



# CS 6375

## Linear Regression

Rishabh Iyer

University of Texas at Dallas

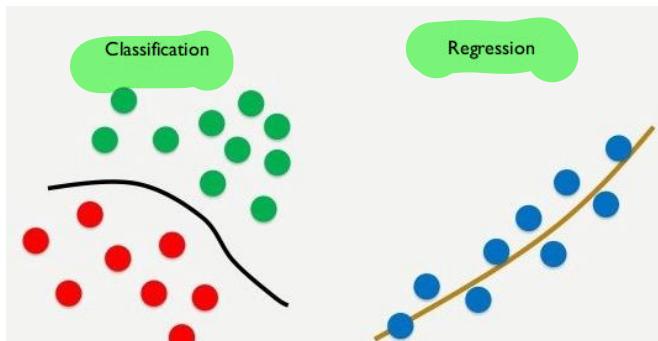
---

# Part I: Recap of Supervised Learning and Linear Regression Setup

# Recap: Supervised Learning



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

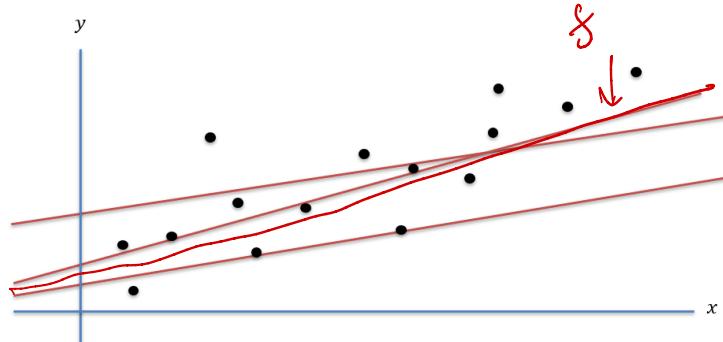


# Recap: Classification vs Regression

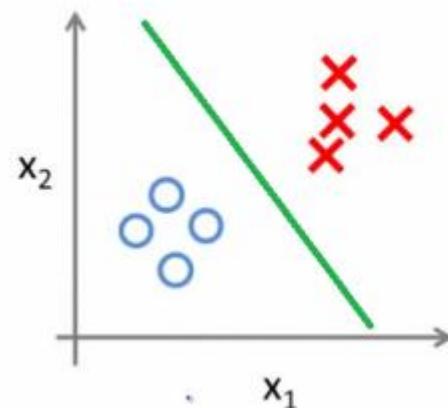


## Classification vs Regression

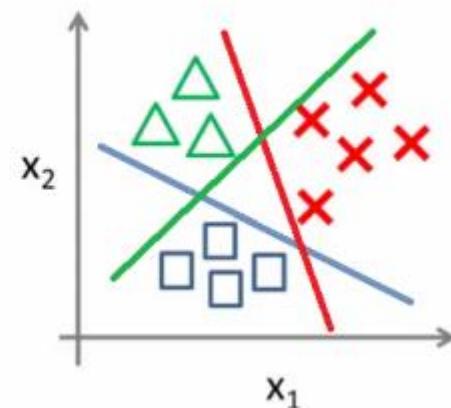
- Input: pairs of points  $(x^{(1)}, y^{(1)}), \dots, (x^{(M)}, y^{(M)})$  with  $x^{(m)} \in \mathbb{R}^n$  [Feature vector]
- Regression case:  $y^{(m)} \in \mathbb{R}$
- Classification case:  $y^{(m)} \in [0, k - 1]$  [k-class classification]
- If  $k = 2$ , we get Binary classification



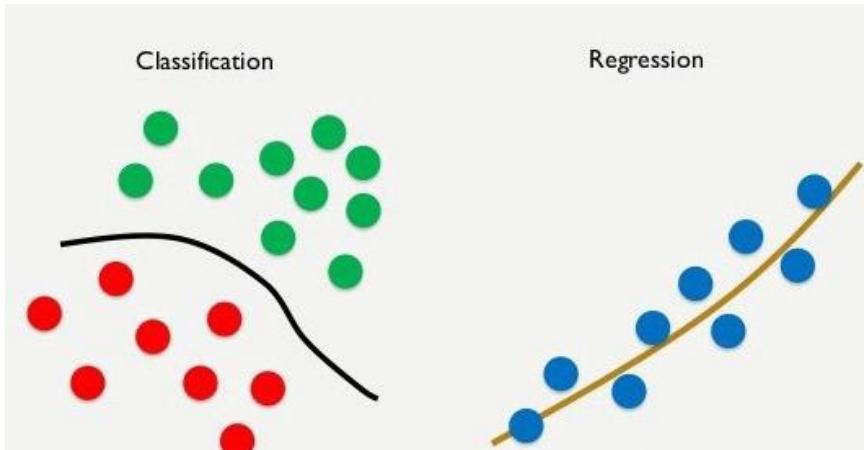
Binary classification:



Multi-class classification:



# Recap: Examples of Supervised Learning



## Classification

- Spam email detection ( $k=2$ )
- Handwritten digit recognition
- Medical Diagnosis ( $k=10$ )
- Fraud Detection
- Face Recognition (*Multiclass*)

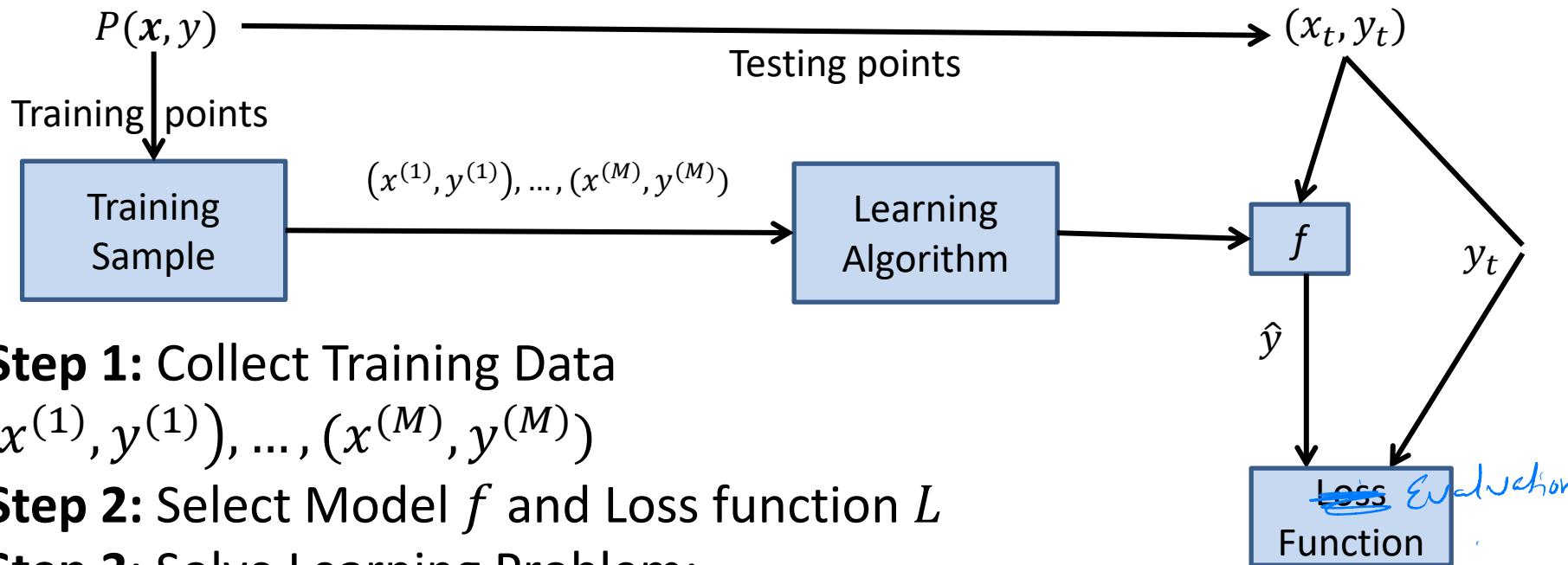
## Regression

- Housing Price Prediction
- Stock Market Prediction
- Weather Prediction
- Market Analysis and Business Trends

# Recap: Hypothesis Space

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

# Recap: Supervised Learning Workflow



- **Step 1:** Collect Training Data  $(x^{(1)}, y^{(1)}), \dots, (x^{(M)}, y^{(M)})$
- **Step 2:** Select Model  $f$  and Loss function  $L$
- **Step 3:** Solve Learning Problem:

$$\min_f \sum_{i=1:m} L(f(x^i), y^i)$$

- **Step 4:** Obtain Predictions  $\hat{y}_t = f(x_t)$  on all Test Data
- **Step 5:** Evaluation -- Measure the error  $Err(\hat{y}_t, y_t)$

