



# CS 4375

## Introduction to Machine Learning

Nicholas Ruozzi  
University of Texas at Dallas

# What is ML?

---



*“A computer program is said to learn from experience  $E$  with respect to some task  $T$  and some performance measure  $P$ , if its performance on  $T$ , as measured by  $P$ , improves with experience  $E$ .”*

*- Tom Mitchell*

# Course Info.



- Instructor: Nicholas Ruozzi
  - Office: ECSS 3.409
  - Office hours: M 1pm-2pm, W 10:30am-11:30am
- TA: ?
  - Office hours and location: ?
- [Course website](#)
- Book: none required
- Piazza (online forum): sign-up link on eLearning

# Optional Recitation Section



- New this semester: a recitation section for this course
- This supplemental addition to the course is strictly **optional**
  - Instructor: Jim Amato
  - Where: ECSW 3.250
  - When: Thursdays, 5:30pm-6:30pm

# Prerequisites



- CS3345, Data Structures and Algorithms
- CS3341, Probability and Statistics in Computer Science
- “Mathematical sophistication”
  - Basic probability
  - Linear algebra: eigenvalues/vectors, matrices, vectors, etc.
  - Multivariate calculus: derivatives, gradients, etc.
- I’ll review some concepts as we come to them, but **you should brush up on areas that you aren’t as comfortable**

- Dimensionality reduction
  - PCA
  - Matrix Factorizations
- Learning
  - Supervised, unsupervised, active, reinforcement, ...
  - SVMs & kernel methods
  - Decision trees, k-NN, logistic regression, ...
  - Parameter estimation: Bayesian methods, MAP estimation, maximum likelihood estimation, expectation maximization, ...
  - Clustering: k-means & spectral clustering
- Probabilistic models
  - Bayesian networks
  - Naïve Bayes
- Neural networks
- Evaluation
  - AOC, cross-validation, precision/recall
- Statistical methods
  - Boosting, bagging, bootstrapping
  - Sampling

- 5-6 problem sets (50%)
  - See collaboration policy on the web
  - Mix of theory and programming (in MATLAB or Python)
  - Available and turned in on eLearning
  - Approximately one assignment every two weeks
- Midterm Exam (20%)
- Final Exam (30%)
- Attendance policy?

*-subject to change-*



# Basic Machine Learning Paradigm



- Collect data
- Build a model using “training” data
- Use model to make predictions

- **Input:**  $(x^{(1)}, y^{(1)}), \dots, (x^{(M)}, y^{(M)})$ 
  - $x^{(m)}$  is the  $m^{th}$  data item and  $y^{(m)}$  is the  $m^{th}$  **label**
- **Goal:** find a function  $f$  such that  $f(x^{(m)})$  is a “good approximation” to  $y^{(m)}$ 
  - Can use it to predict  $y$  values for previously unseen  $x$  values

# Examples of Supervised Learning

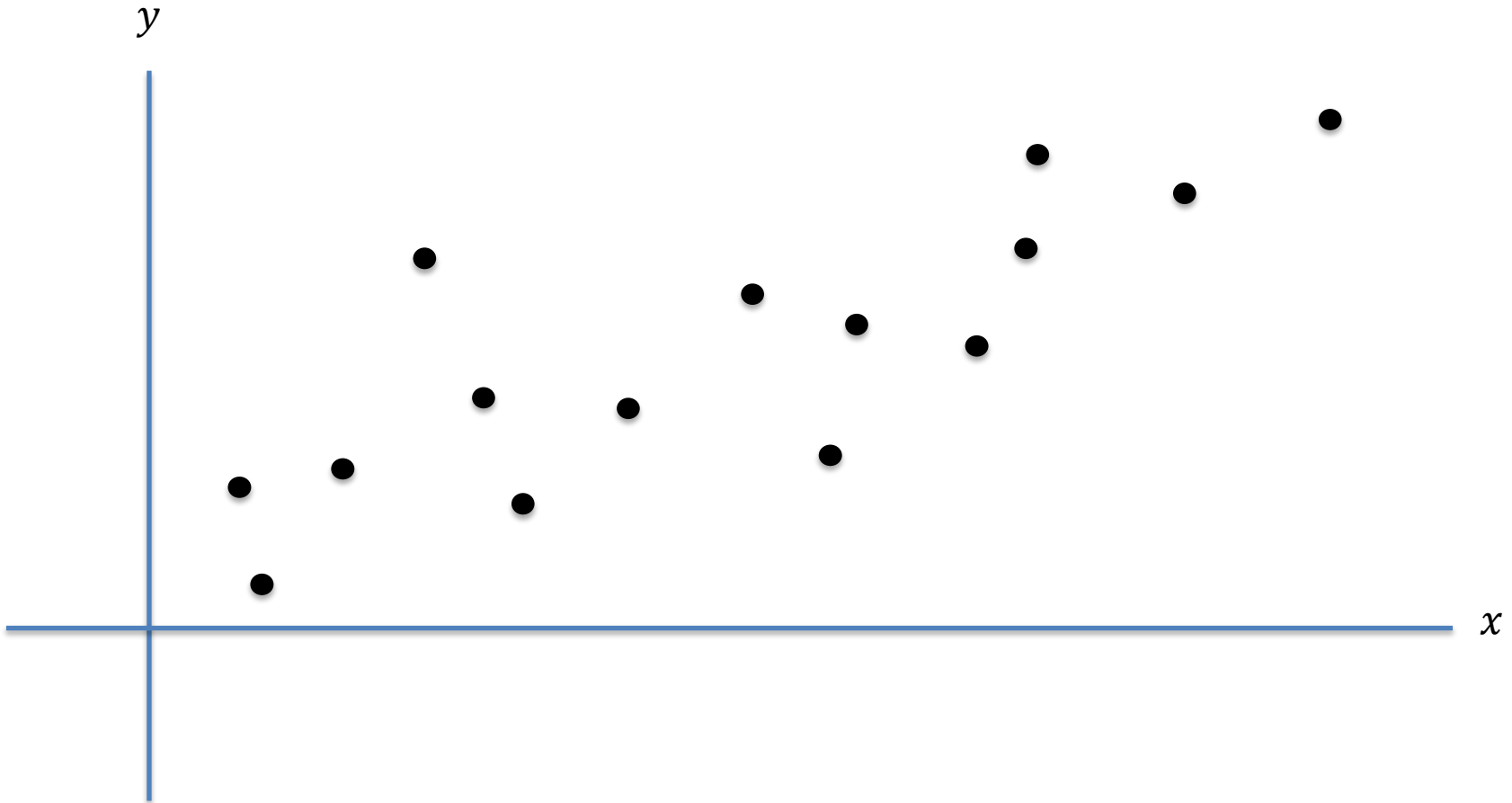


- Spam email detection
- Handwritten digit recognition
- Stock market prediction
- More?

