

# Simple Regression:

## Linear regression with one input

Emily Fox & Carlos Guestrin

Machine Learning Specialization

University of Washington

# Recall Task: Predicting house prices

# How much is my house worth?



# How much is my house worth?



# Look at recent sales in my neighborhood

- How much did they sell for?



# Regression fundamentals: data, model, task

# Data

*input      output*



( $x_1 = \text{sq.ft.}$ ,  $y_1 = \$$ )



( $x_2 = \text{sq.ft.}$ ,  $y_2 = \$$ )



( $x_3 = \text{sq.ft.}$ ,  $y_3 = \$$ )



( $x_4 = \text{sq.ft.}$ ,  $y_4 = \$$ )



( $x_5 = \text{sq.ft.}$ ,  $y_5 = \$$ )

:

# Data



*input*      *output*  
 $(x_1 = \text{sq.ft.}, y_1 = \$)$



$(x_2 = \text{sq.ft.}, y_2 = \$)$



$(x_3 = \text{sq.ft.}, y_3 = \$)$



$(x_4 = \text{sq.ft.}, y_4 = \$)$



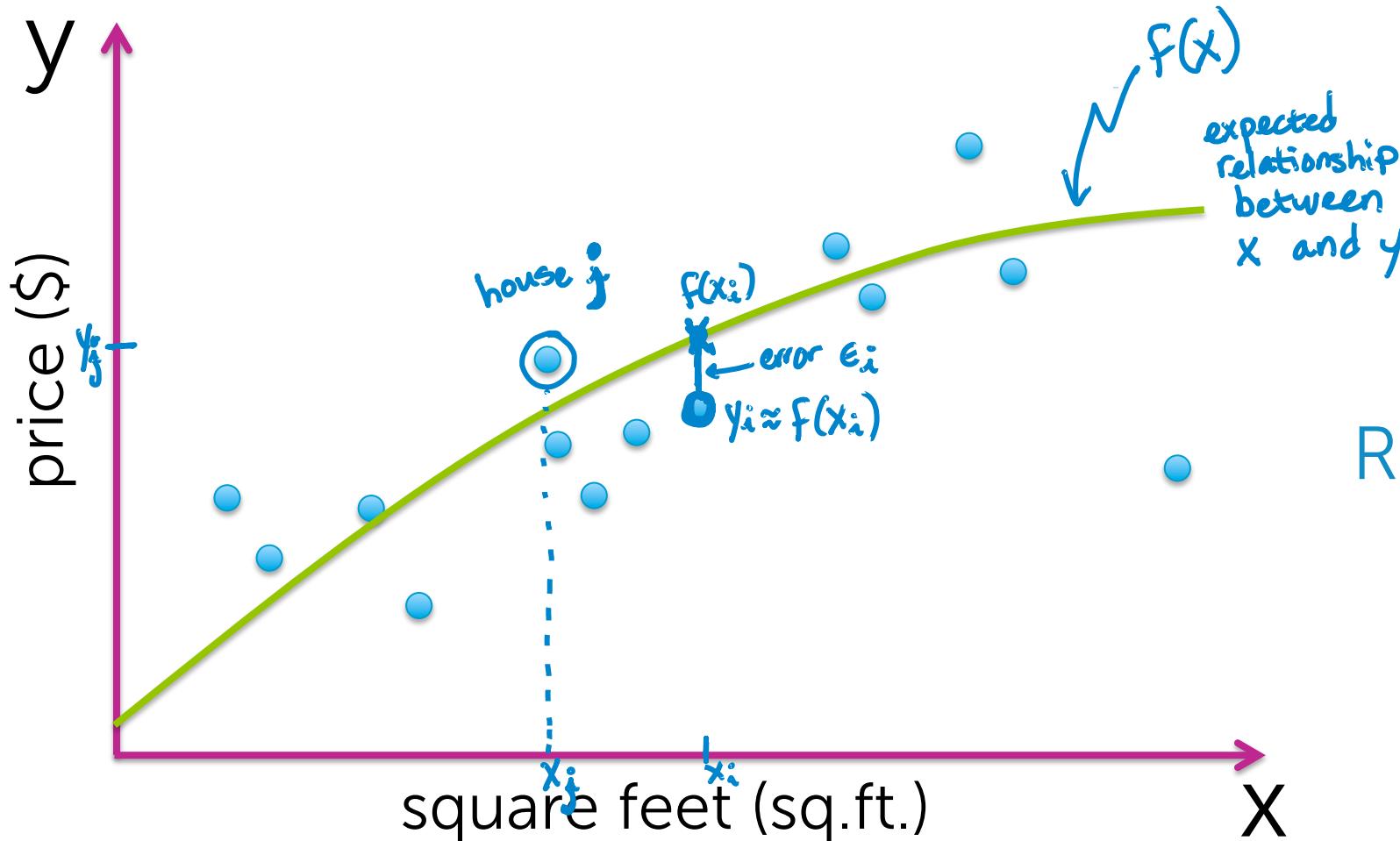
$(x_5 = \text{sq.ft.}, y_5 = \$)$

⋮

## Input vs. Output:

- $y$  is the quantity of interest
- assume  $y$  can be predicted from  $x$

# Model – How we assume the world works



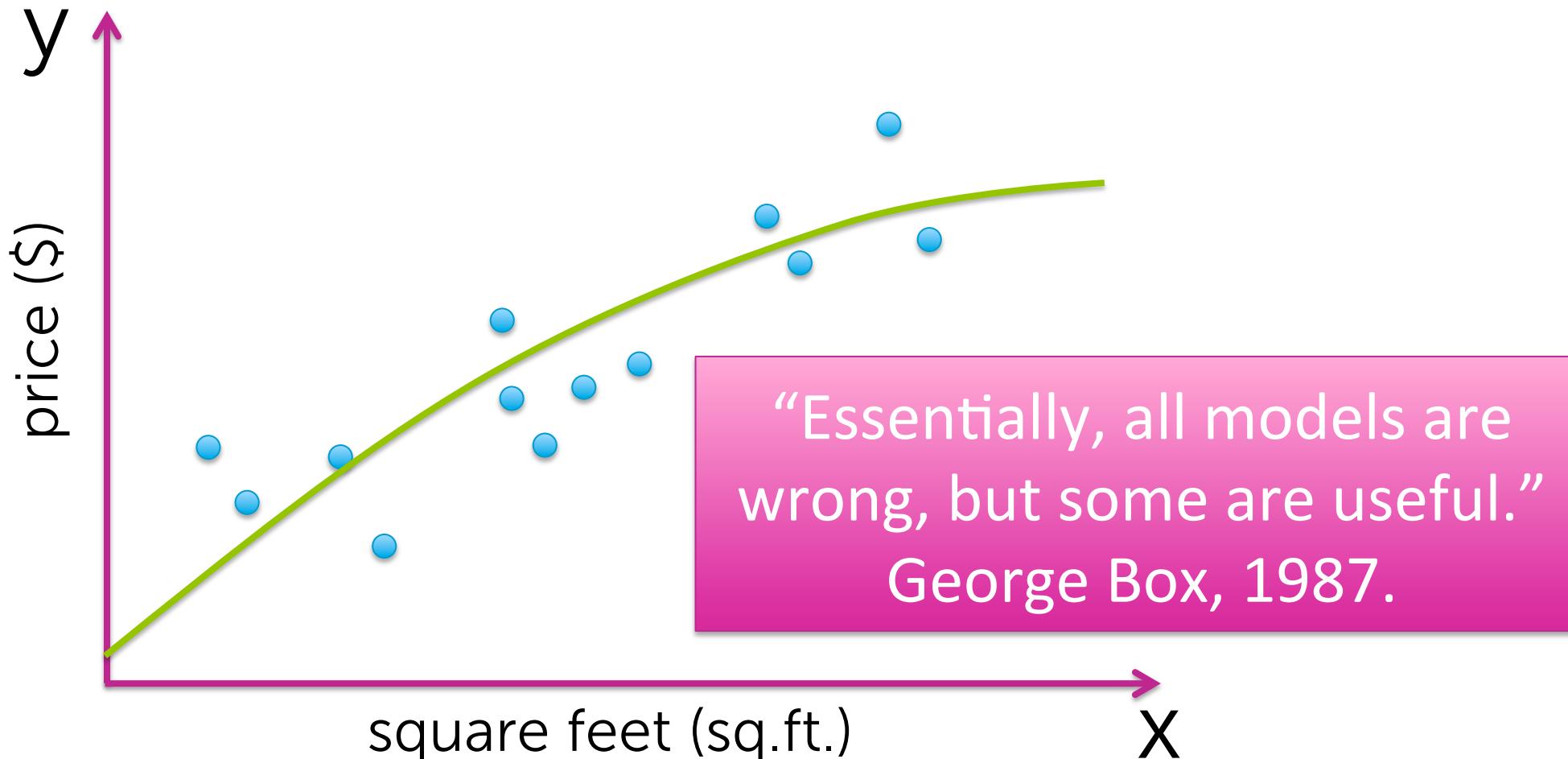
Regression model:

$$y_i = f(x_i) + e_i$$

$E[e_i] = 0$  ← equally likely that error is + or -  
↑ expected value

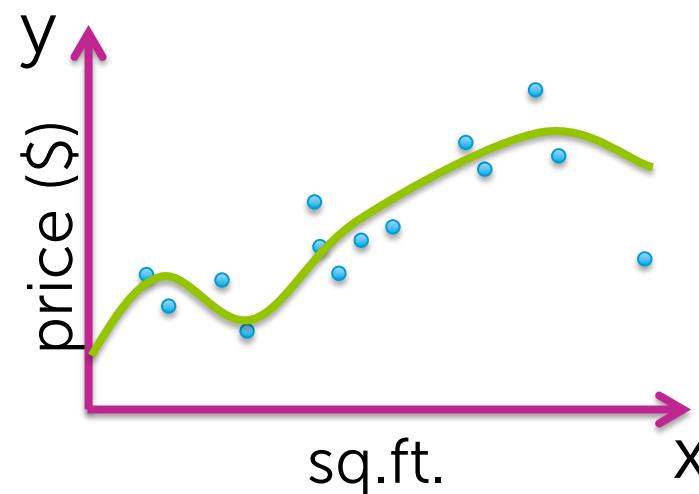
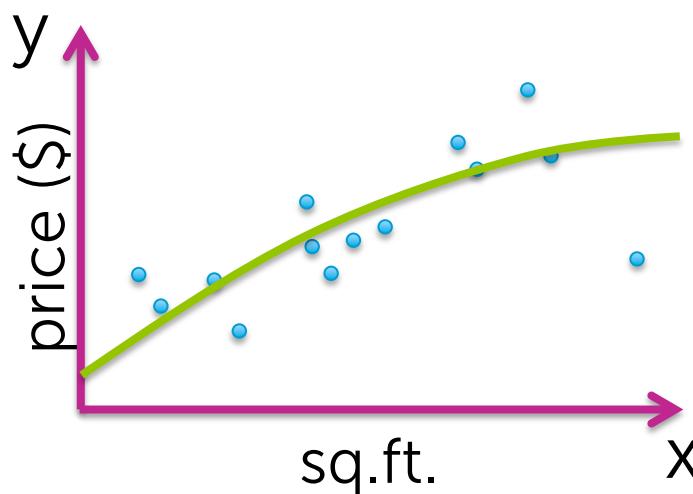
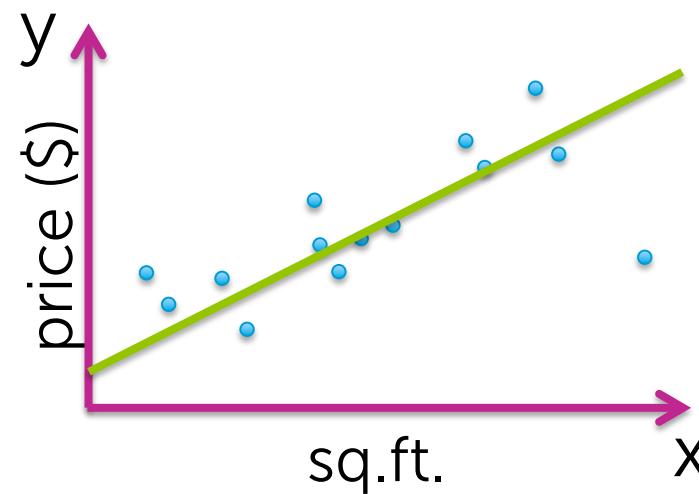
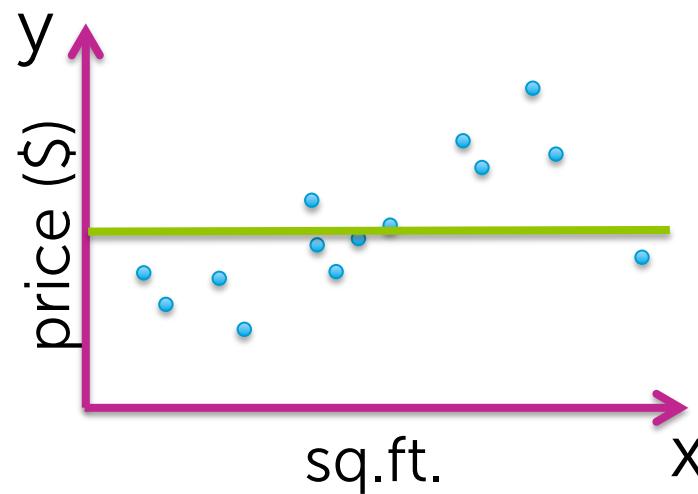
$y_i$  is equally likely to be above or below  $f(x_i)$