~~Loss Evaluation Function~~  
~~Err~~  
~~Err( $\hat{y}_t, y_t$ )~~

# Linear Regression

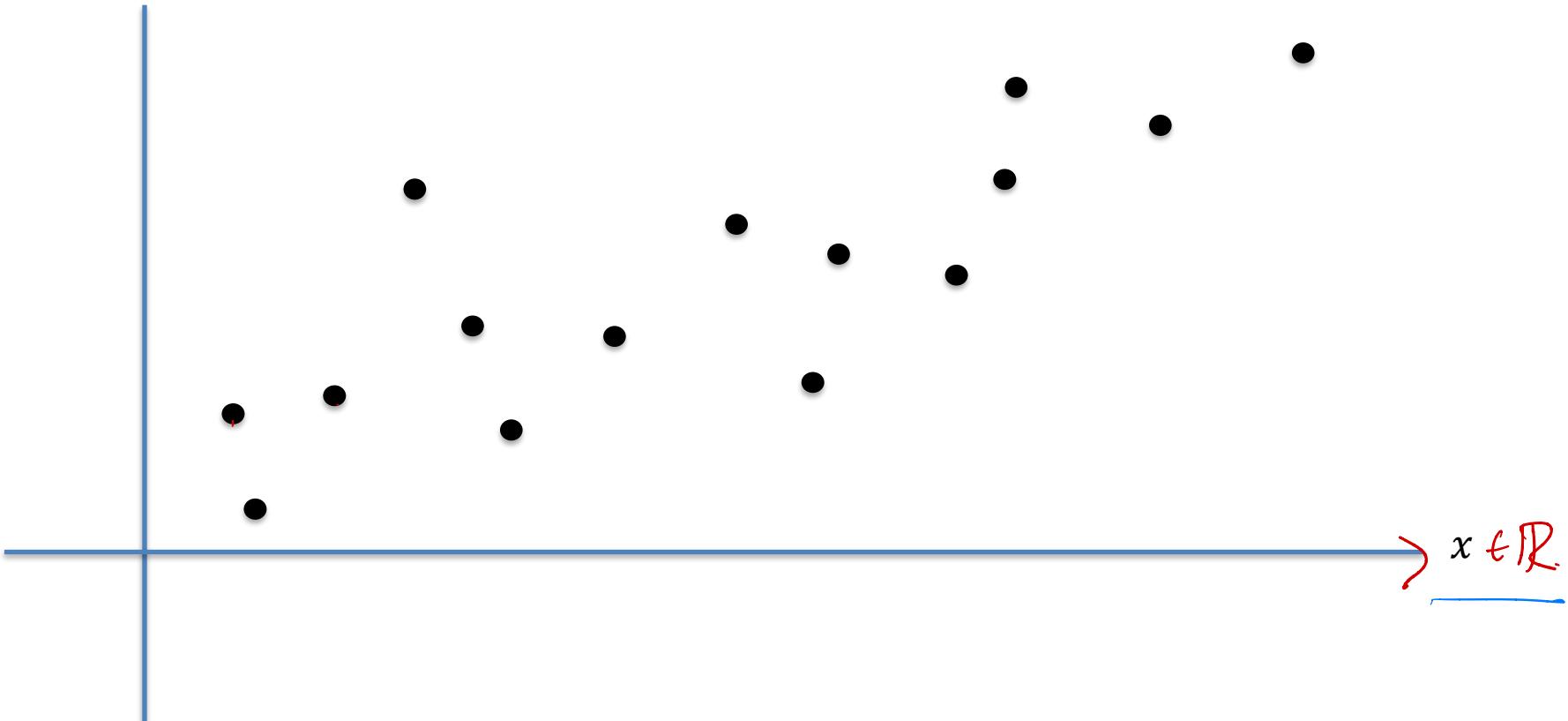
- Simple linear regression
  - Input: pairs of points  $(x^{(1)}, y^{(1)}), \dots, (x^{(M)}, y^{(M)})$  with  $x^{(m)} \in \mathbb{R}^d$  and  $y^{(m)} \in \mathbb{R}$  (Regression)
  - Hypothesis space: set of linear functions  $f(x) = a^T x + b$  with  $a \in \mathbb{R}^n, b \in \mathbb{R}$
  - In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

$$L(y, \hat{y}) = \text{err}(y, \hat{y}) = [y - \hat{y}]^2$$

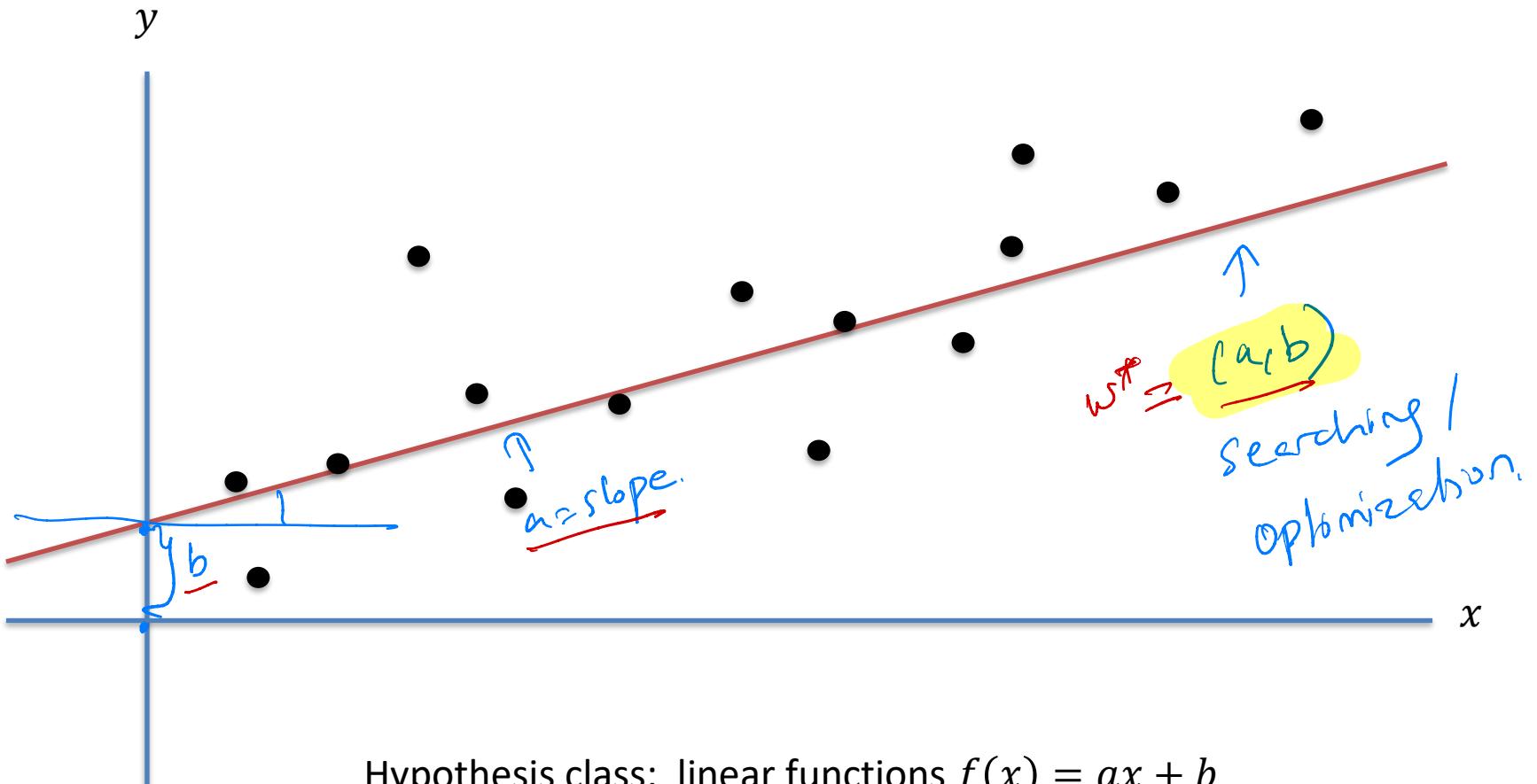
# Regression



$y$  (Label)



# Regression



How do we compute the error of a specific hypothesis?

# Linear Regression

- For any data point,  $x$ , the learning algorithm predicts  $f(x)$
- In typical regression applications, measure the fit using a squared loss function

$$L(a, b) = \frac{1}{M} \sum_{m=1}^M (f(x^{(m)}) - y^{(m)})^2 = \frac{1}{M} \sum_{m=1}^M (a^T x^{(m)} + b - y^{(m)})^2$$

$f(x) = a \cdot x + b$

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

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

$a \in \mathbb{R}^n, x^{(m)} \in \mathbb{R}^n, b \in \mathbb{R}$

Detour

①  $x^{(m)}$   $\rightarrow$

$m^{\text{th}}$  training data point.