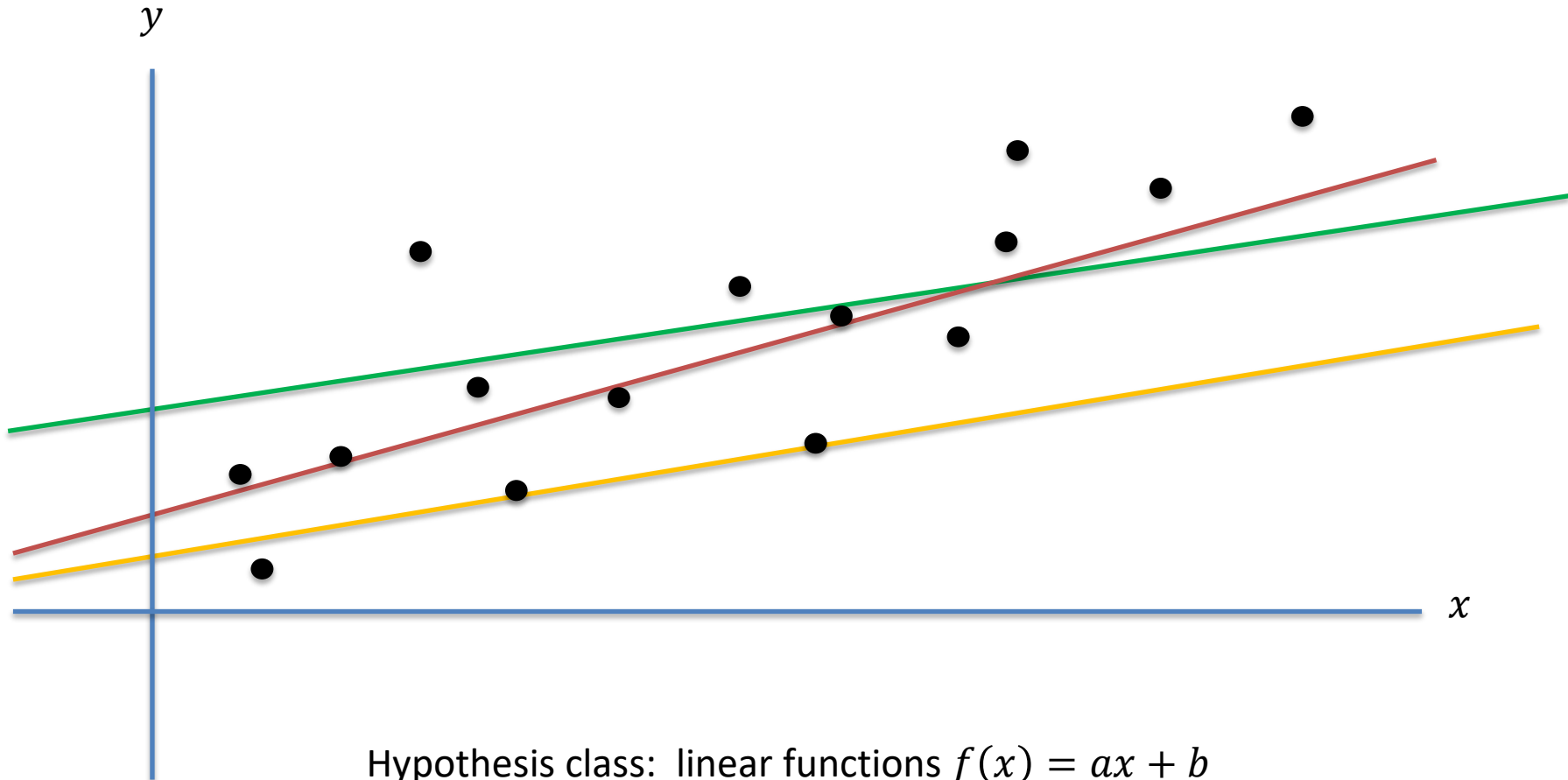
- **Hypothesis space**: set of allowable functions  $f: X \rightarrow Y$
- Goal: find the “best” element of the hypothesis space
  - How do we measure the quality of  $f$ ?

- Linear regression in  $\mathbb{R}$ 
  - Input: pairs of points  $(x^{(1)}, y^{(1)}), \dots, (x^{(M)}, y^{(M)})$  with  $x^{(m)} \in \mathbb{R}$  and  $y^{(m)} \in \mathbb{R}$
  - Hypothesis space: set of linear functions  $f(x) = ax + b$  with  $a, b \in \mathbb{R}$
  - Error metric: squared difference between the predicted value and the actual value

# Regression



# Regression



How do we compute the error of a specific hypothesis?

# Linear Regression



- For any data point,  $x$ , the learning algorithm predicts  $f(x)$
- Can assess the fit using a squared **loss function**

$$L(f) = \frac{1}{M} \sum_m (f(x^{(m)}) - y^{(m)})^2$$

- Want to minimize the average loss on the **training data**
- The optimal linear hypothesis is then given by

$$\min_{a,b} \frac{1}{M} \sum_m (ax^{(m)} + b - y^{(m)})^2$$



# Linear Regression



$$\min_{a,b} \frac{1}{M} \sum_m (ax^{(m)} + b - y^{(m)})^2$$

- How do we find the optimal  $a$  and  $b$ ?

$$\min_{a,b} \frac{1}{M} \sum_m (ax^{(m)} + b - y^{(m)})^2$$

- How do we find the optimal  $a$  and  $b$ ?
  - Solution 1: take derivatives and solve  
(there is a closed form solution!)
  - Solution 2: use gradient descent

$$\min_{a,b} \frac{1}{M} \sum_m (ax^{(m)} + b - y^{(m)})^2$$

- How do we find the optimal  $a$  and  $b$ ?
  - Solution 1: take derivatives and solve  
(there is a closed form solution!)
  - Solution 2: use gradient descent
    - This approach is much more likely to be useful for general loss functions

Iterative method to minimize a **(convex) differentiable** function  $f$

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

$$\lambda f(x) + (1 - \lambda)f(y) \geq f(\lambda x + (1 - \lambda)y)$$

for all  $\lambda \in [0,1]$  and all  $x, y \in \mathbb{R}^n$

Iterative method to minimize a (convex) differentiable function  $f$

- Pick an initial point  $x_0$
- Iterate until convergence

$$x_{t+1} = x_t - \gamma_t \nabla f(x_t)$$

where  $\gamma_t$  is the  $t^{\text{th}}$  step size (sometimes called learning rate)