# Model – How we *assume* the world works

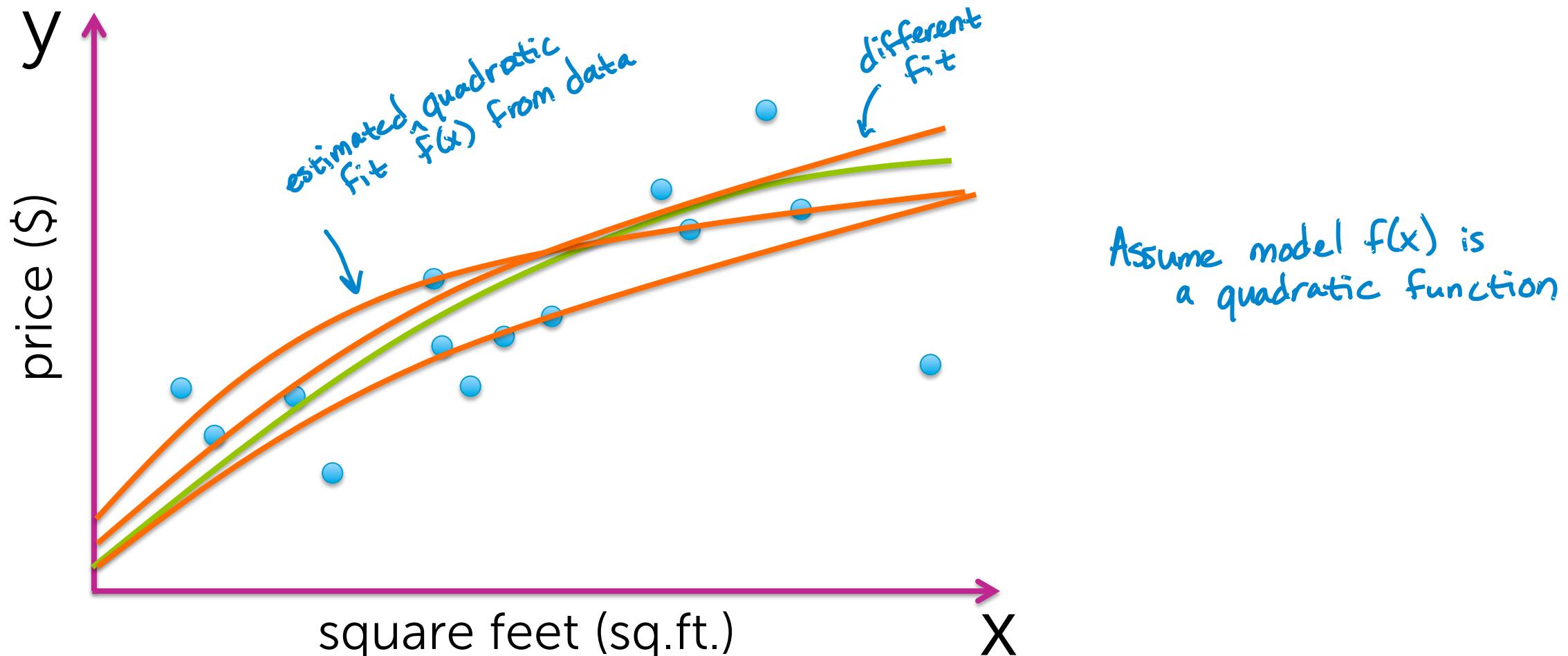


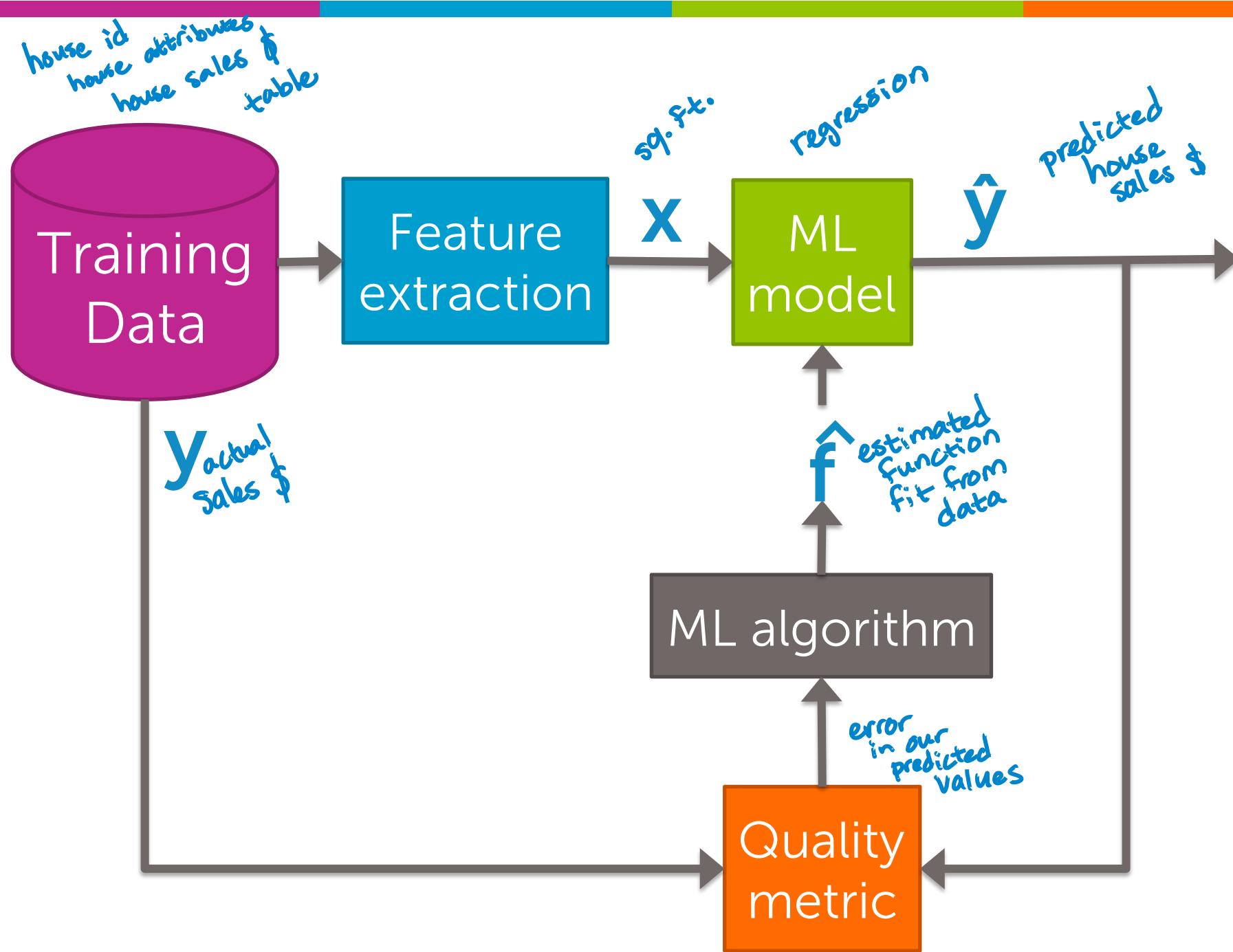
# Task 1–

## Which model $f(x)$ ?

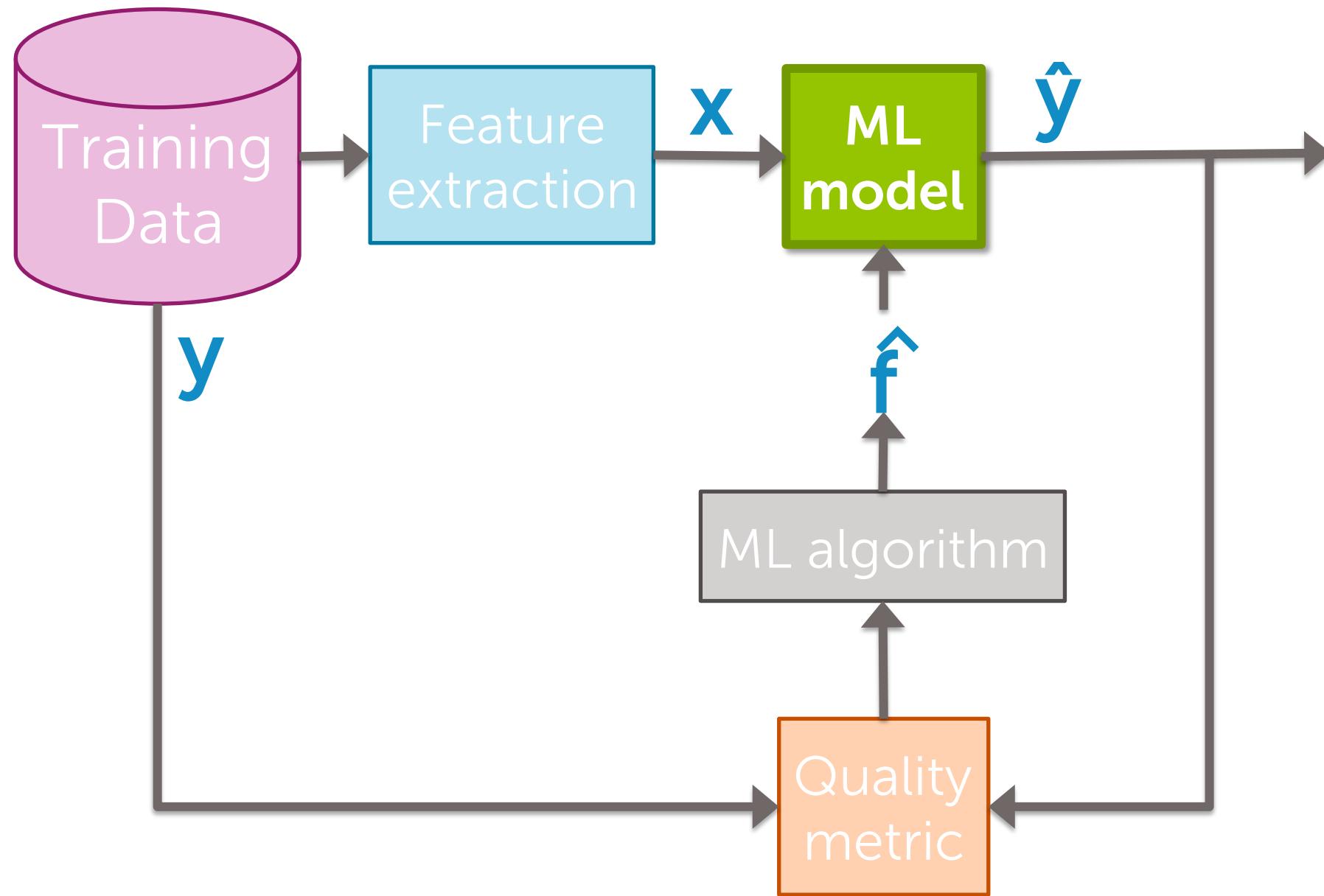


# Task 2 – For a given model $f(x)$ , estimate function $\hat{f}(x)$ from data

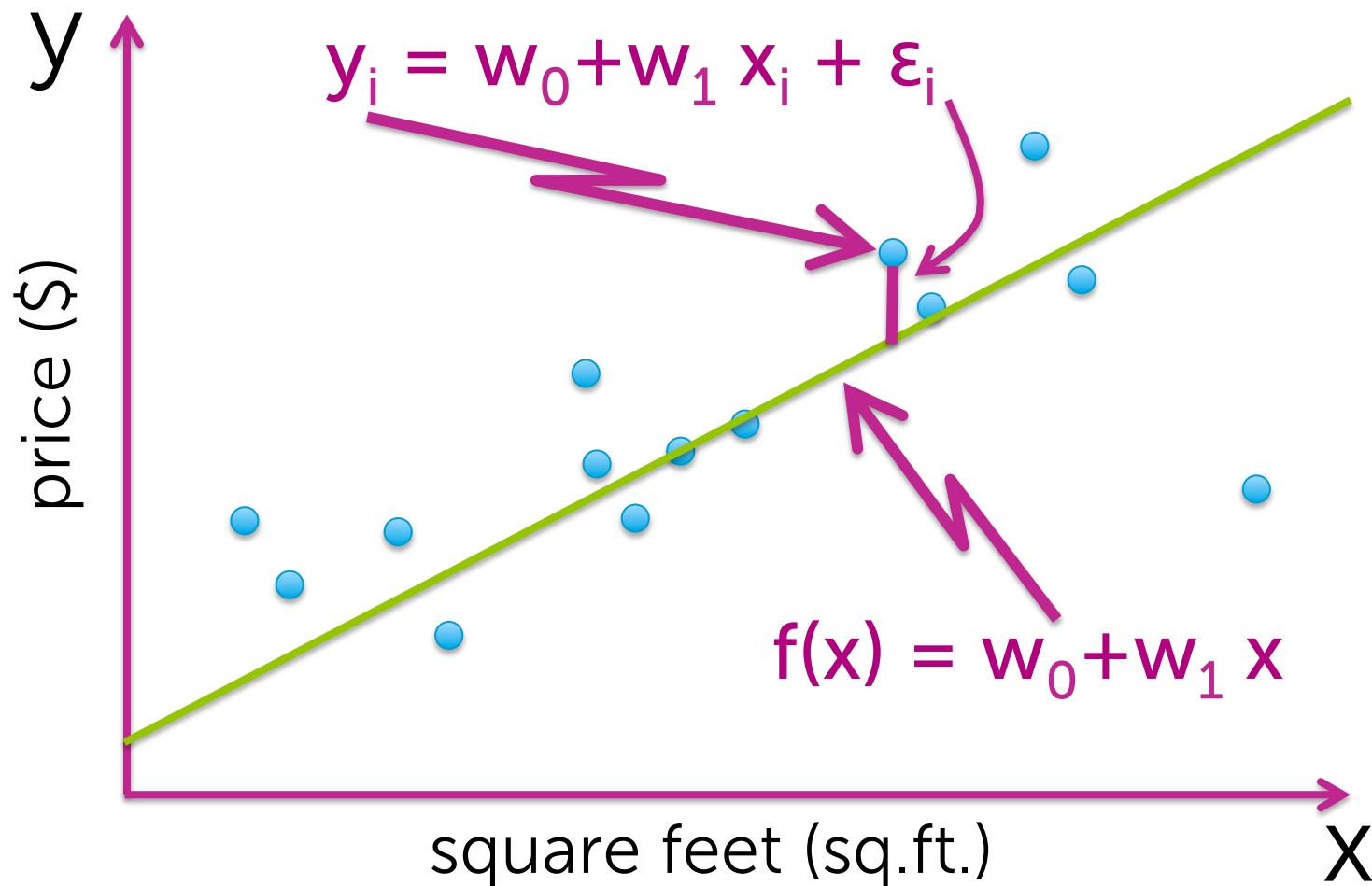




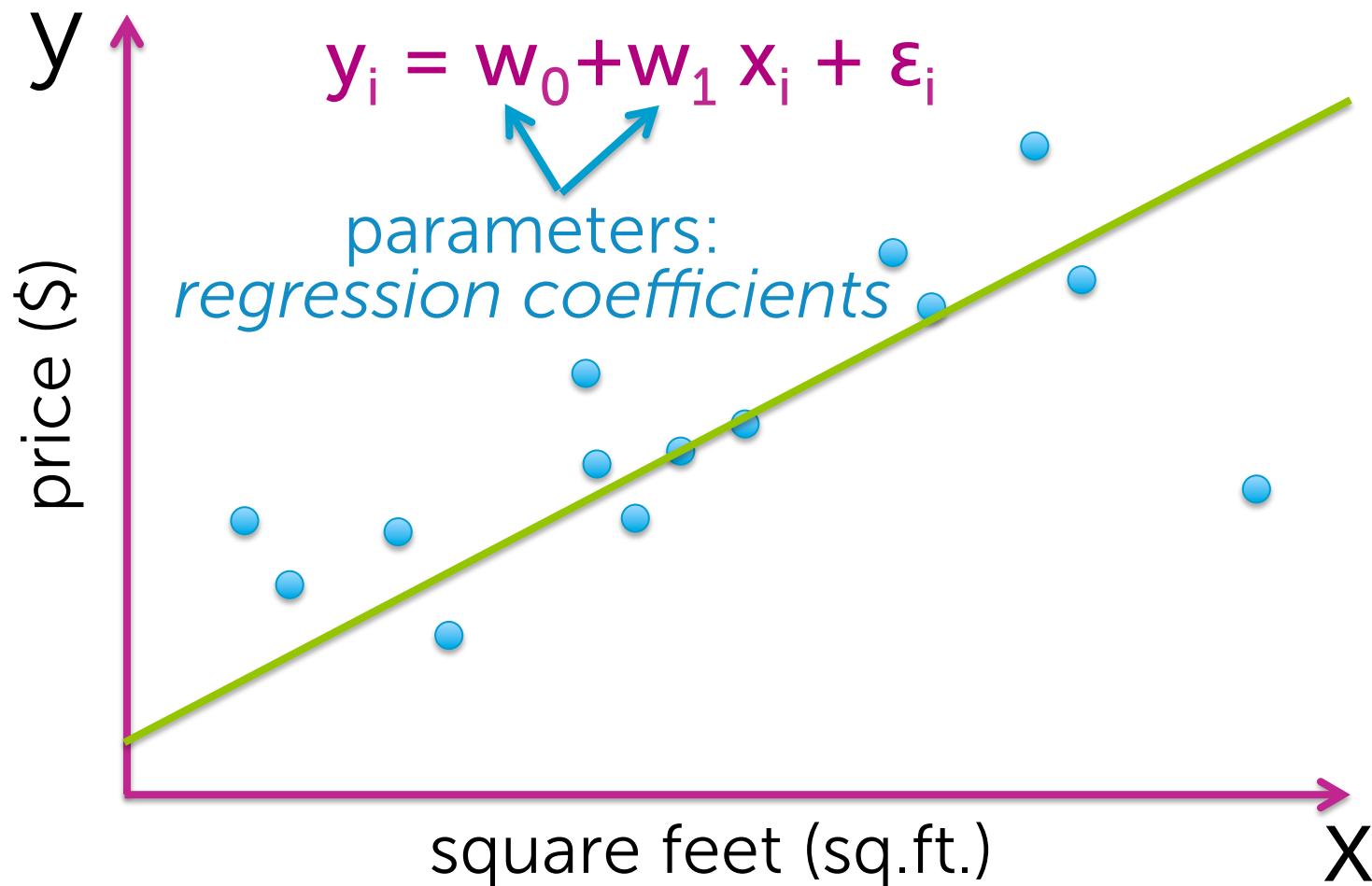
# Simple linear regression



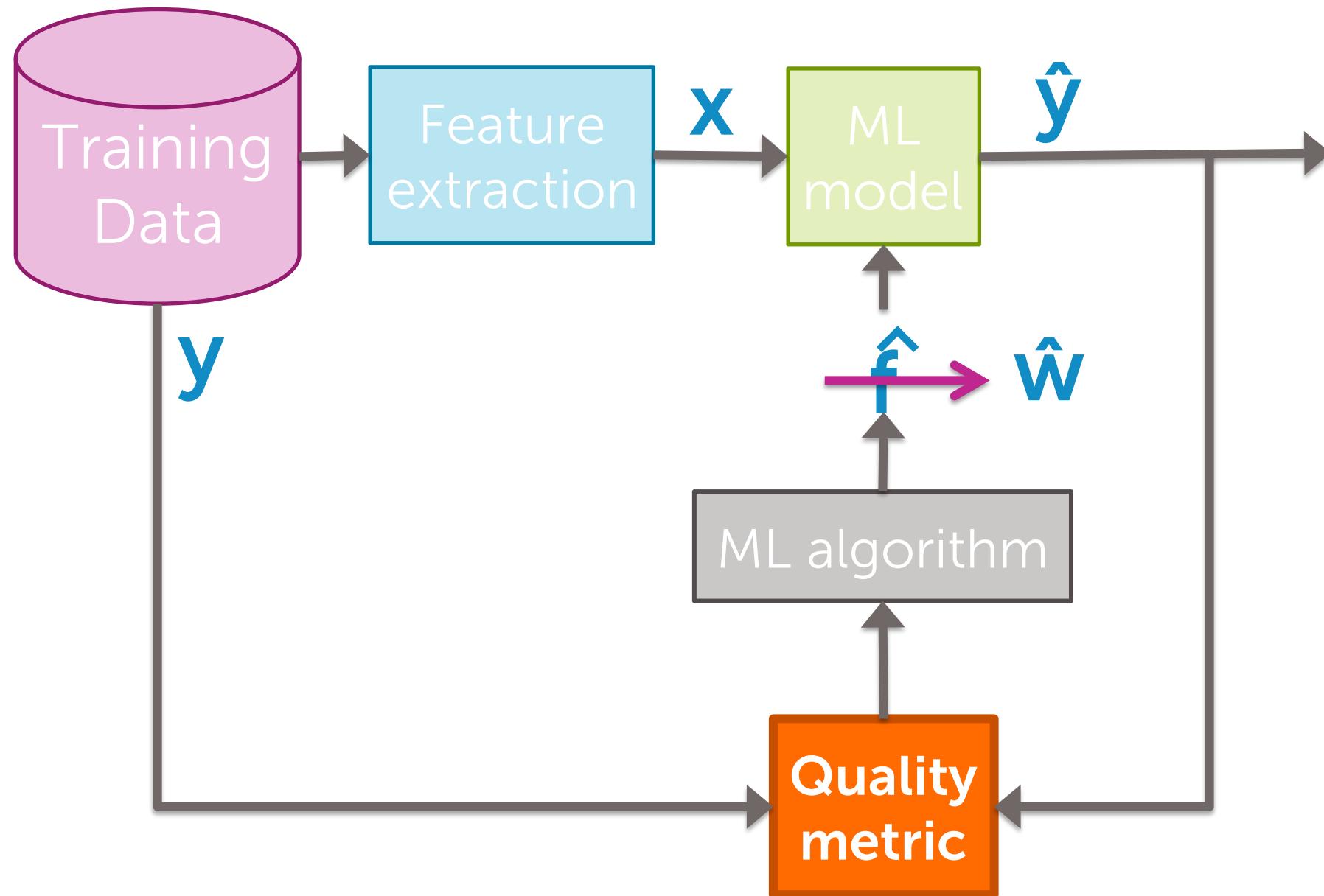
# Simple linear regression model



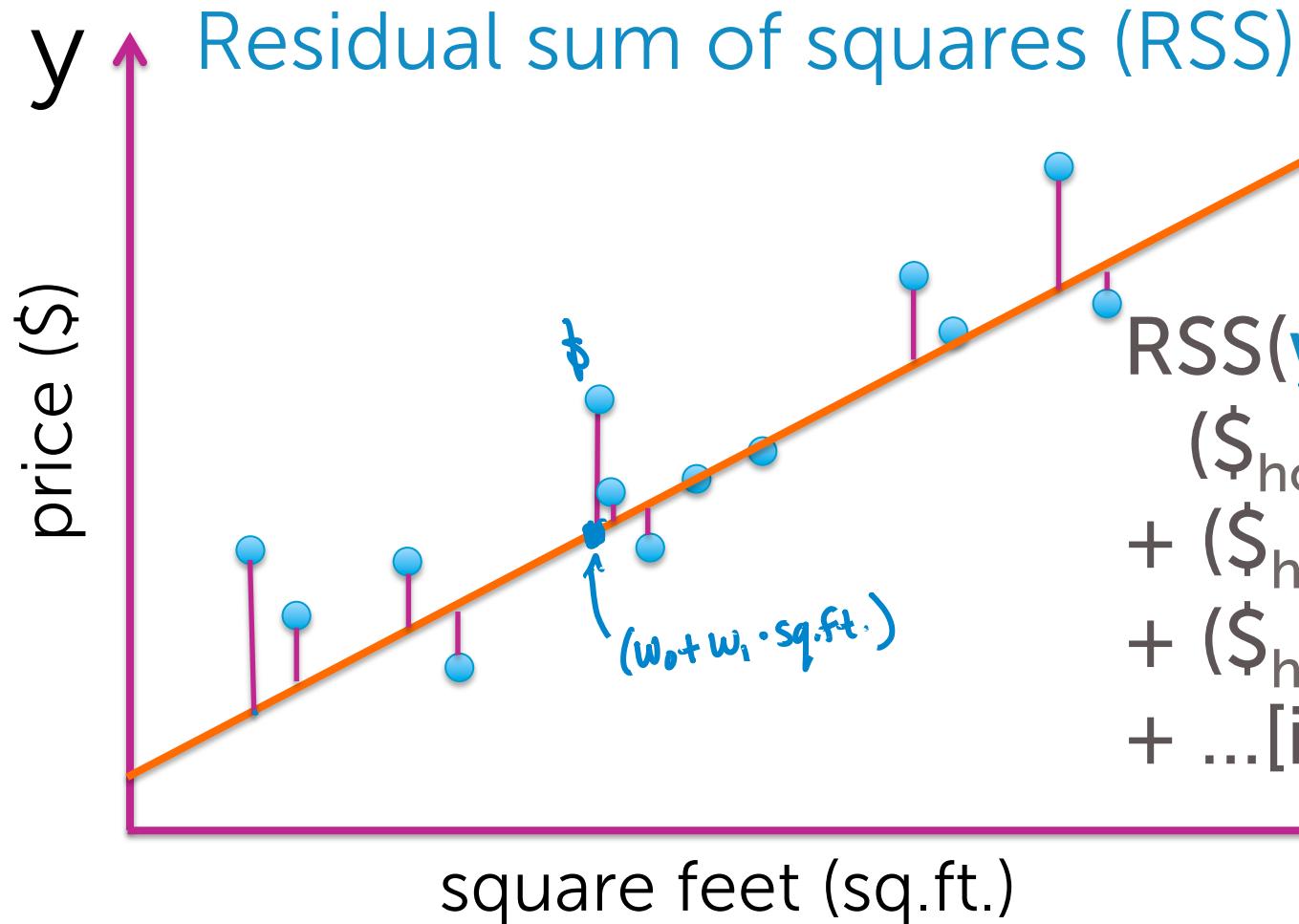