②  $x_n \rightarrow$

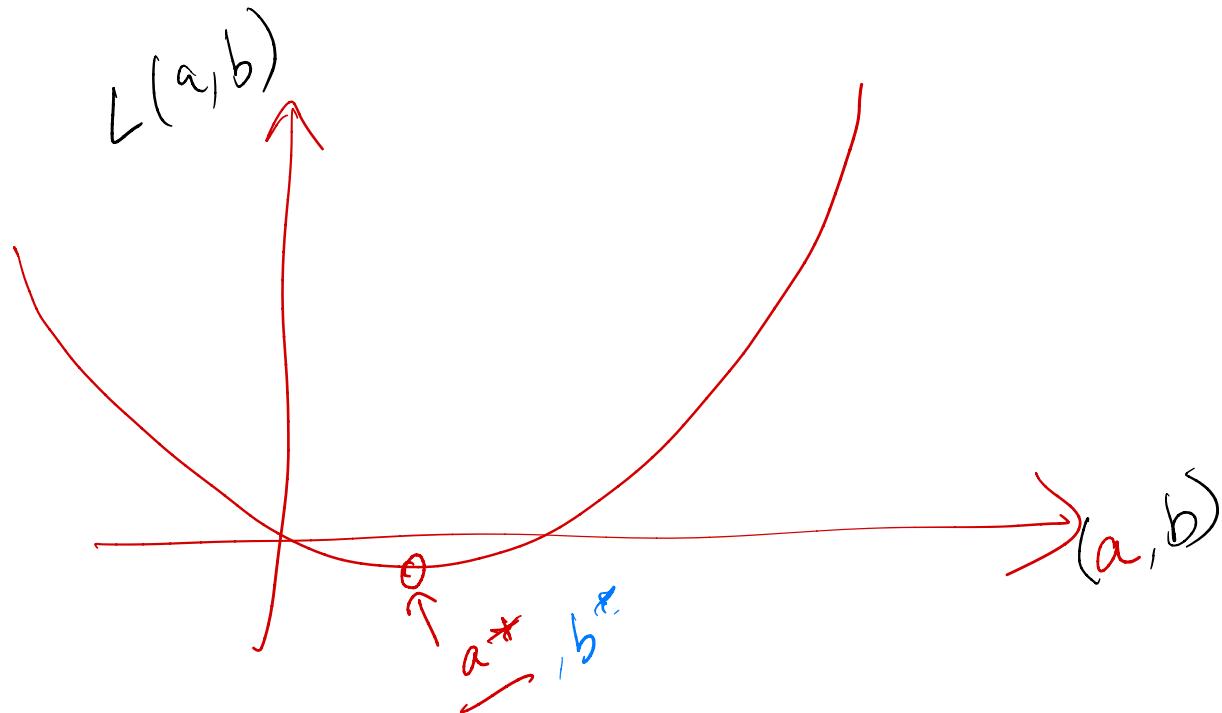
$n^{\text{th}}$  co-ordinate / feature.

# Linear Regression [Optimization / Searching]



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

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



# Linear Regression



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

$$a \in \mathbb{R}^d, b \in \mathbb{R}$$

- 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

$$\min_{a,b} L(a,b)$$

$$a^*: DL(a) = 0$$

$$b^*: DL(b) = 0$$

Solve  $DL\left(\begin{matrix} a^* \\ b \end{matrix}\right) = 0$

## linear regression work flow

- ① model  $f: a^T x + b$
- ② loss  $\ell: [a^T x + b - y]^2$   
↑ feet      ↑ label

- ③ opt  $\min_{a, b} \sum_m (a^T x^{(m)} + b - y^{(m)})^2$

↳ opt algo: gradient descent.

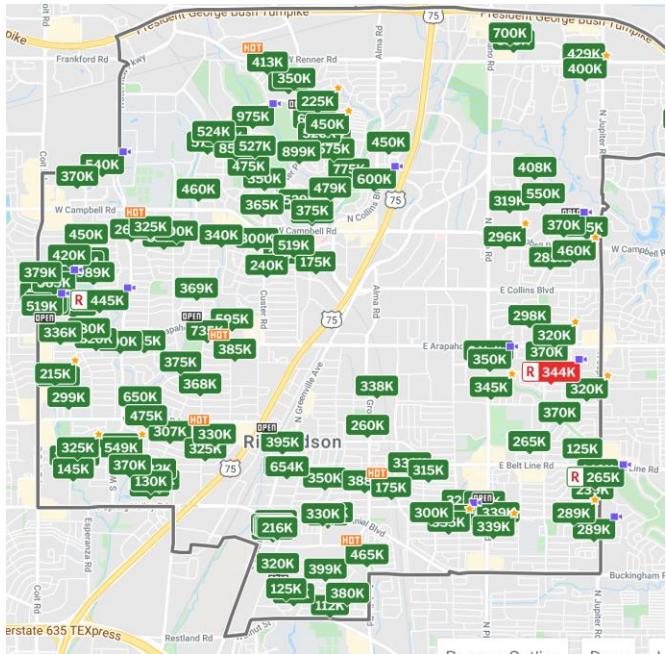
- ④ eval RMS E R2

# Linear Regression

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

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

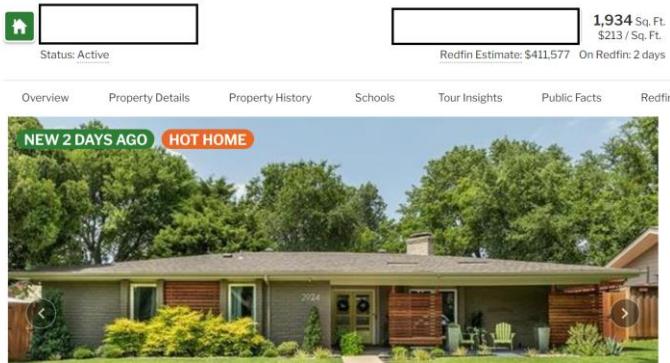
# Recap – Housing Price Prediction Application



$y = \text{Price of house}$

$x = \text{features}$

(loc<sup>n</sup>, size, Bdr(Bath),  
[price], crime, ...)



## Home Facts

Status	Active	Time on Redfin	2 days
Property Type	Residential, Single Family	HOA Dues	\$4/month
Year Built	1969	Style	Single Detached, Mid-Century Modern, Ranch, Traditional
Community	Canyon Creek Country Club 9	Lot Size	10,019 Sq. Ft.
MLS#	14375892		

$y = \text{Price of house}$   
 $x = \text{features}$   
 $(\text{loc}^n, \text{size}, \text{Bd}(Bath), \text{crime}, \dots)$   
 $\text{HOA}, \text{Prop Tax}, \text{Crime}, \dots$

---

# Part II: Gradient Descent and Optimization

# Gradient Descent

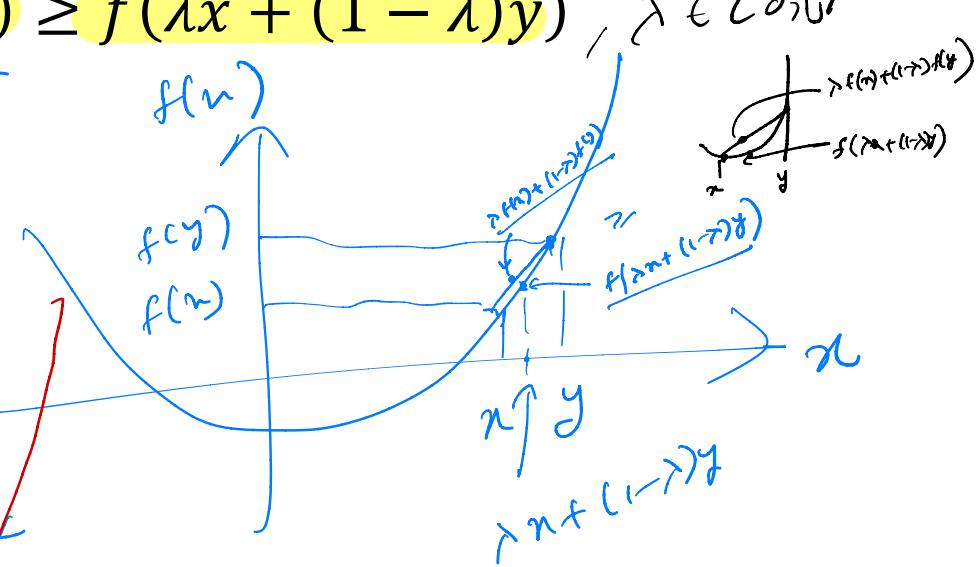
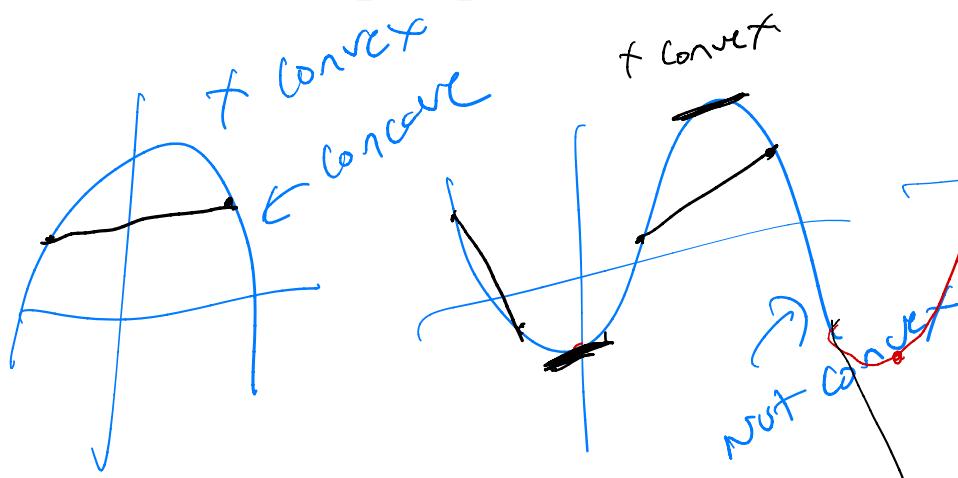