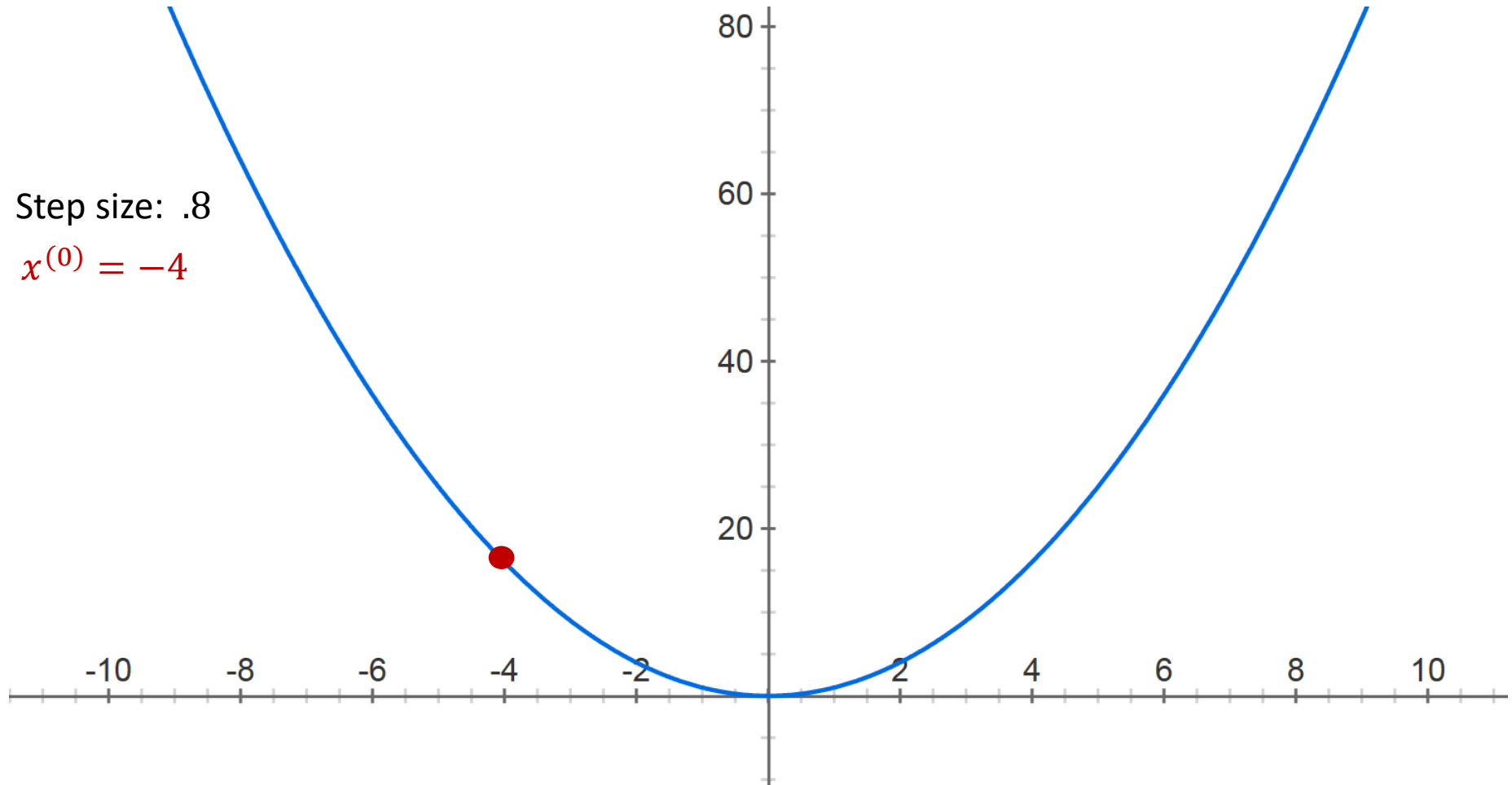
# Gradient Descent



$$f(x) = x^2$$

Step size: .8

$$x^{(0)} = -4$$



# Gradient Descent

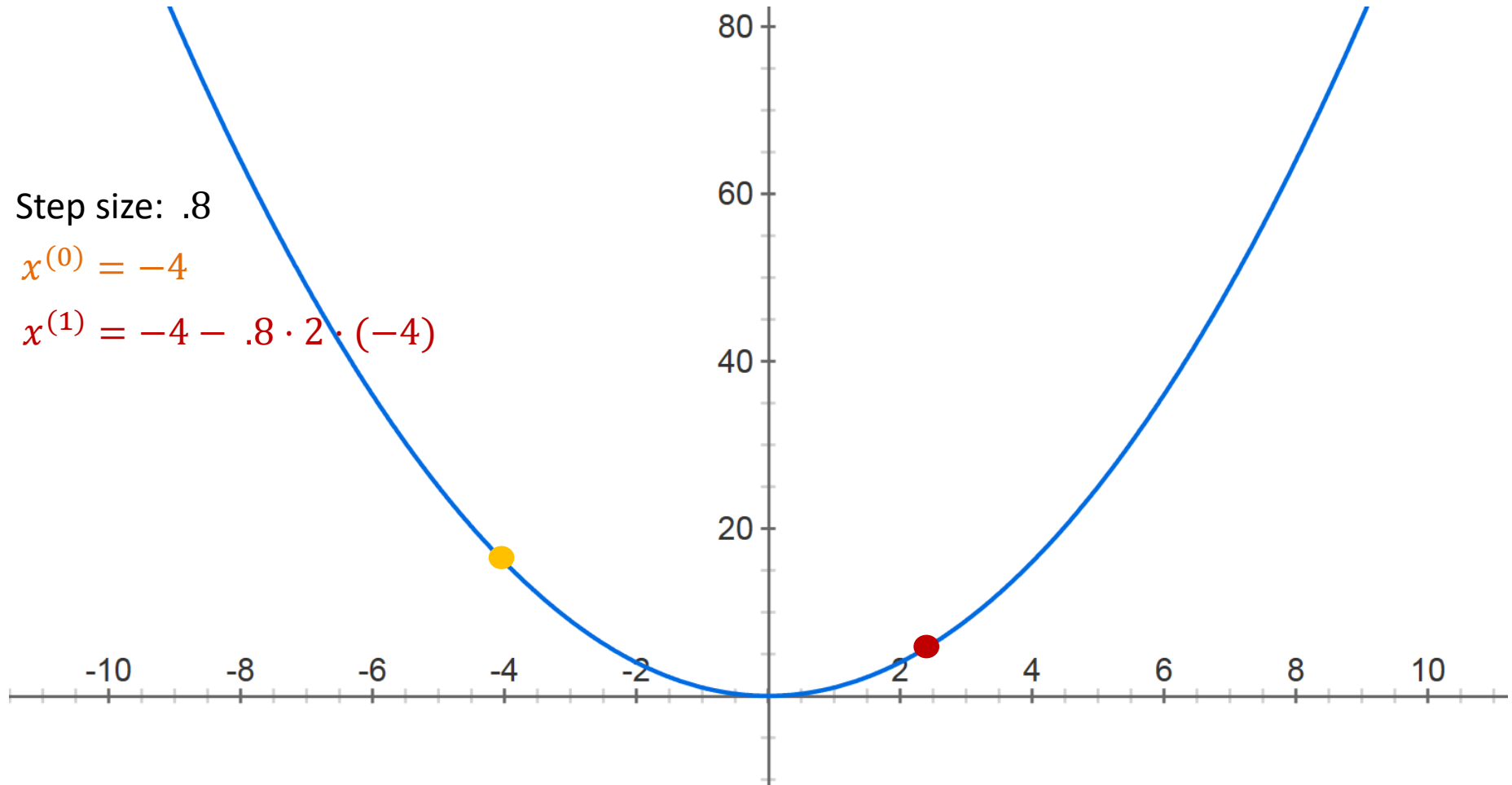


$$f(x) = x^2$$

Step size: .8

$$x^{(0)} = -4$$

$$x^{(1)} = -4 - .8 \cdot 2 \cdot (-4)$$



# Gradient Descent

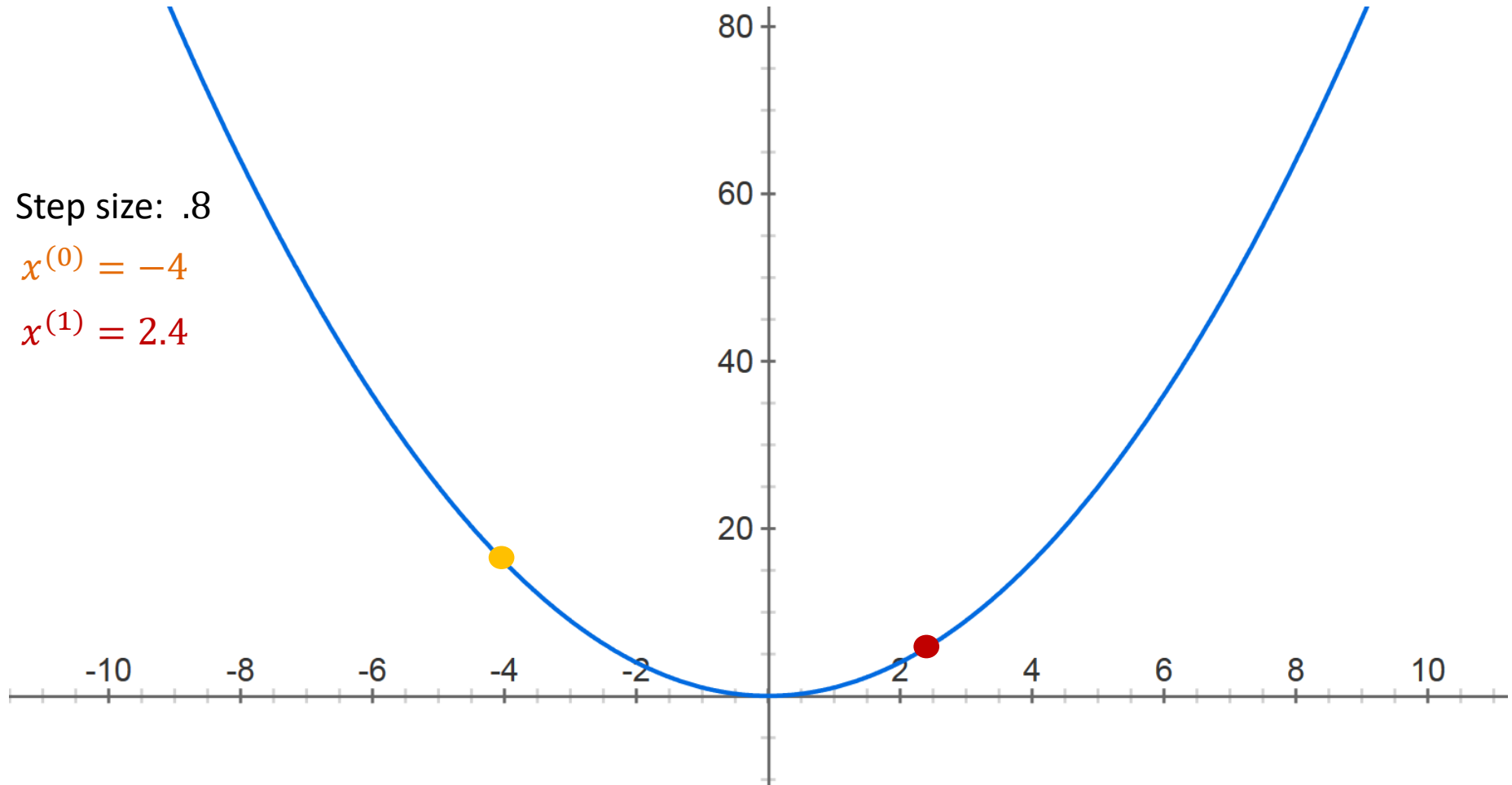