# Simple linear regression model



# Fitting a line to data

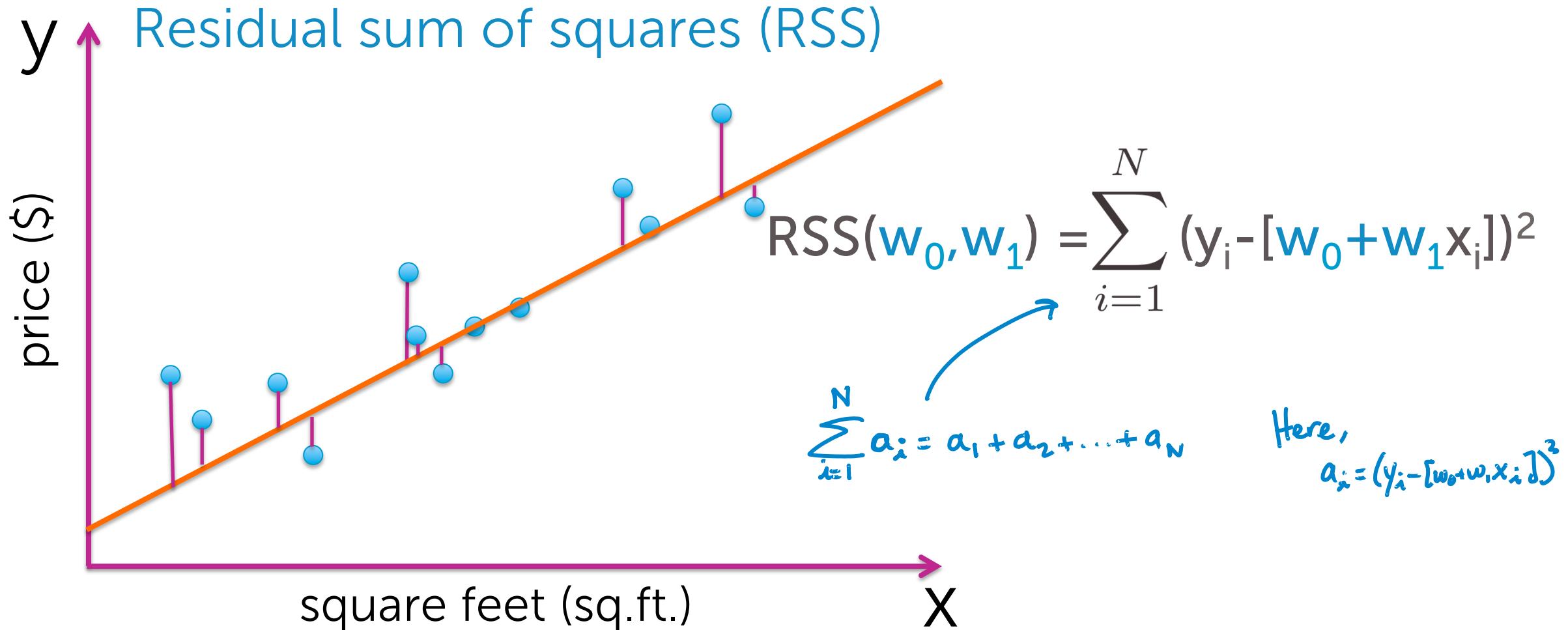


# "Cost" of using a given line

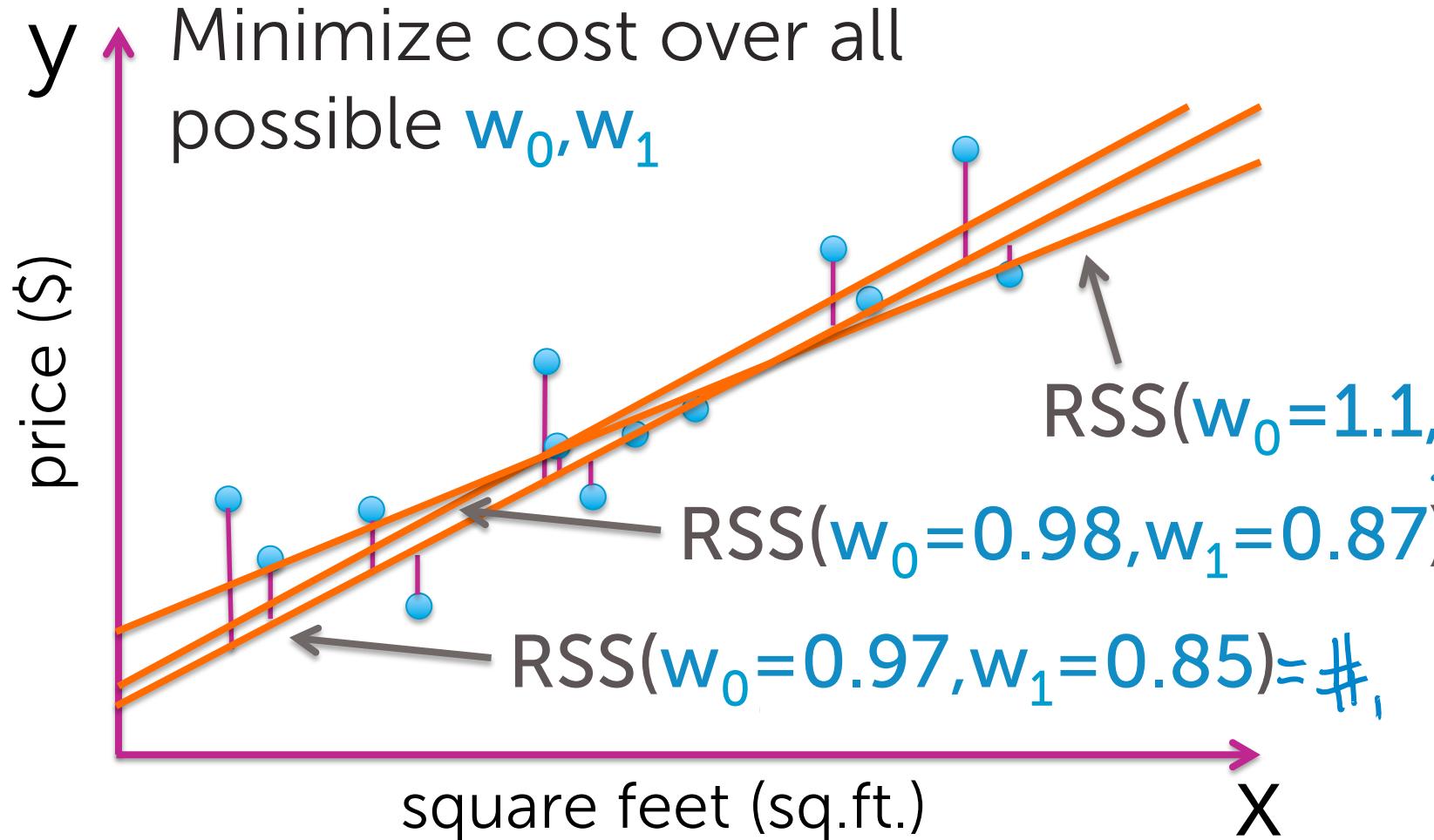


$$\begin{aligned} \text{RSS}(\underline{w}_0, \underline{w}_1) = & (\$_{\text{house 1}} - [\underline{w}_0 + \underline{w}_1 \text{sq.ft.}_{\text{house 1}}])^2 \\ & + (\$_{\text{house 2}} - [\underline{w}_0 + \underline{w}_1 \text{sq.ft.}_{\text{house 2}}])^2 \\ & + (\$_{\text{house 3}} - [\underline{w}_0 + \underline{w}_1 \text{sq.ft.