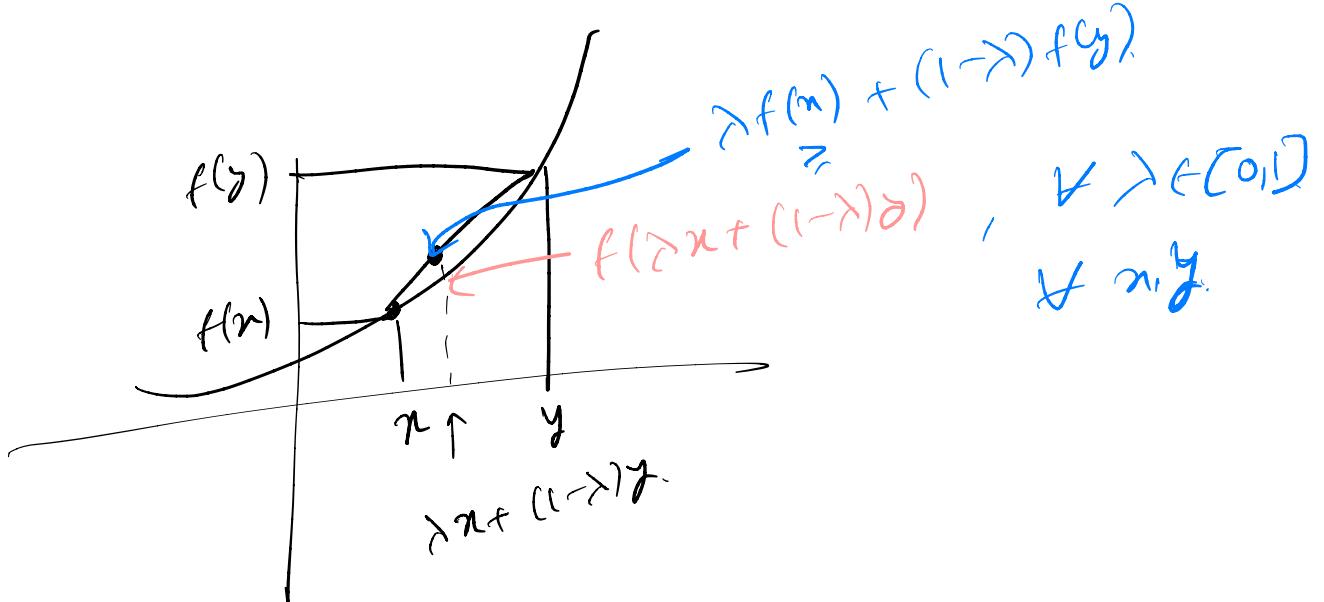
Iterative method to minimize a **(convex)** differentiable function  $f$

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

$$\lambda f(x) + (1 - \lambda)f(y) \geq f(\lambda x + (1 - \lambda)y), \quad \lambda \in [0, 1]$$

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





# Gradient Descent



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

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

$$L(a, b) = \sum_m [a^T x^{(m)} + b - y^{(m)}]^2$$

$\theta$        $M$       convex

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

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

Pick  $a_0, b_0$  random / zeros

$$\left. \begin{array}{l} a_{t+1} = a_t - r_t \nabla L(a_t) \\ b_{t+1} = b_t - r_t \nabla L(b_t) \end{array} \right\} \text{repeat } k \text{ times}$$

until convergence.

$$\theta_{t+1} = \theta_t - \gamma_t \nabla f(\theta_t)$$

Learning Rate.

Convergence

- ①  $x^t \rightarrow x^{t+1}$
- $$|f(x^{t+1}) - f(x^t)| \leq \delta [10^{-5}]$$
- ② Fix number of iterations  
 $\|\nabla f(x^t)\| \leq \delta [10^{-5}]$
- ③  $\|x^{t+1} - x^t\| \leq \delta$
- ④  $\|x^t\| \leq \delta$

# Basics of Convexity and Gradient Desc

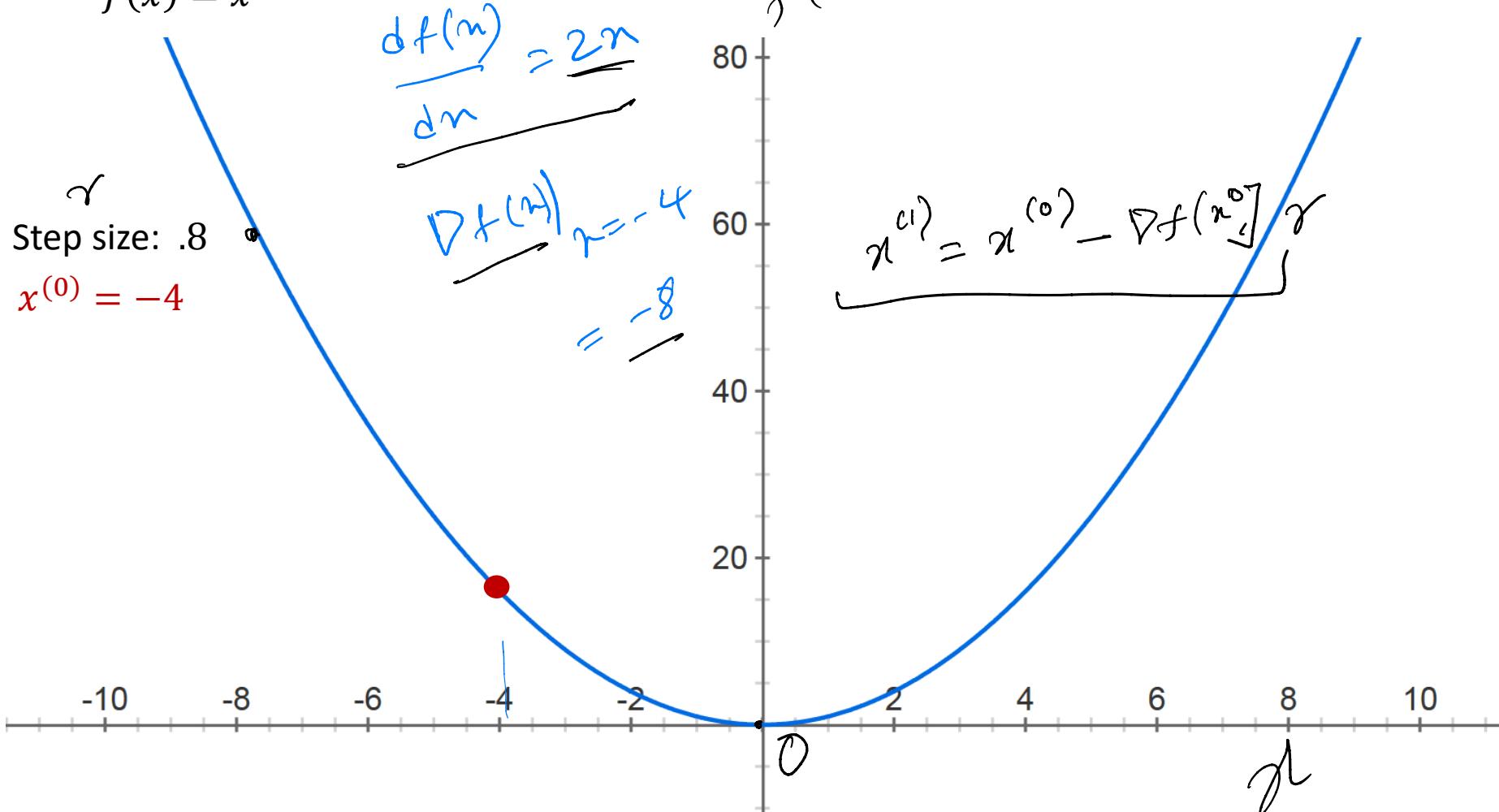


- For additional reading, please see some of my slides from my Spring 2020 Class “Optimization in Machine Learning”  
<sup>+ 2021</sup>
- Github Location for Lecture Notes and Slides:  
<https://github.com/rishabhk108/OptimizationML>
- Please skim through:
  - Lectures 1 and 2 for basics
  - Lectures 3-5 for convex functions
  - Lectures 6-8 on Gradient Descent
  - This includes slightly more mathematical details like convergence analysis and proofs for convergence etc.