$$f(x) = x^2$$

Step size: .8

$$x^{(0)} = -4$$

$$x^{(1)} = 2.4$$





# Gradient Descent



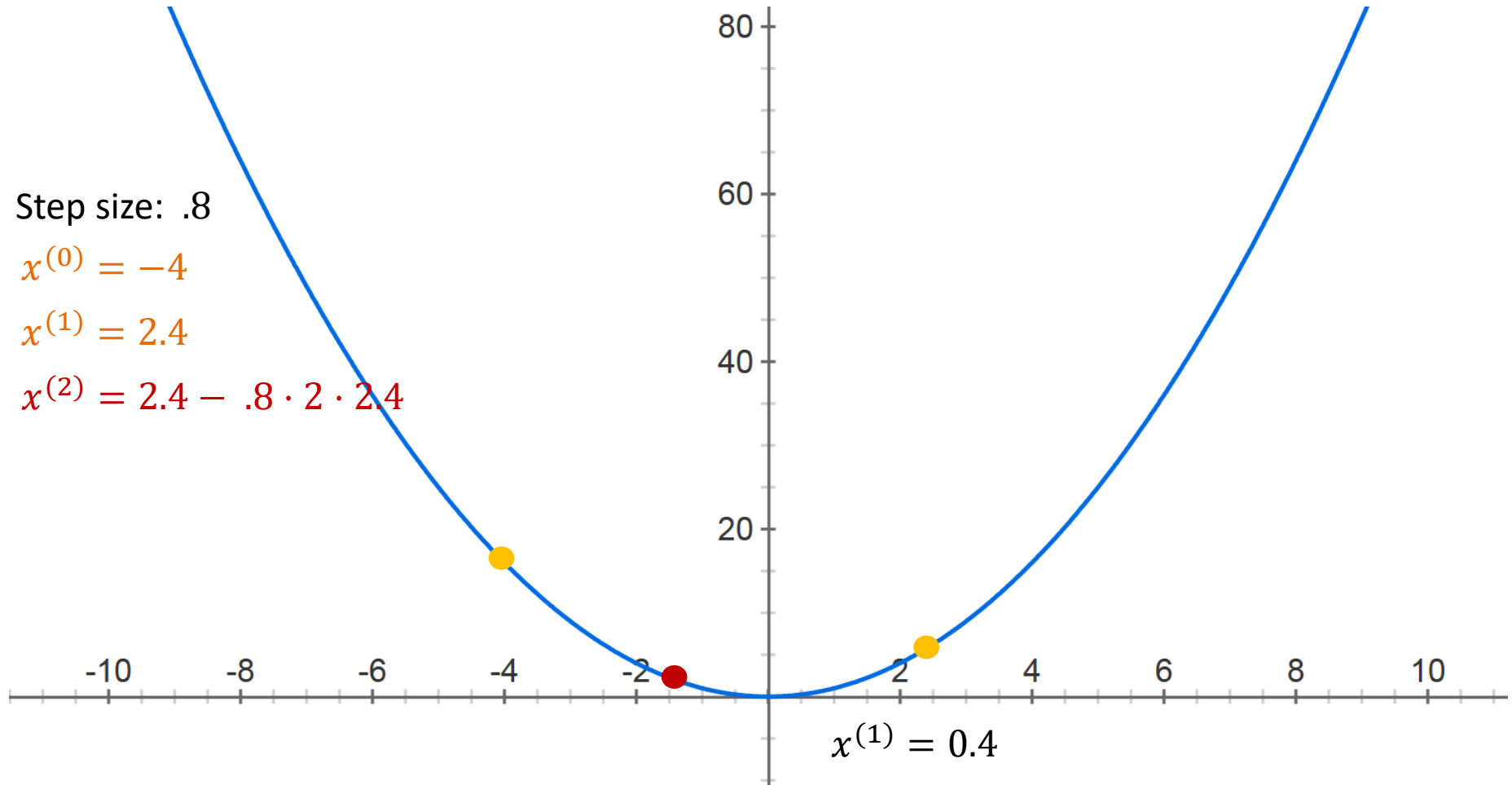
$$f(x) = x^2$$

Step size: .8

$$x^{(0)} = -4$$

$$x^{(1)} = 2.4$$

$$x^{(2)} = 2.4 - .8 \cdot 2 \cdot 2.4$$



# Gradient Descent



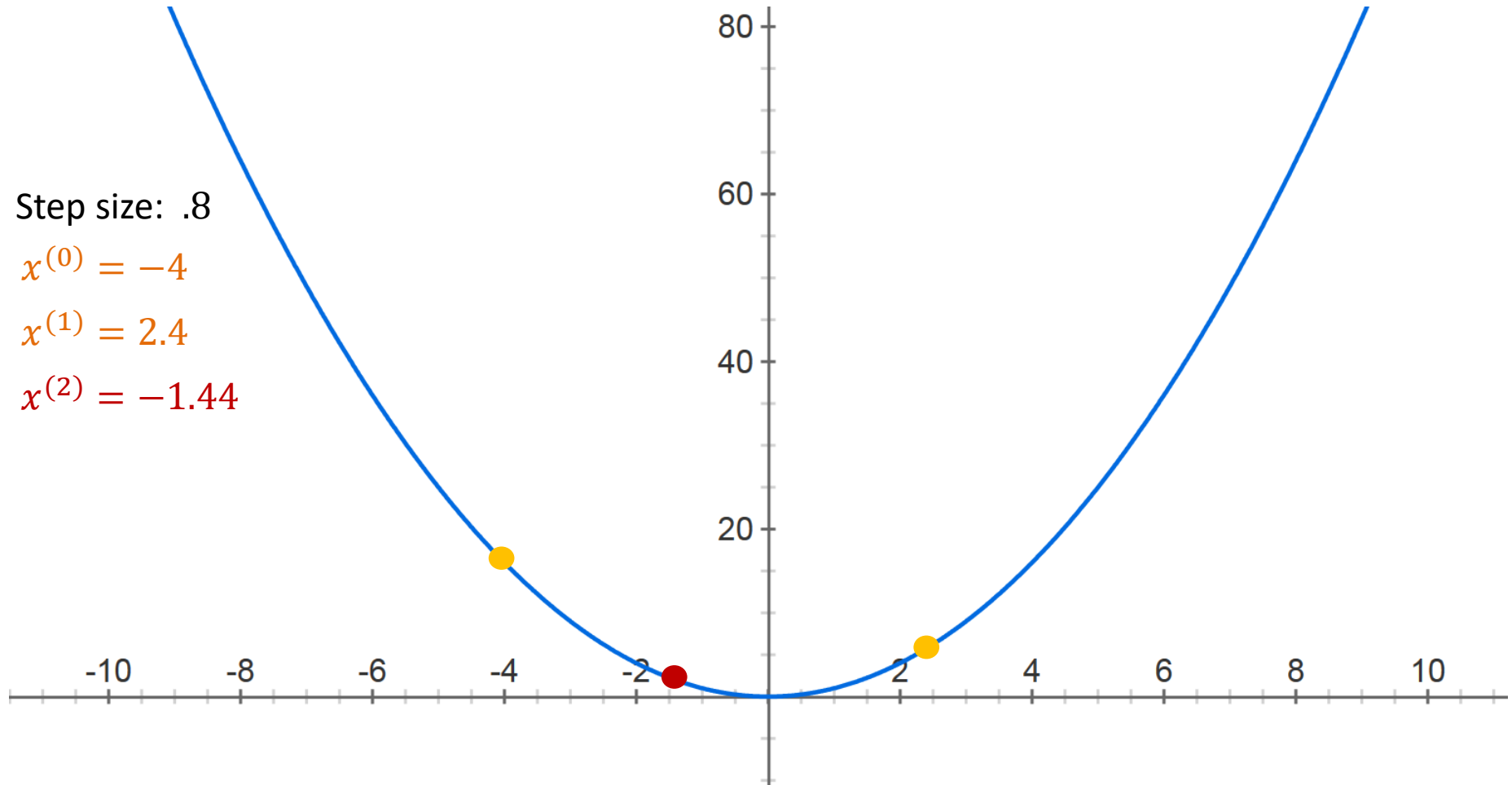
$$f(x) = x^2$$