}_{\text{house 3}}])^2 \\ & + \dots \text{[include all training houses]} \end{aligned}$$

# "Cost" of using a given line



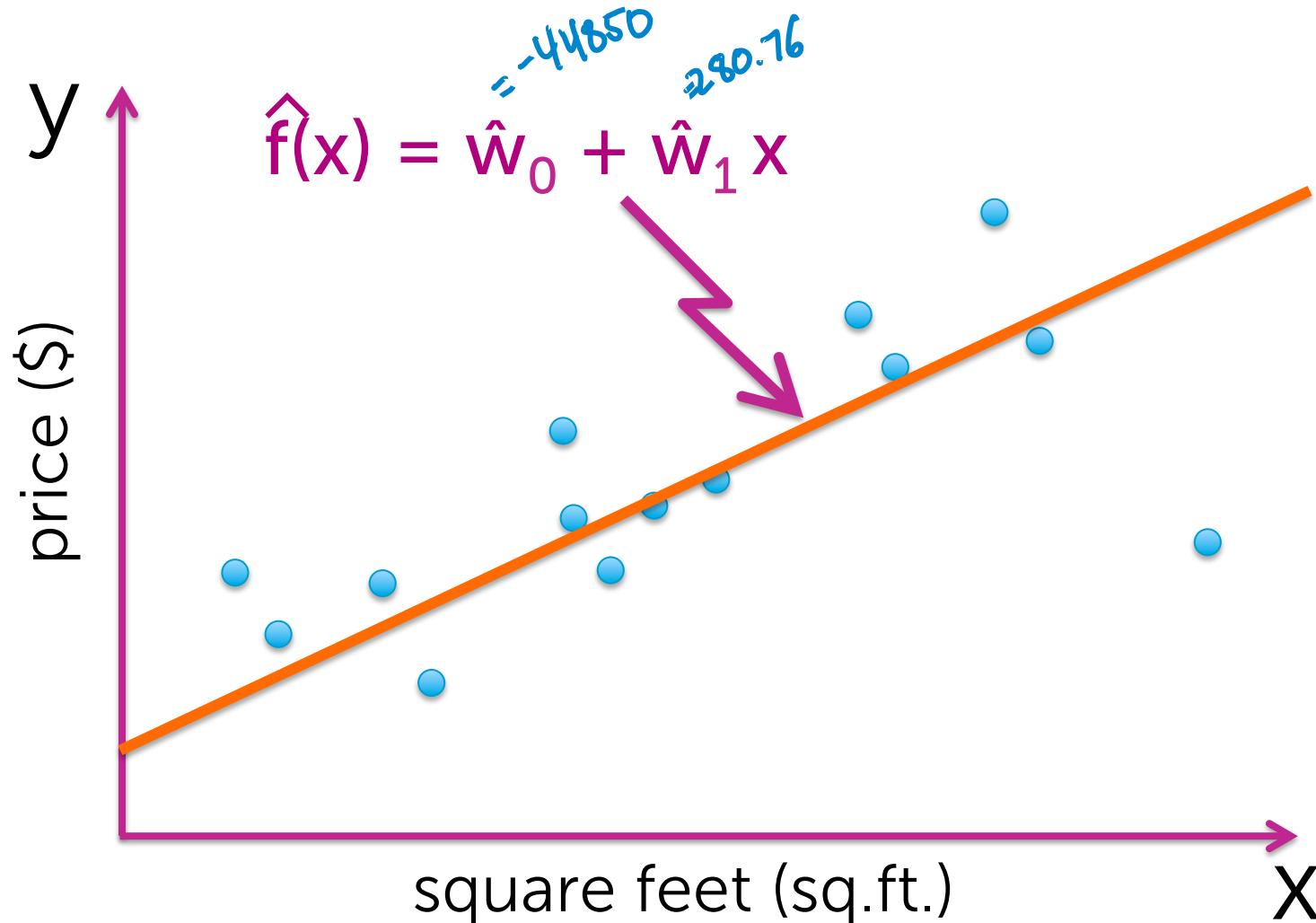
# Find “best” line



$$RSS(w_0, w_1) = \sum_{i=1}^N (y_i - [w_0 + w_1 x_i])^2$$

# The fitted line: use + interpretation

# Model vs. fitted line



Regression model:

$$y_i = w_0 + w_1 x_i + \epsilon_i$$

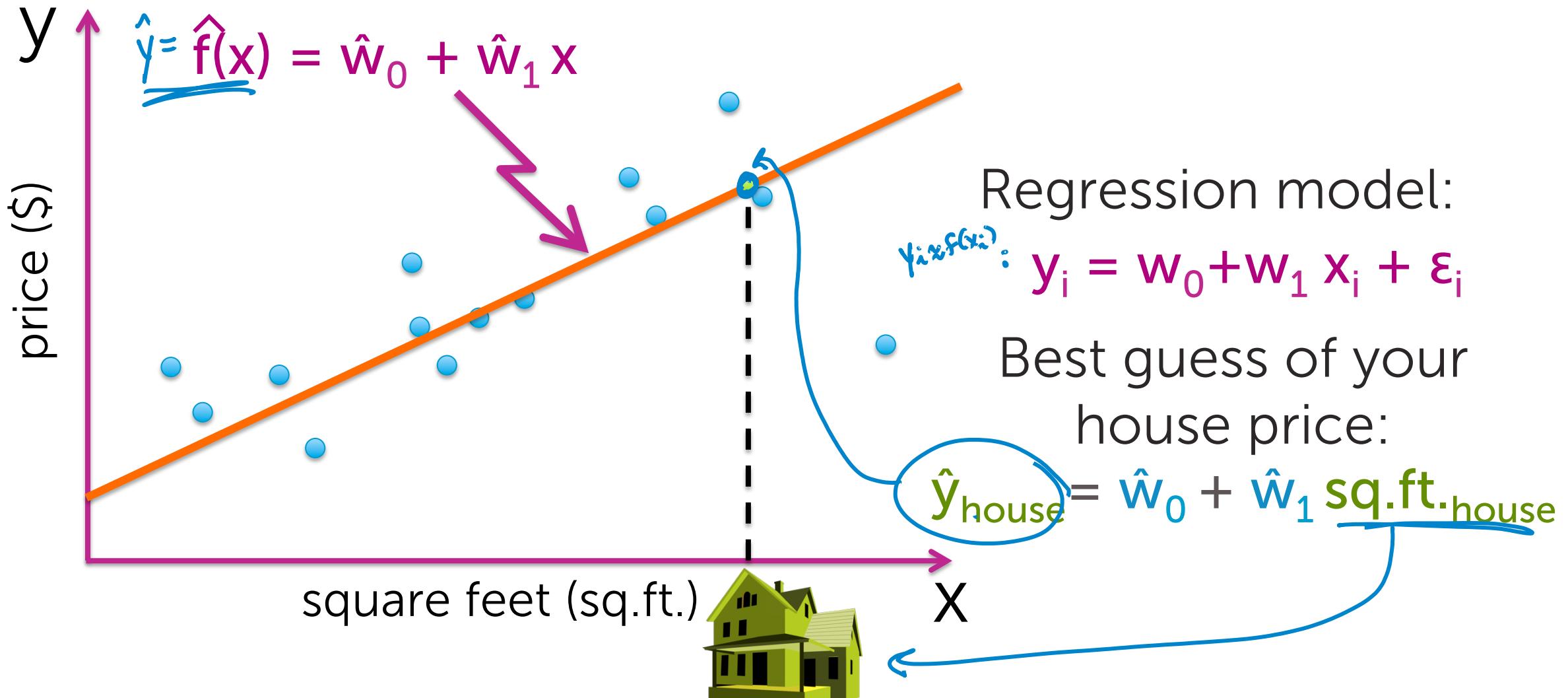
parameters (unknown variables)

Estimated parameters:

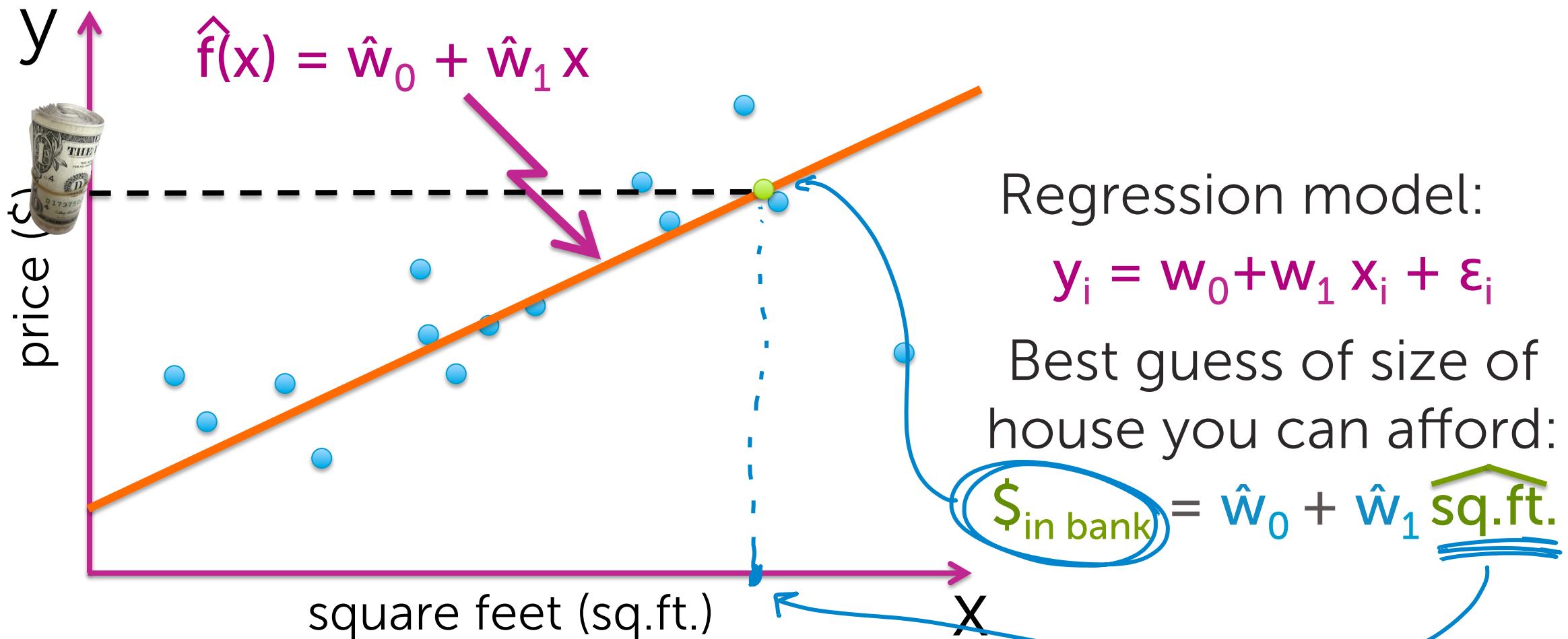
$$\hat{w}_0 = -44950, \hat{w}_1 = 280.76$$

take actual values

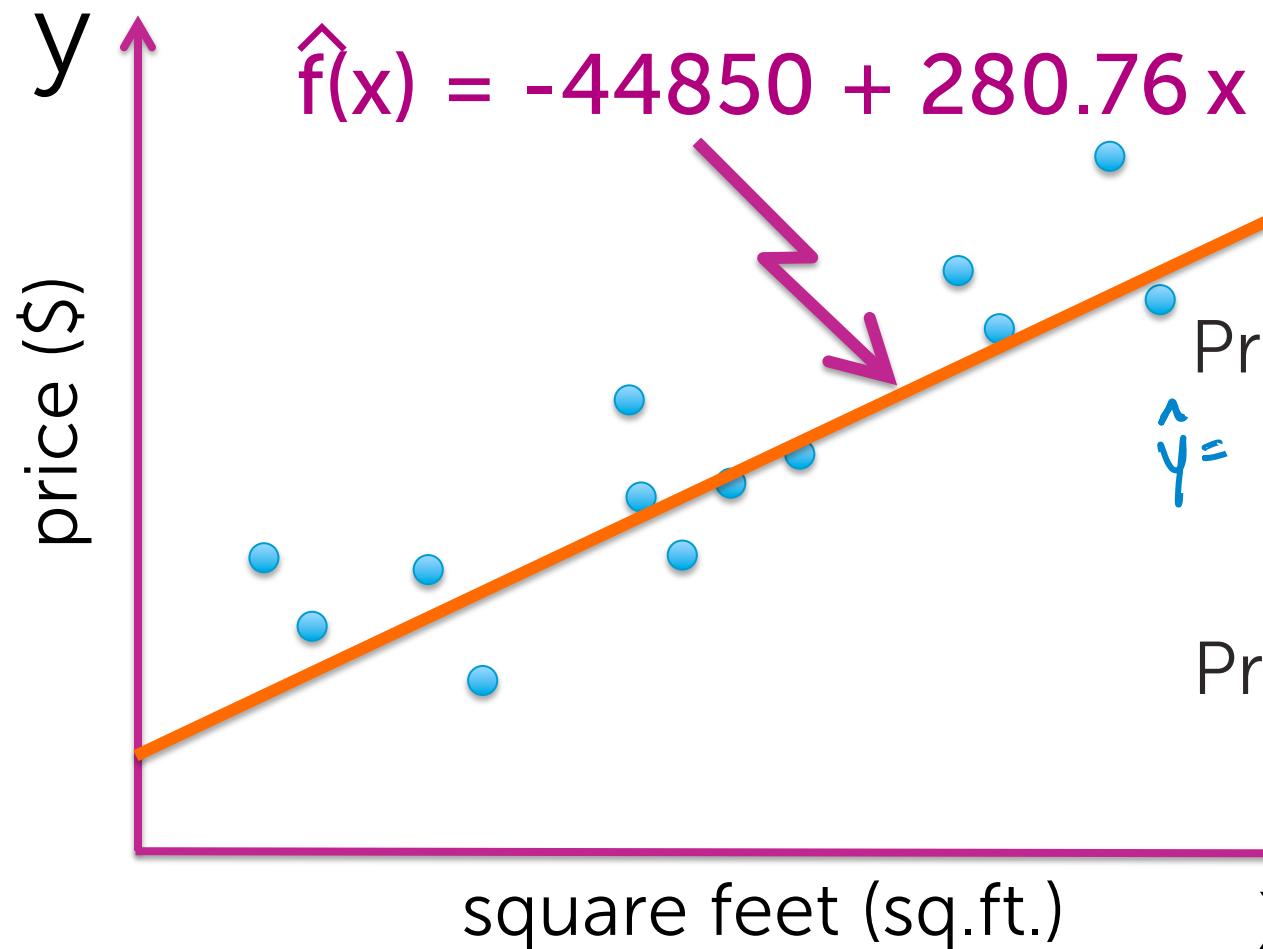
# Seller: Predicting your house price



# Buyer: Predicting size of house



# A concrete example



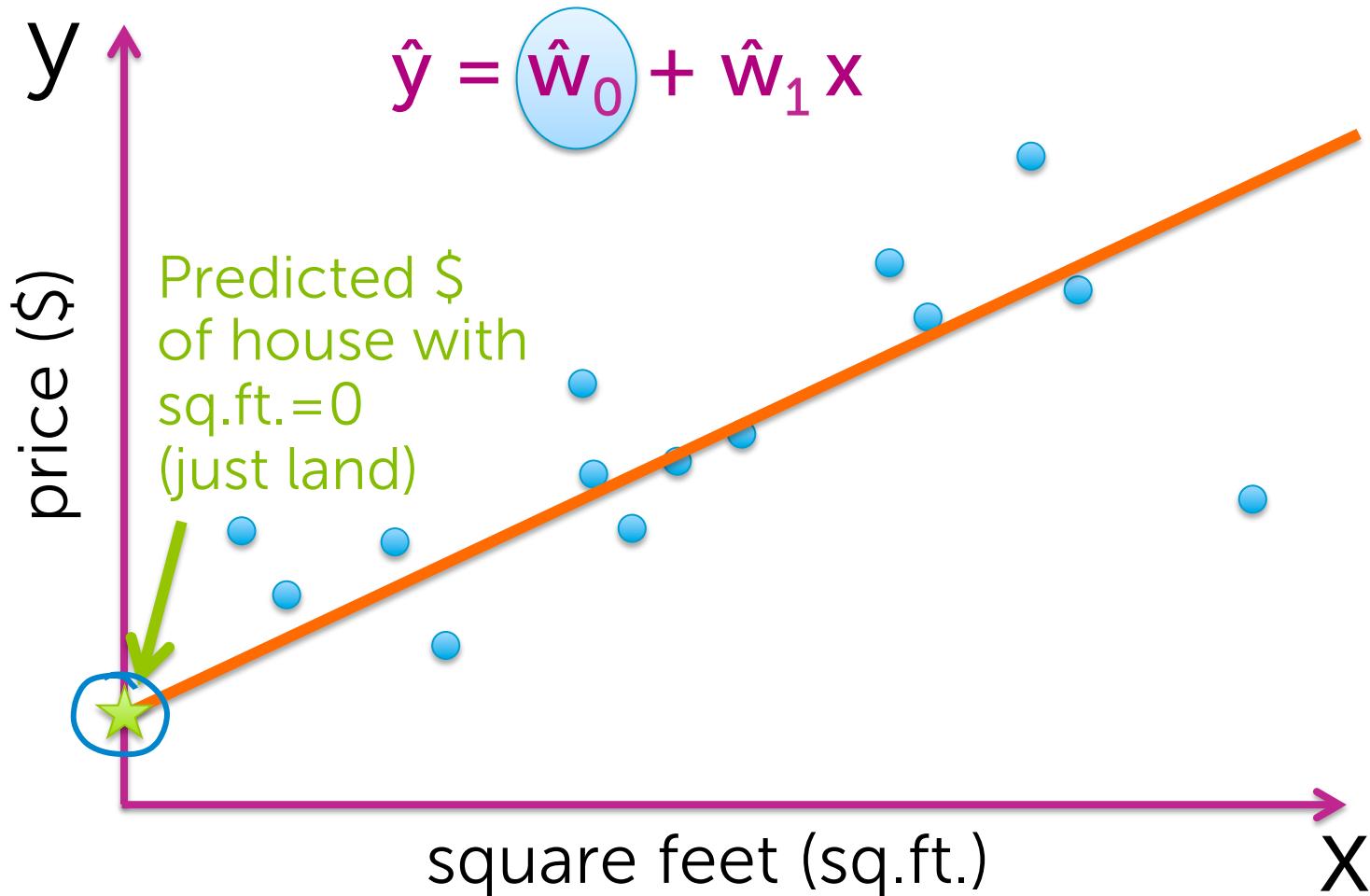
Predict \$ of 2,640 sq.ft. house:

$$\begin{aligned}\hat{y} &= -44850 + 280.76 * \underline{2,640} \\ &= \$696,356\end{aligned}$$

Predict sq.ft. of \$859,000 sale:

$$\begin{aligned}(859000 + 44850) / 280.76 \\ &= 3,219 \text{ sq.ft.}\end{aligned}$$