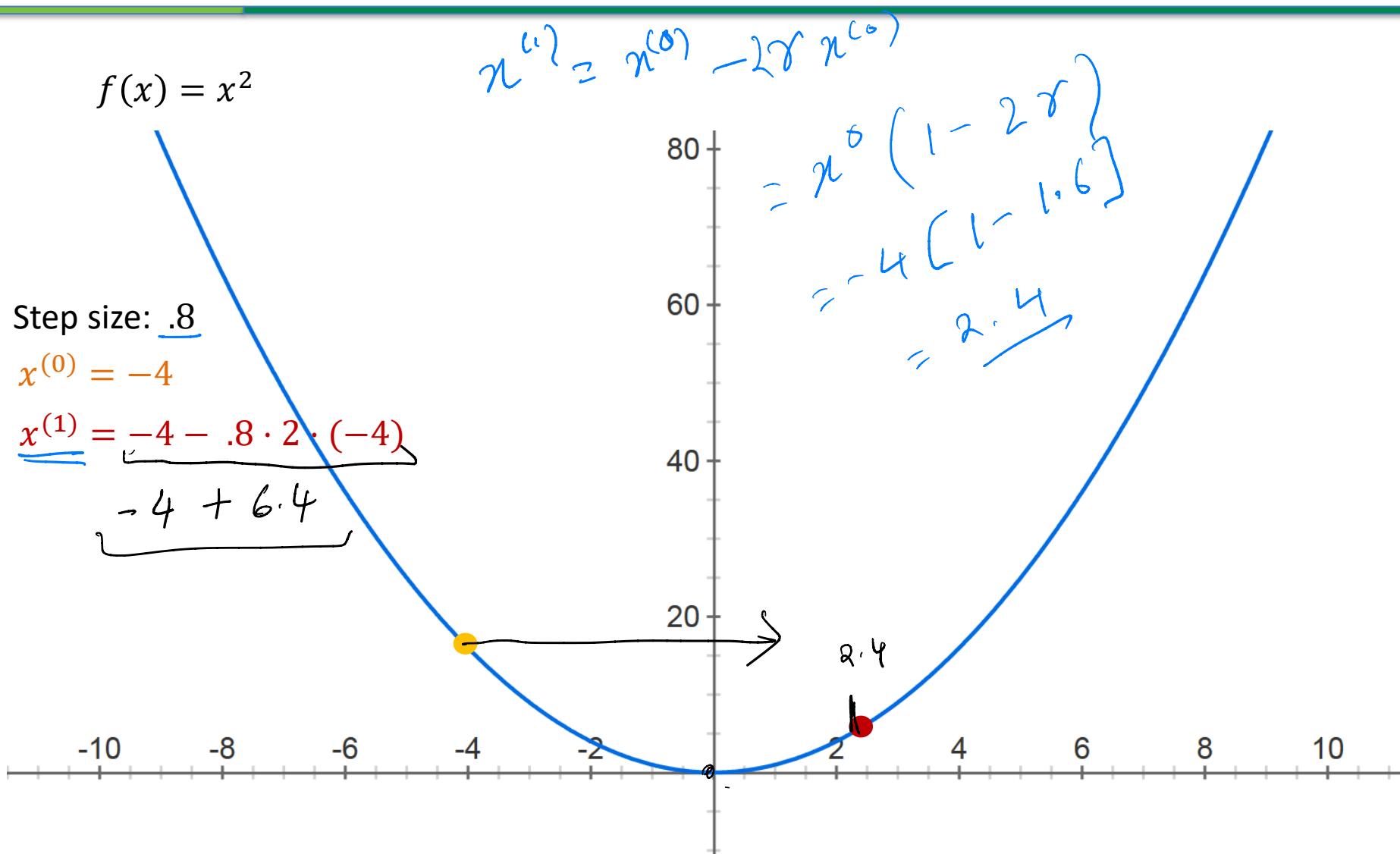
# Gradient Descent

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

Step size: .8  
 $x^{(0)} = -4$



# Gradient Descent



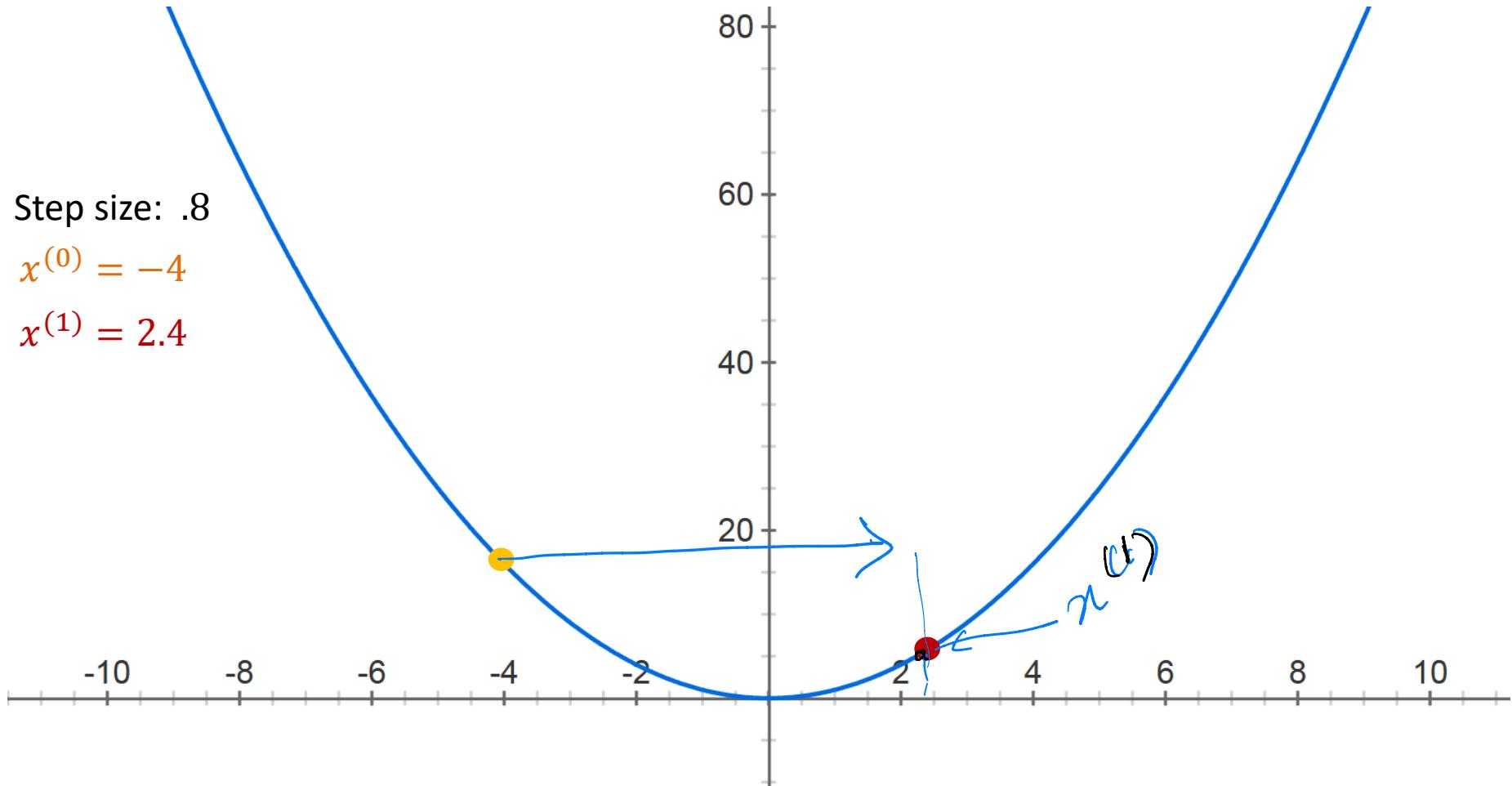
# Gradient Descent

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

Step size: .8

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

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



# Gradient Descent

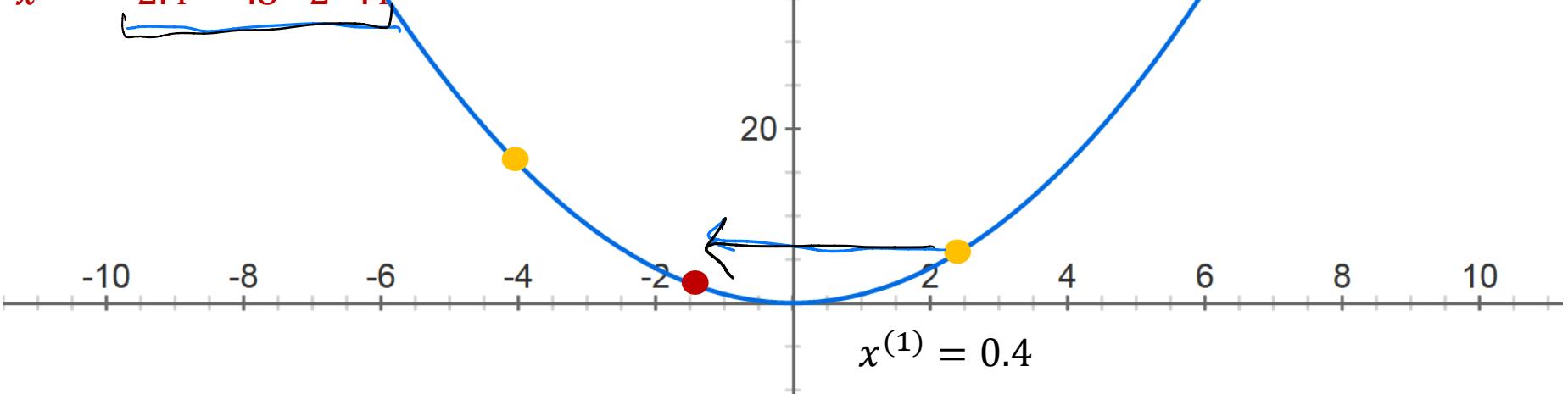
$$f(x) = x^2$$

Step size: .8

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

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

$$x^{(2)} = 2.4 - .8 \cdot 2 \cdot .4$$



# Gradient Descent

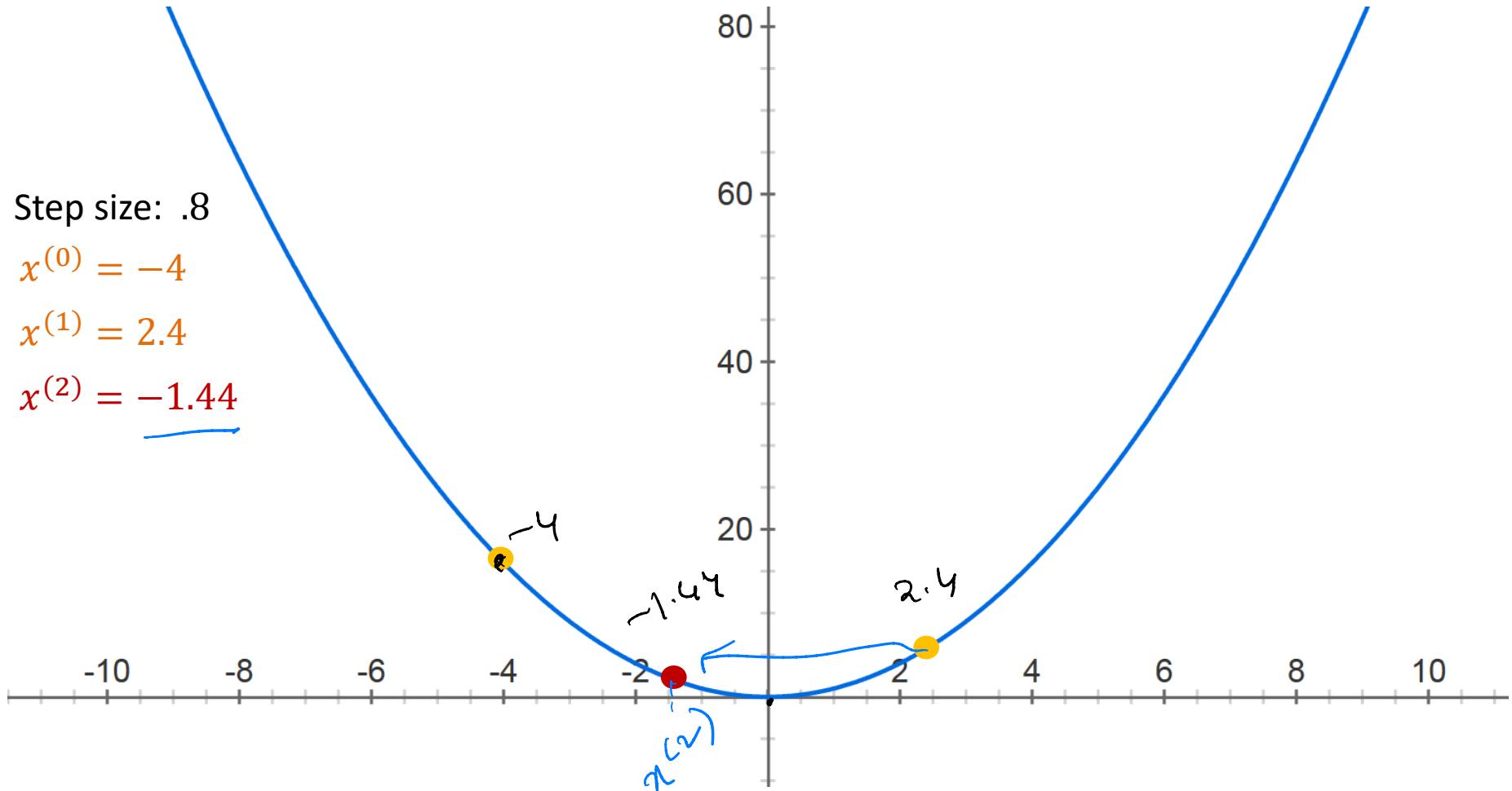
$$f(x) = x^2$$

Step size: .8

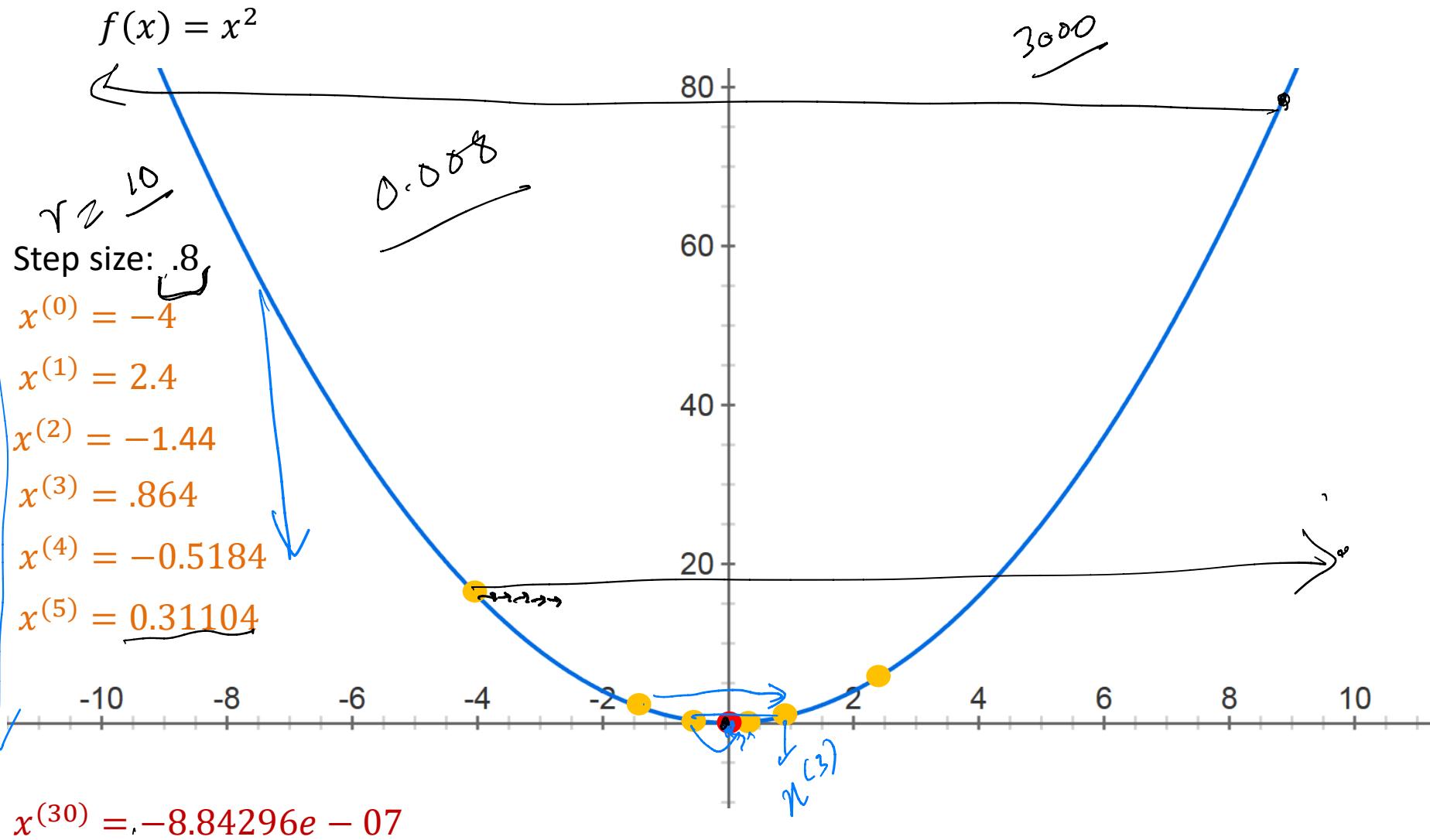
$$x^{(0)} = -4$$

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

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



# Gradient Descent (Numerical Algo)



# Gradient Descent



$$a, n^{(m)} \in \mathbb{R}$$

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

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

$$\frac{\partial L}{\partial a} = \frac{1}{M} \sum_m 2(ax^{(m)} + b - y^{(m)}) \underbrace{x^{(m)}}_{\text{gradient}}$$