Step size: .8

$$x^{(0)} = -4$$

$$x^{(1)} = 2.4$$

$$x^{(2)} = -1.44$$



# Gradient Descent



$$f(x) = x^2$$

Step size: .8

$$x^{(0)} = -4$$

$$x^{(1)} = 2.4$$

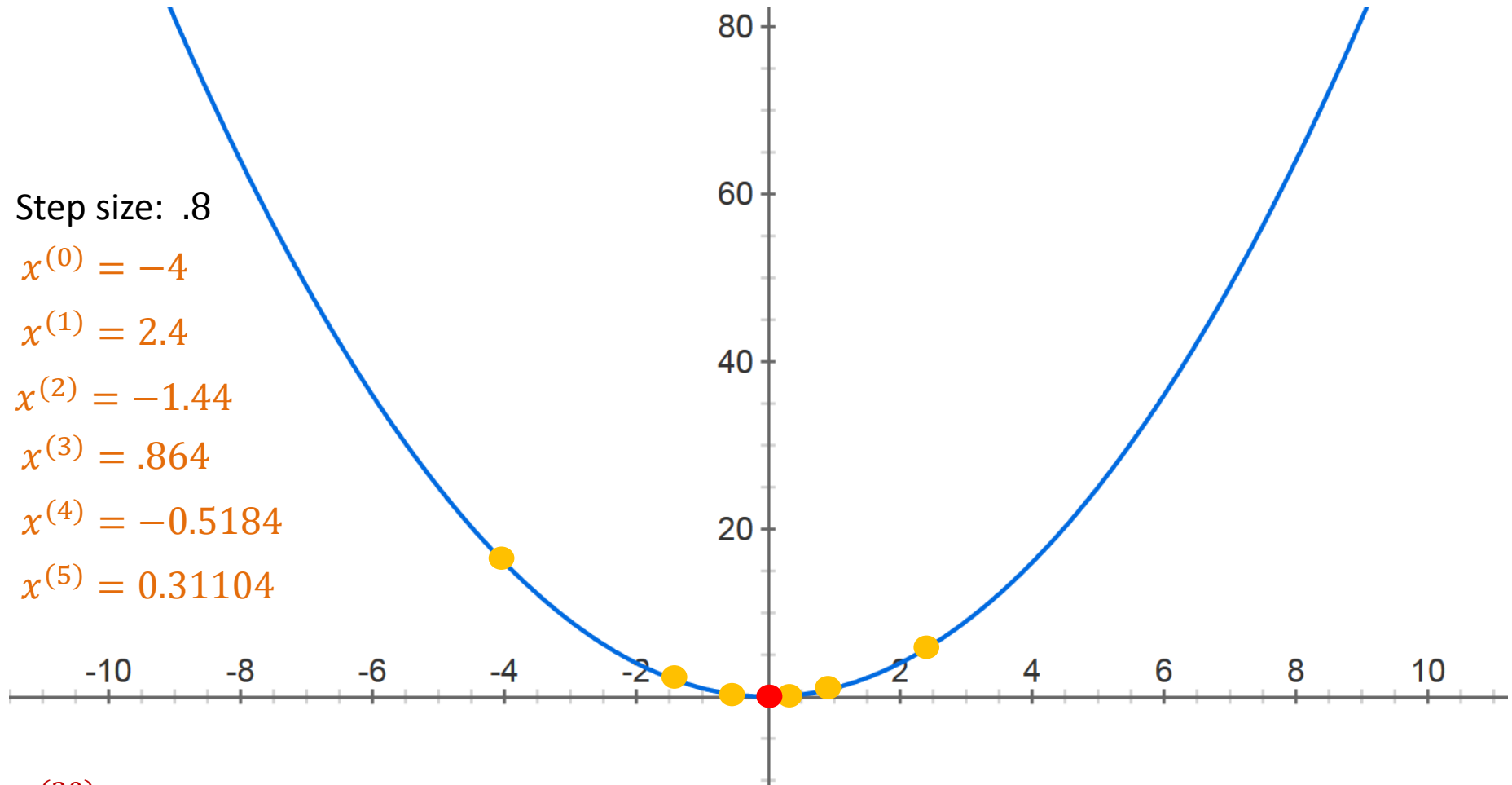
$$x^{(2)} = -1.44$$

$$x^{(3)} = .864$$

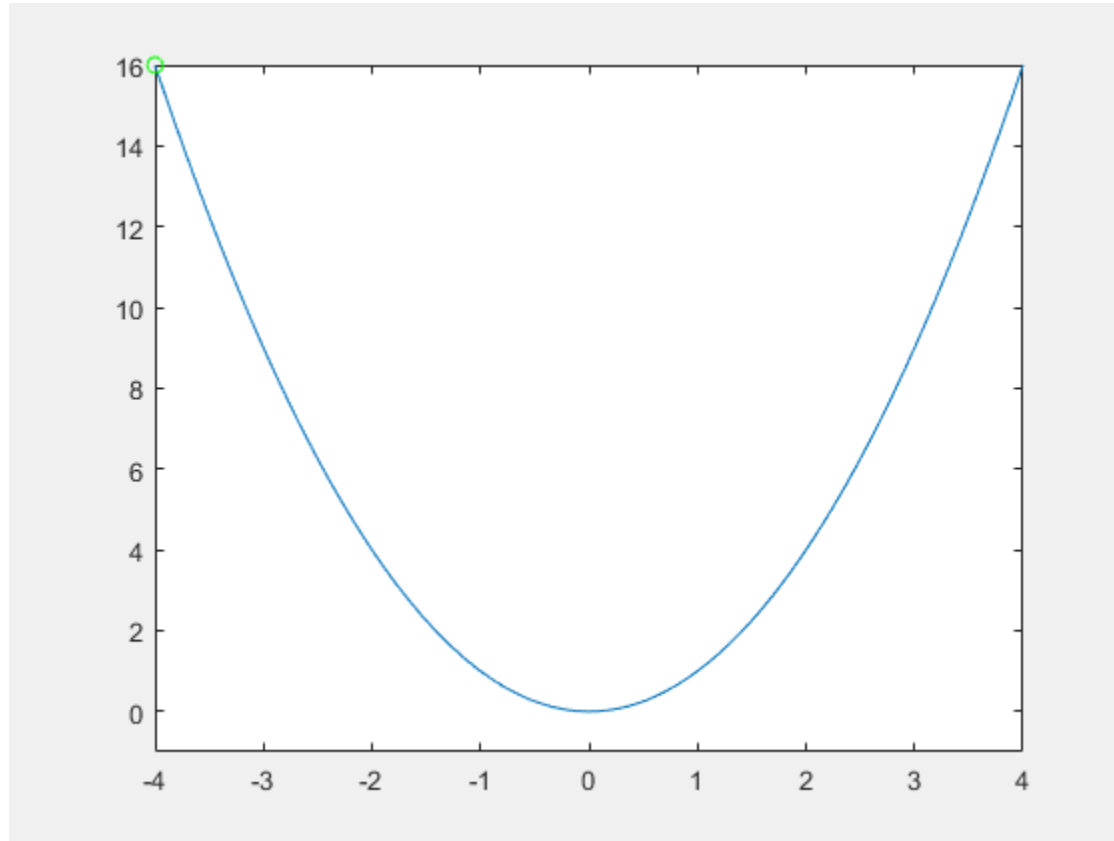
$$x^{(4)} = -0.5184$$

$$x^{(5)} = 0.31104$$