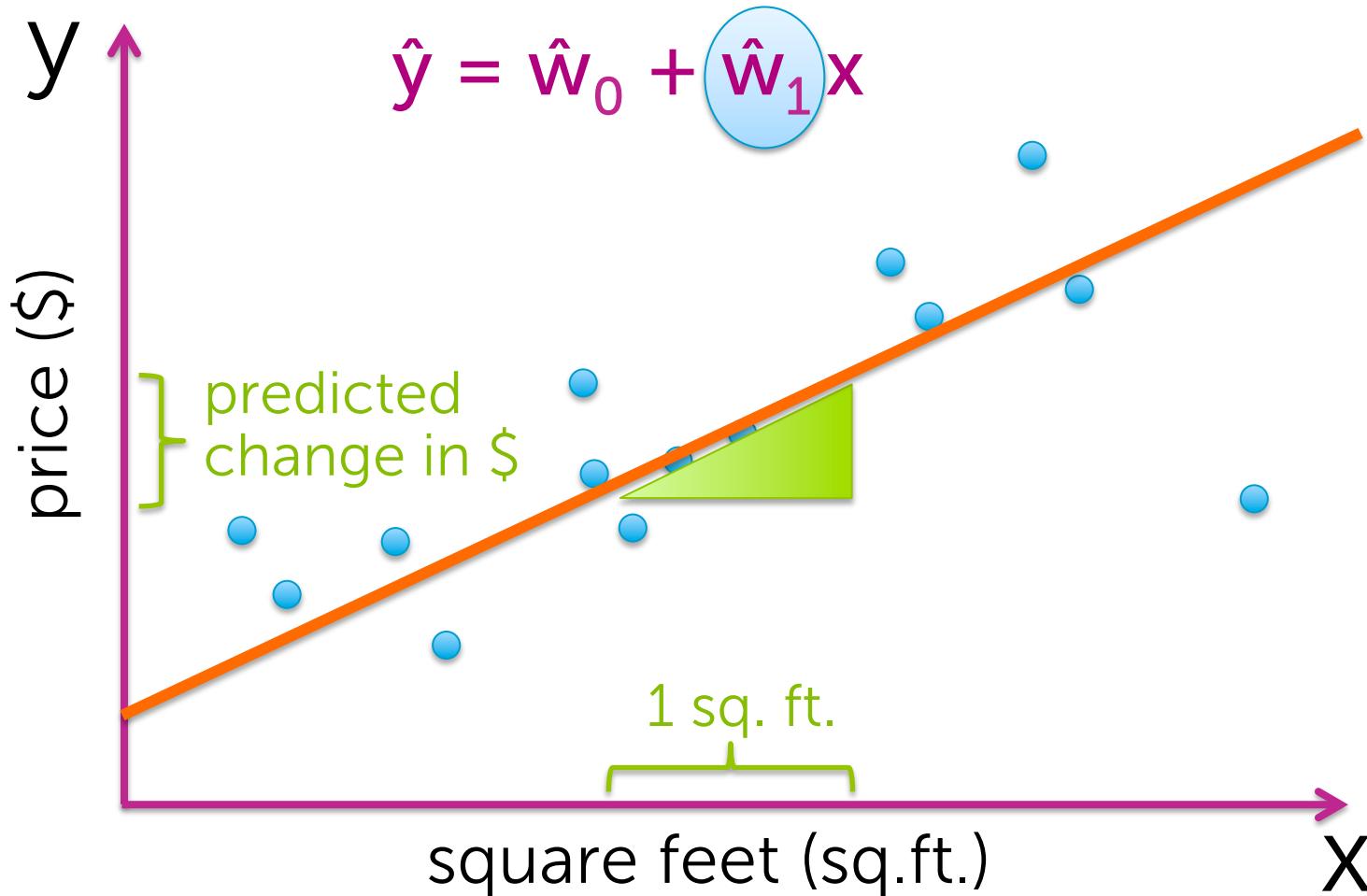
# Interpreting the coefficients



$$\hat{y} = \hat{w}_0 \text{ when } x=0$$

not very meaningful

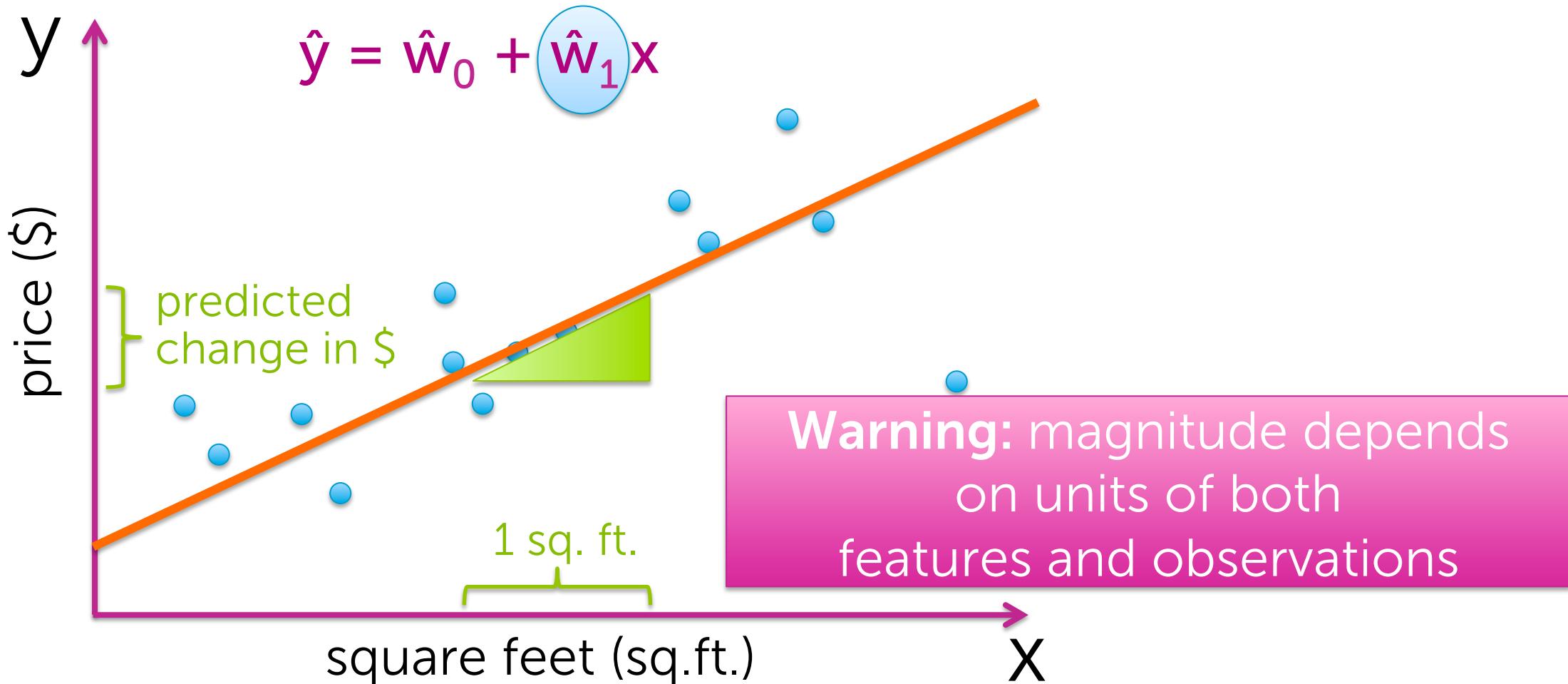
# Interpreting the coefficients



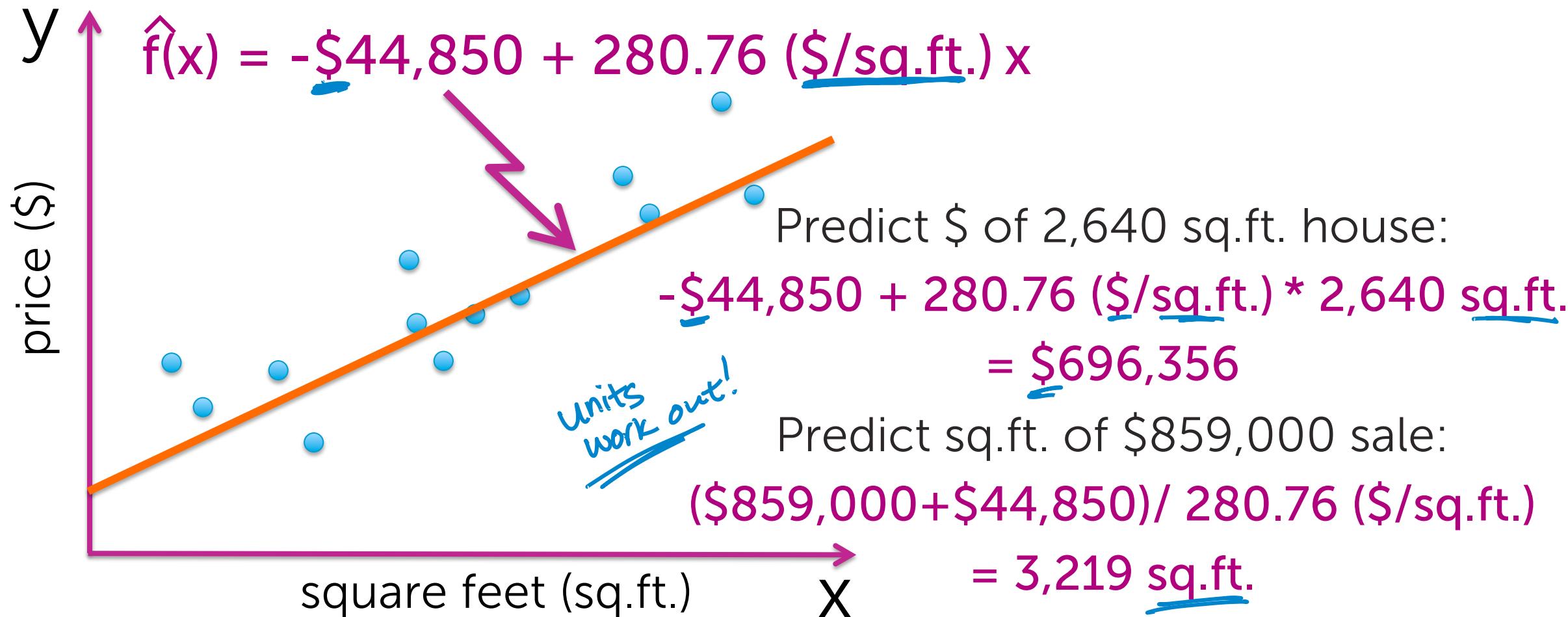
$$\begin{aligned}\hat{y}_{1001 \text{ sq.ft.}} - \hat{y}_{1000 \text{ sq.ft.}} &= \hat{w}_0 + \hat{w}_1 \cdot 1001 \text{ sq.ft.} \\ &\quad - (\hat{w}_0 + \hat{w}_1 \cdot 1000 \text{ sq.ft.}) \\ &= \hat{w}_1\end{aligned}$$