# Derivation of the Gradient (one dim)



$$f(a, b) = \frac{1}{M} \sum_{m=1}^M [a \cdot x^{(m)} + b - y^{(m)}]^2$$

$$\frac{\partial f}{\partial a} = \nabla f(a) = \frac{1}{M} \sum_{m=1}^M 2 \left[ a \cdot x^{(m)} + b - y^{(m)} \right] x^{(m)}$$

$$\nabla f(b) = \frac{1}{M} \sum_{m=1}^M 2 \left[ a x^{(m)} + b - y^{(m)} \right]$$

$$a^{(t+1)} = a^{(t)} - r_b \nabla f(a) \Big|_{a=a^t}$$

$$b^{(t+1)} = b^{(t)} - r_x \nabla f(b) \Big|_{b=b^t}$$

# Linear Regression

- 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

# Derivation of the Gradient (> 1 dim)



$$f(a, b) = \frac{1}{M} \sum_{m=1}^M [a^T x^{(m)} + b - y^{(m)}]^2$$

$$\nabla f(a) = \frac{1}{M} \sum_{m=1}^M 2 \underbrace{[a^T x^{(m)} + b - y^{(m)}]}_{\text{Scalar}} x^{(m)} \in \mathbb{R}^n$$

$\nabla f(b)$ : same as 1d calc.

$$g(a) = a^T x^{(m)}$$
$$\nabla g(a) = x^{(m)}$$
$$\nabla f(a) = \begin{cases} \frac{\partial g}{\partial a_1}, & \frac{\partial g}{\partial a_1} \\ \vdots & \vdots \\ \frac{\partial g}{\partial a_n}, & \frac{\partial g}{\partial a_n} \end{cases}$$

$$g(a) = \sum_{i=1}^n a_i x_i^{(m)}$$
$$\frac{\partial g}{\partial a_i} = x_i^{(m)}$$