$$x^{(30)} = -8.84296e - 07$$

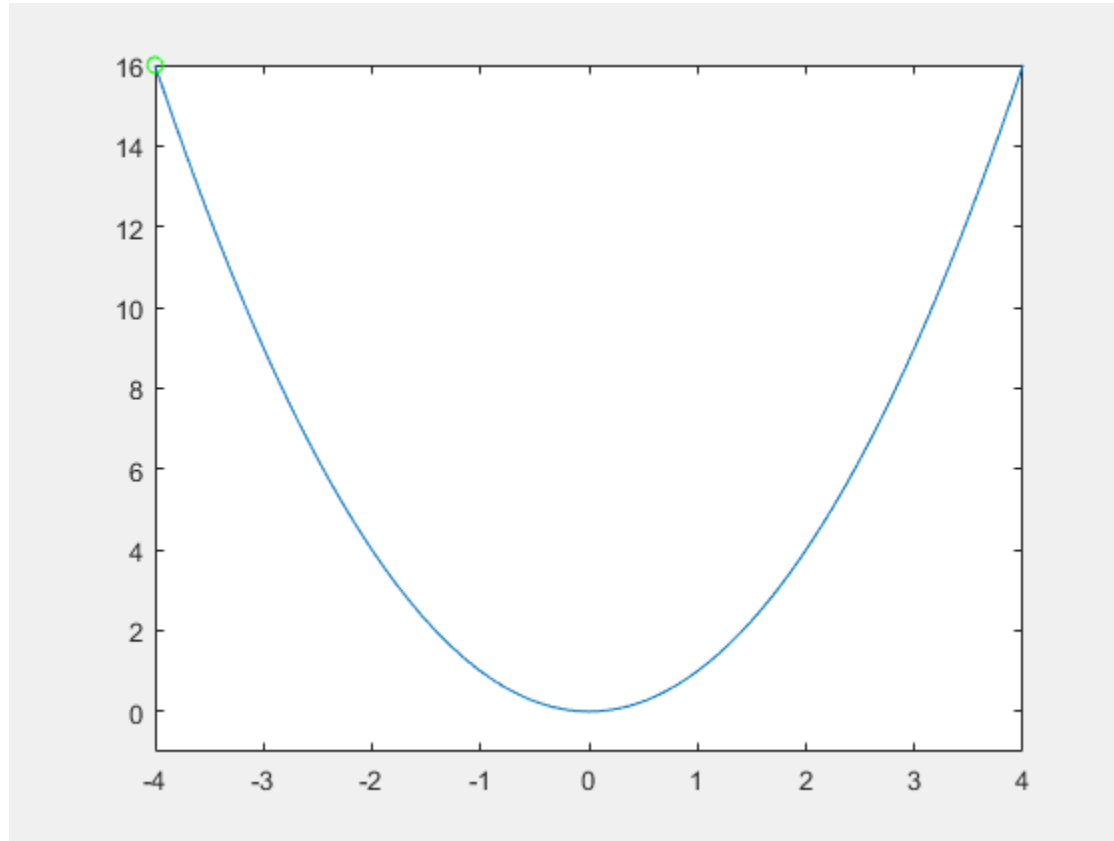


# Gradient Descent



Step size: .9

# Gradient Descent



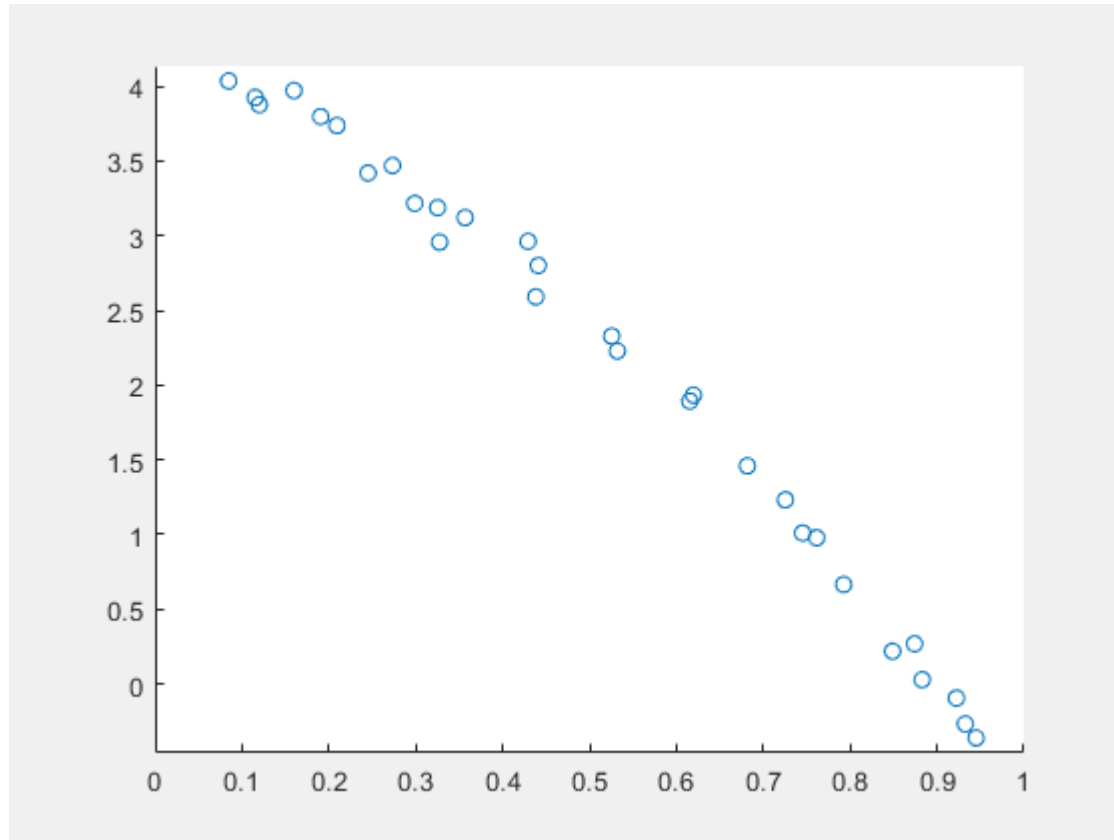
Step size: .2

$$\min_{a,b} \frac{1}{M} \sum_m (ax^{(m)} + b - y^{(m)})^2$$

- What is the gradient of this function?
