model in this manner multiple times. If we repeatedly get nearly identical results, we can be more confident that we've arrived at the global optimum.

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.





MACHINE LEARNING = GRADIENT DESCENT