# Gradient Descent

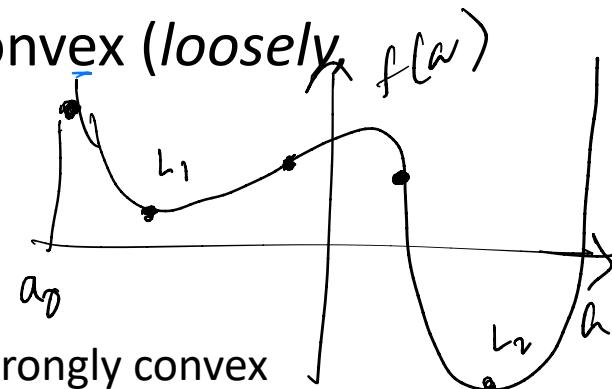
$f = \text{convex}$

- 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
  - Very fast convergence because the Linear Regression is *smooth* (loosely, think *differentiable*) and strongly convex (*loosely bounded below by a quadratic function*)\*

local minima



\* See Lectures 6-8 for better understanding of smooth and strongly convex



---

# Part III: Polynomial Regression

# Polynomial Regression

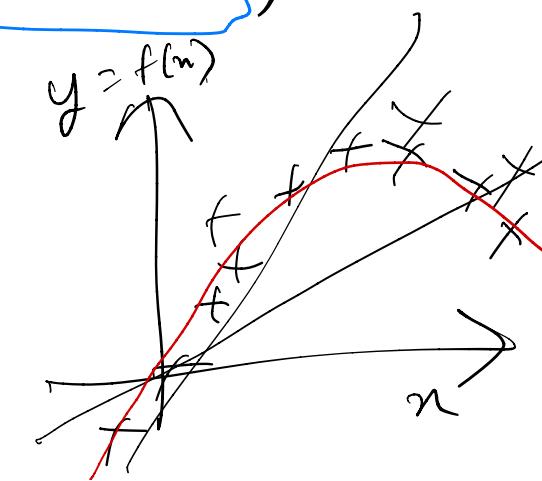
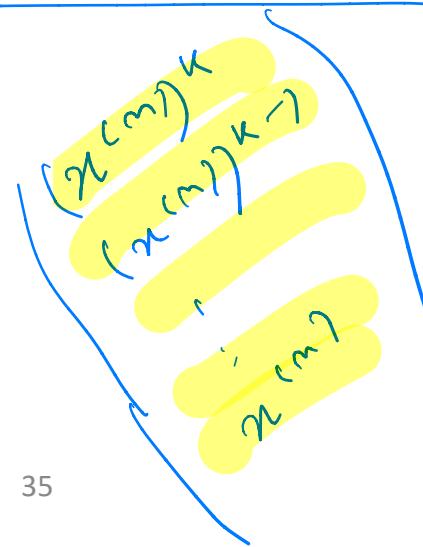
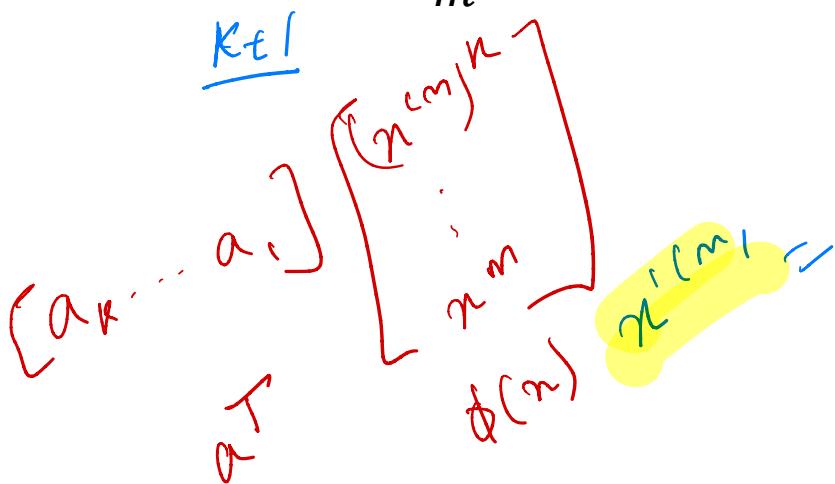
$$(n/d = 1)$$



- What if we enlarge the hypothesis class?  $\dim$

- Quadratic functions:  $ax^2 + bx + c = y$ .
- $k$ -degree polynomials:  $a_k x^k + a_{k-1} x^{k-1} + \dots + a_1 x + a_0$

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



## Polynomial Regression as Linear regression

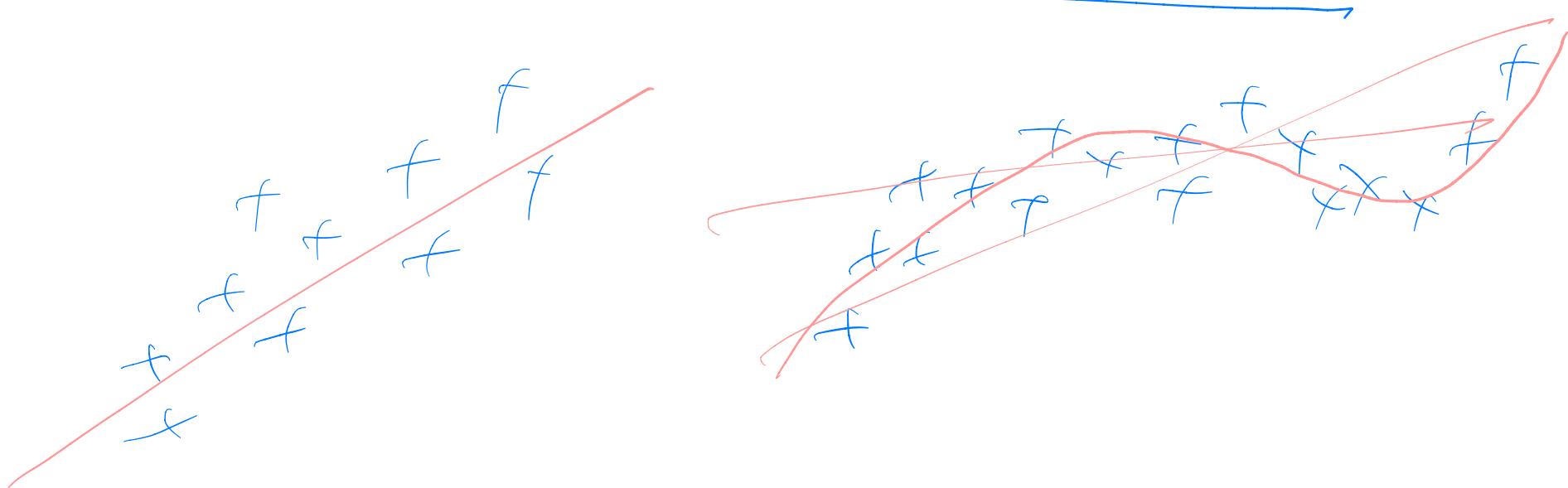
$$L(a_0, \dots, a_k) = \sum_m \left[ a_k (x^{(m)})^k + a_{k-1} (x^{(m)})^{k-1} + \dots + a_0 - y \right]^2$$

$[a_k, a_{k-1}, \dots, a_0]$        $\begin{bmatrix} (x^{(m)})^k \\ (x^{(m)})^{k-1} \\ \vdots \\ x^{(m)} \end{bmatrix}$        $- y$ 
 $^2$

1D Poly-Regression  $\equiv$  k-Dim Lin Regression with poly features

# Polynomial Regression

- What if we enlarge the hypothesis class?
  - Quadratic functions:  $ax^2 + bx + c$
  - $k$ -degree polynomials:  $a_kx^k + a_{k-1}x^{k-1} + \dots + a_1x + a_0$
- Can we always learn “better” with a larger hypothesis class?



# Polynomial Regression



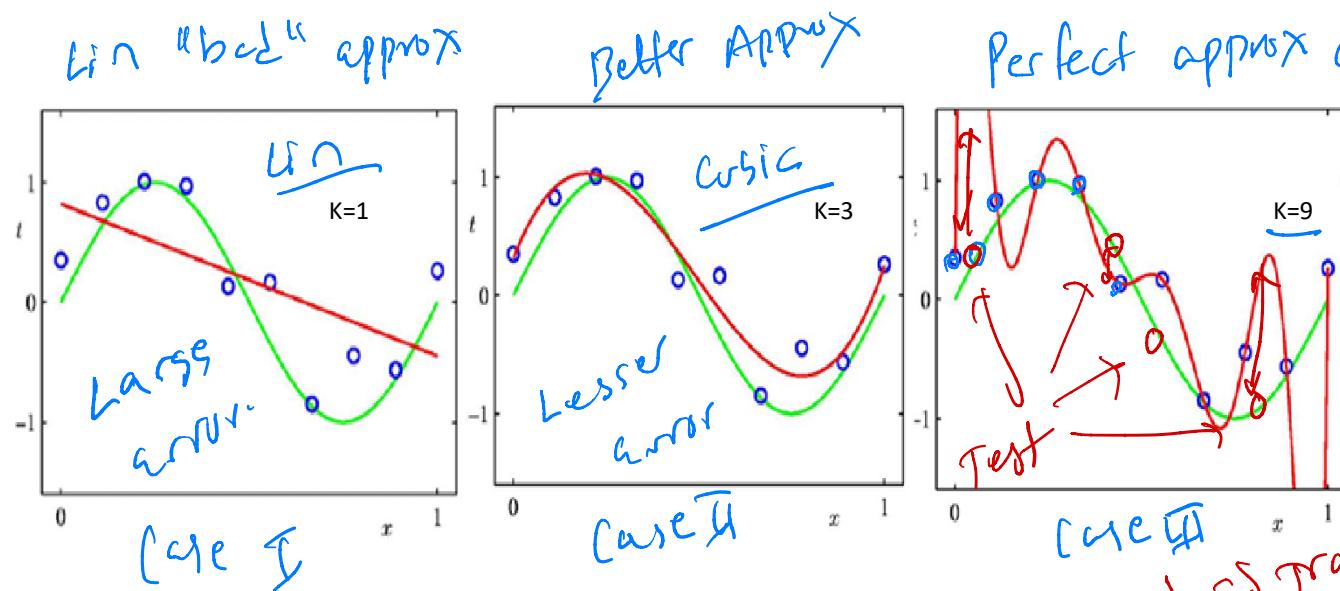
- What if we enlarge the hypothesis class?