- What does a gradient descent iteration look like for this simple regression problem?

(on board)

# Gradient Descent



- In higher dimensions, the linear regression problem is essentially the same with  $x^{(m)} \in \mathbb{R}^n$

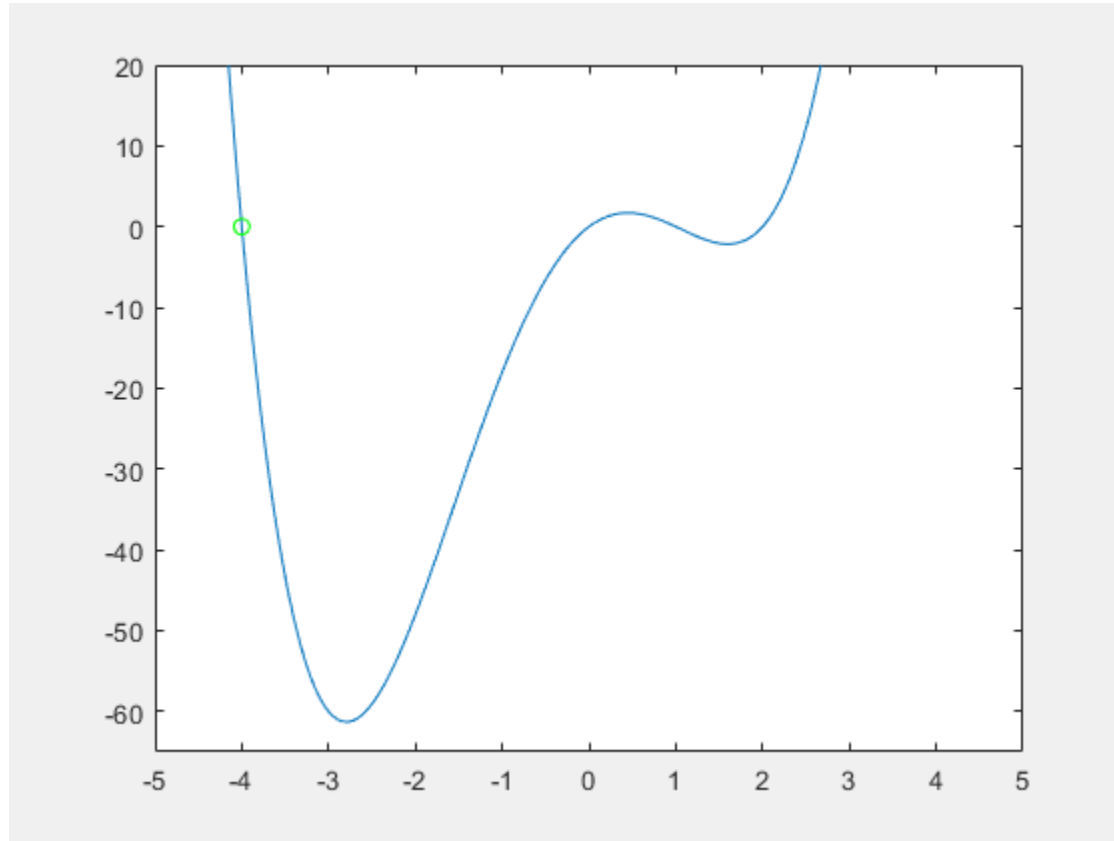
$$\min_{a \in \mathbb{R}^n, b} \frac{1}{M} \sum_m (a^T x^{(m)} + b - y^{(m)})^2$$