- Quadratic functions:  $ax^2 + bx + c$

Overfitting ( $K=9$ )

- $k$ -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?



Training  
Perfect approx on Given set of Points

Excellent  $\Rightarrow$  Train ( $O^n$ )  
Poor  $\Rightarrow$  Test

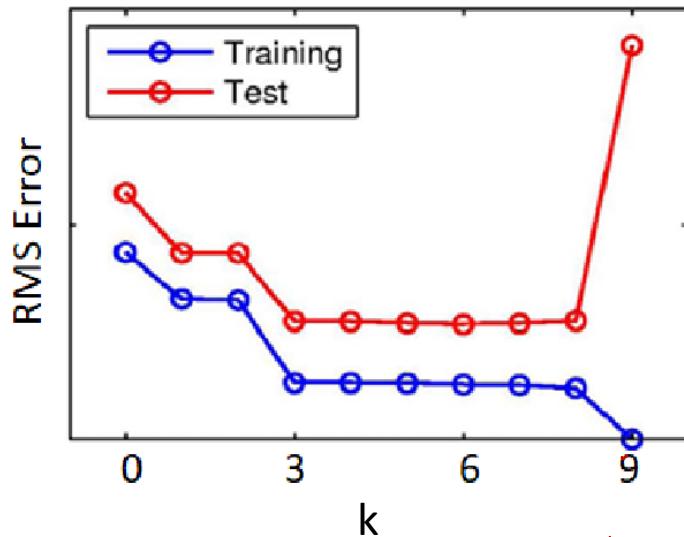
# Caveats with Polynomial Regression



- Lesser training Error
- Larger hypothesis space always decreases the cost function, but this does NOT necessarily mean better predictive performance *may not generalize better*
  - This phenomenon is known as **overfitting**
  - Ideally, we would select the **simplest** hypothesis consistent with the observed data *GENERALIZATION*
  - 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)

# Overfitting

- As the *degree of the polynomial ( $k$ )* increases, training error decreases monotonically
- As  $k$  increases test error can increase
- Test error can decrease at first, but increases
- Overfitting can occur
  - When the model is too complex and trivially fits the data (i.e., too many parameters)
  - When the data is not enough to estimate the parameters
  - Model captures the noise (or the chance)



Too many parameters

|| Regularization



---

# Part IV: Hands On

# House Price Prediction



- Boston House Price Dataset

UCI Repo

**CRIM:** Per capita crime rate by town

(crime)

**ZN:** Proportion of residential land zoned for lots over 25,000 sq. ft

**INDUS:** Proportion of non-retail business acres per town

**CHAS:** Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)

**NOX:** Nitric oxide concentration (parts per 10 million)

**RM:** Average number of rooms per dwelling

**AGE:** Proportion of owner-occupied units built prior to 1940

**DIS:** Weighted distances to five Boston employment centers

**RAD:** Index of accessibility to radial highways

**TAX:** Full-value property tax rate per \$10,000

**PTRATIO:** Pupil-teacher ratio by town

**B:**  $1000(Bk - 0.63)^2$ , where Bk is the proportion of [people of African American descent] by town

**LSTAT:** Percentage of lower status of the population

**MEDV:** Median value of owner-occupied homes in \$1000s

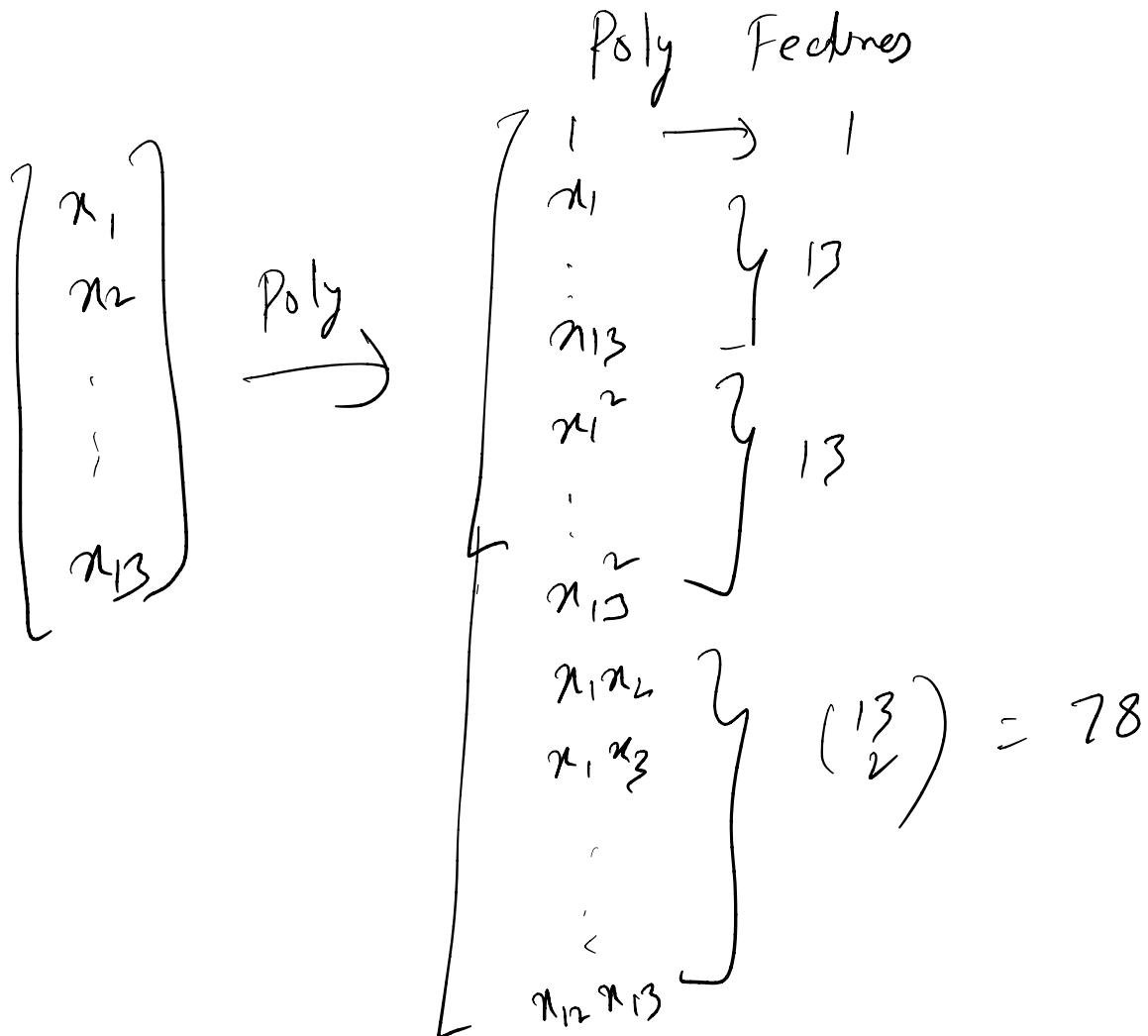
Feet

→ Label Target

# Summary of the Hands On Portion



- Load the Dataset (`sk-learn`)
  - Exploratory Data Analysis (`visual`)
  - Training a Linear Regression Model
  - Training a Polynomial Regression Model
  - Training a Linear/Poly Regression from scratch using gradient descent
- Inbuilt LR in sk-learn
- Contrast to inbuilt LR



## Linear Regression Closed form

$$y = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}$$

$$\begin{aligned} L(w, b) &= \sum_{i=1}^m [w^\top x_i + b - y_i]^2 \\ &= \|X w + b \mathbf{1} - y\|^2 \quad \leftarrow X = \begin{bmatrix} x_1^\top \\ \vdots \\ x_m^\top \end{bmatrix} \end{aligned}$$

$$L(w') = \|X' w' - y\|^2 \quad \text{where} \quad \begin{aligned} X' &= [X \ 1] \\ w' &= (w, b)^T \end{aligned}$$

$$\frac{\partial L}{\partial w'} = 2(X')^\top (X' w' - y) = 0$$

$$X \in \mathbb{R}^{M \times d}$$

$$\Rightarrow (X')^\top X' w' = X' y$$

$$X^\top \in \mathbb{R}^{d \times M}$$

$$\Rightarrow w' = \underbrace{(X')^\top X'}_{(d+1) \times M}^{-1} \underbrace{X' y}_{M \times (d+1)} \quad \Rightarrow (d+1) \times (d+1)$$

$$(M^d) \times O(d^3)$$