- Can still use gradient descent to minimize this
  - Not much more difficult than the  $n = 1$  case



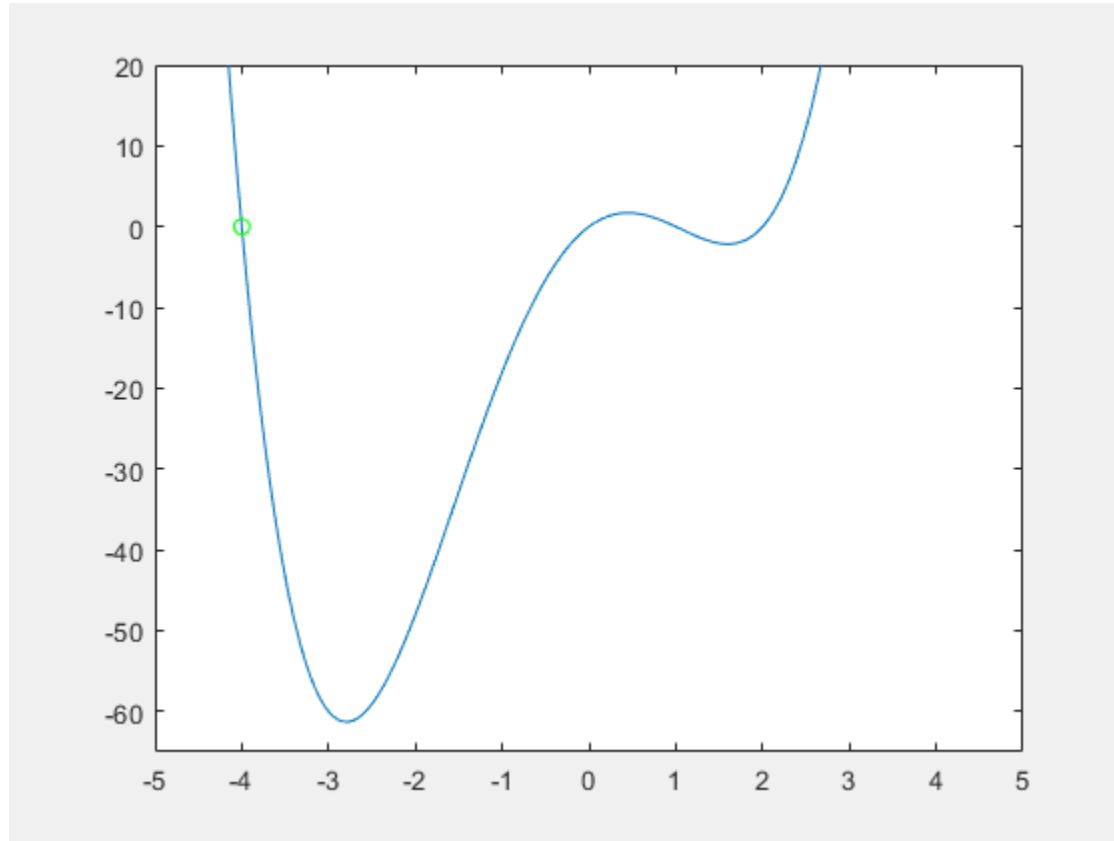
- Gradient descent converges under certain technical conditions on the function  $f$  and the step size  $\gamma_t$ 
  - If  $f$  is convex, then any fixed point of gradient descent must correspond to a global minimum of  $f$
  - In general, for a nonconvex function, may only converge to a local optimum

# Gradient Descent



Step size matters!

# Gradient Descent



Step size matters!

- What if we enlarge the hypothesis class?
  - Quadratic functions:  $ax^2 + bx + c$
  - $k$ -degree polynomials:  $a_k x^k + a_{k-1} x^{k-1} + \dots + a_1 x + a_0$

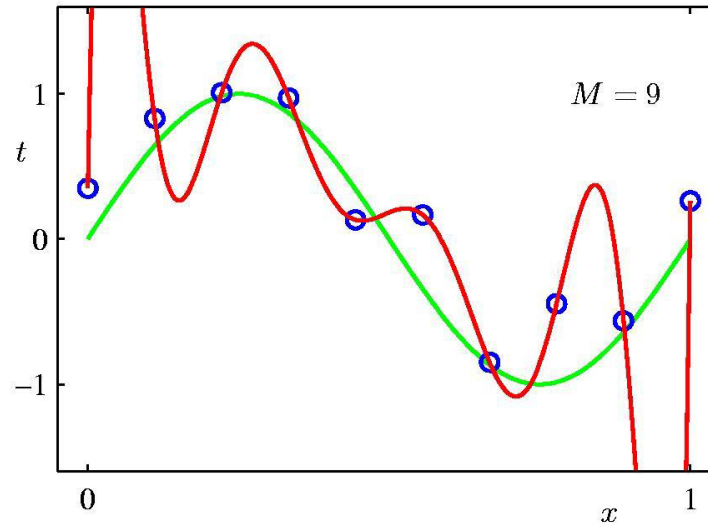
$$\min_{a_k, \dots, a_0} \frac{1}{M} \sum_m \left( a_k (x^{(m)})^k + \dots + a_1 x^{(m)} + a_0 - y^{(m)} \right)^2$$

- What if we enlarge the hypothesis class?
  - Quadratic functions:  $ax^2 + bx + c$
  - $k$ -degree polynomials:  $a_k x^k + a_{k-1} x^{k-1} + \dots + a_1 x + a_0$
- Can we always learn “better” with a larger hypothesis class?

# Regression



- What if we enlarge the hypothesis class?
  - Quadratic functions:  $ax^2 + bx + c$
  - $k$ -degree polynomials:  $a_k x^k + a_{k-1} x^{k-1} + \dots + a_1 x + a_0$
- Can we always learn “better” with a larger hypothesis class?



- Larger hypothesis space always decreases the cost function, but this does **NOT** necessarily mean better predictive performance
  - This phenomenon is known as **overfitting**
  - Ideally, we would select the **simplest** hypothesis consistent with the observed data
- In practice, we cannot simply evaluate our learned hypothesis on the training data, we want it to perform well on unseen data (otherwise, we can just memorize the training data!)
  - Report the loss on some held out **test data** (i.e., data not used as part of the training process)