predicted change in the output per unit change in input

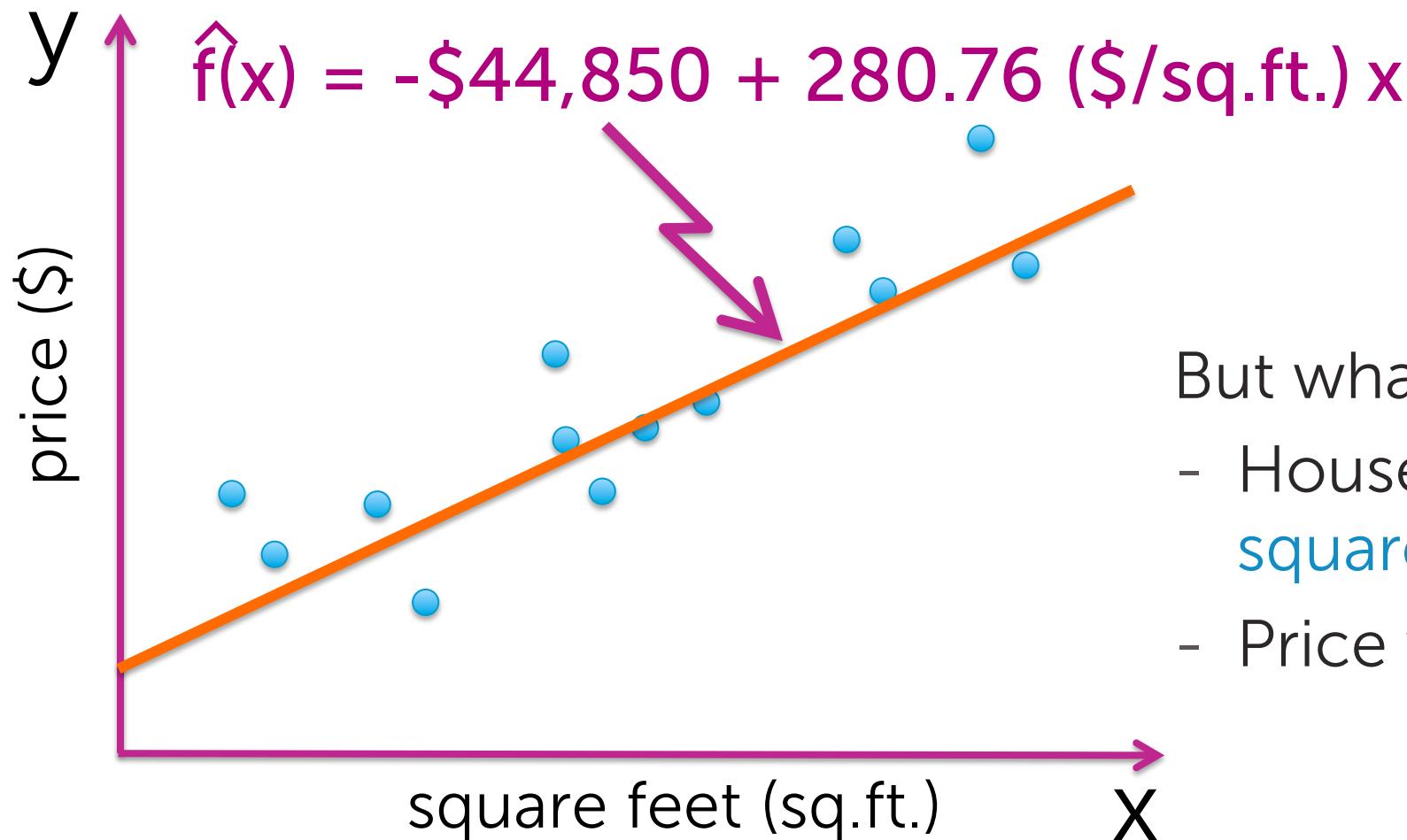
# Interpreting the coefficients



# A concrete example



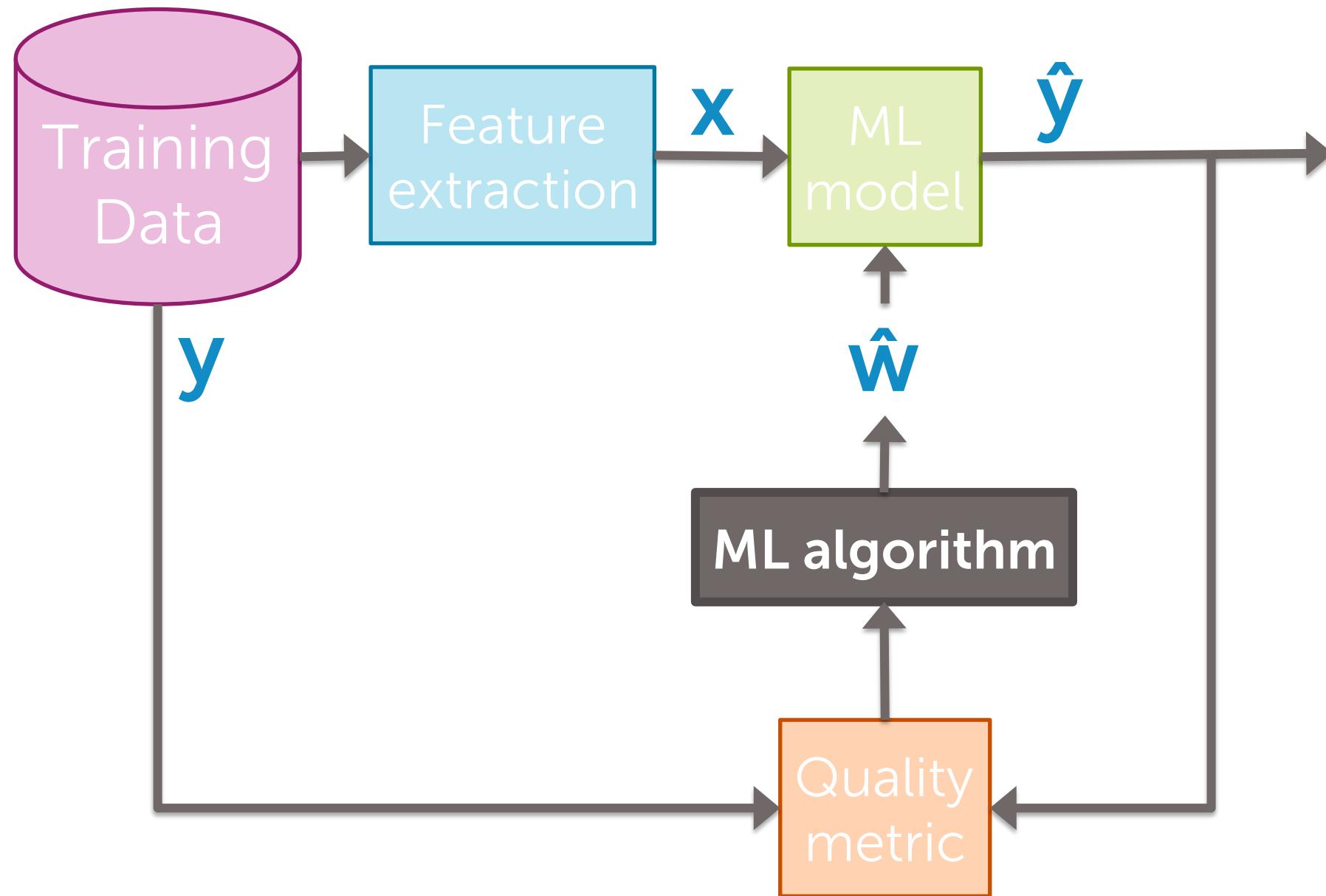
# A concrete example



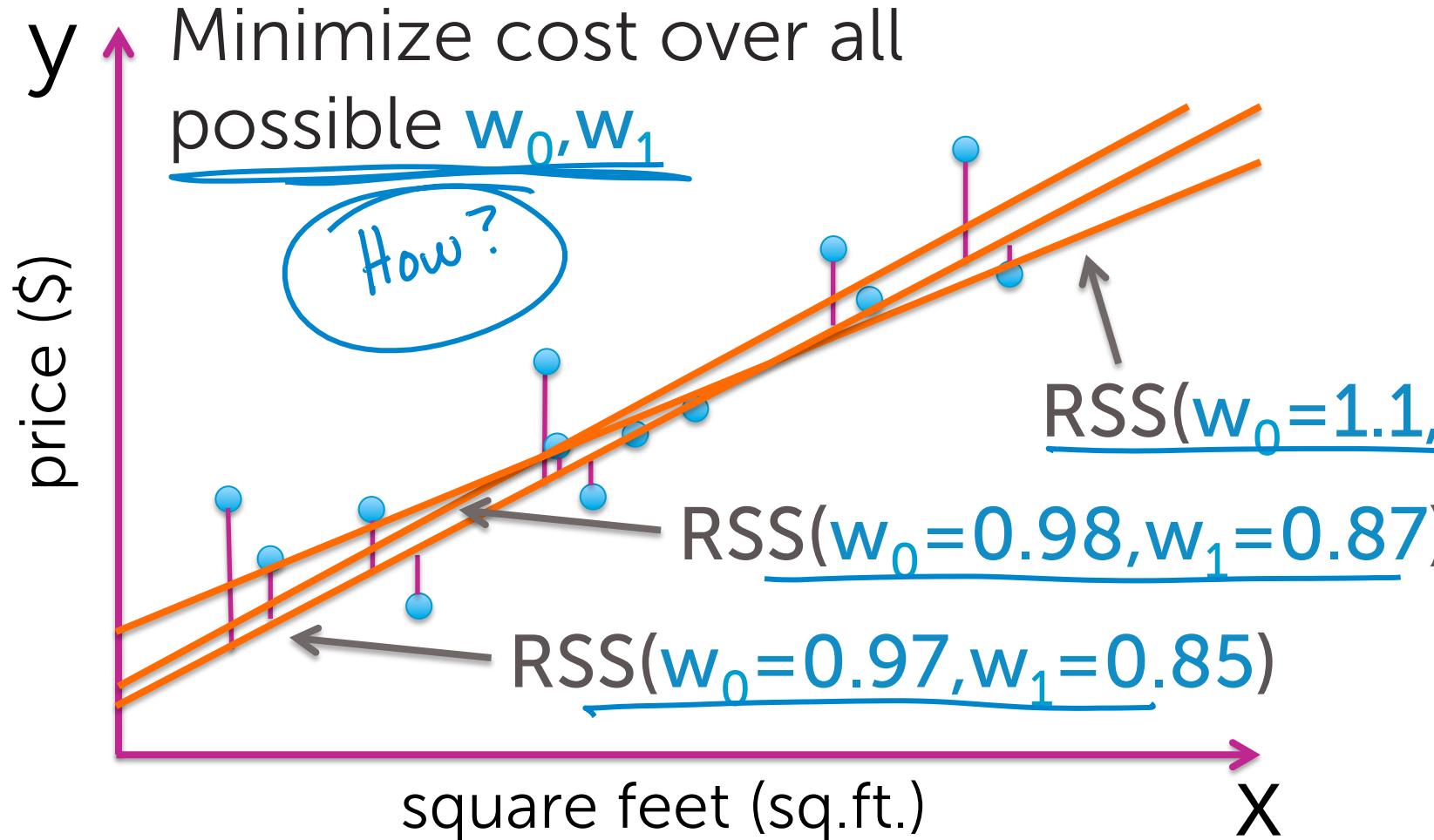
But what if:

- House was measured in square meters?
- Price was measured in RMB?

# Algorithms for fitting the model



# Find “best” line

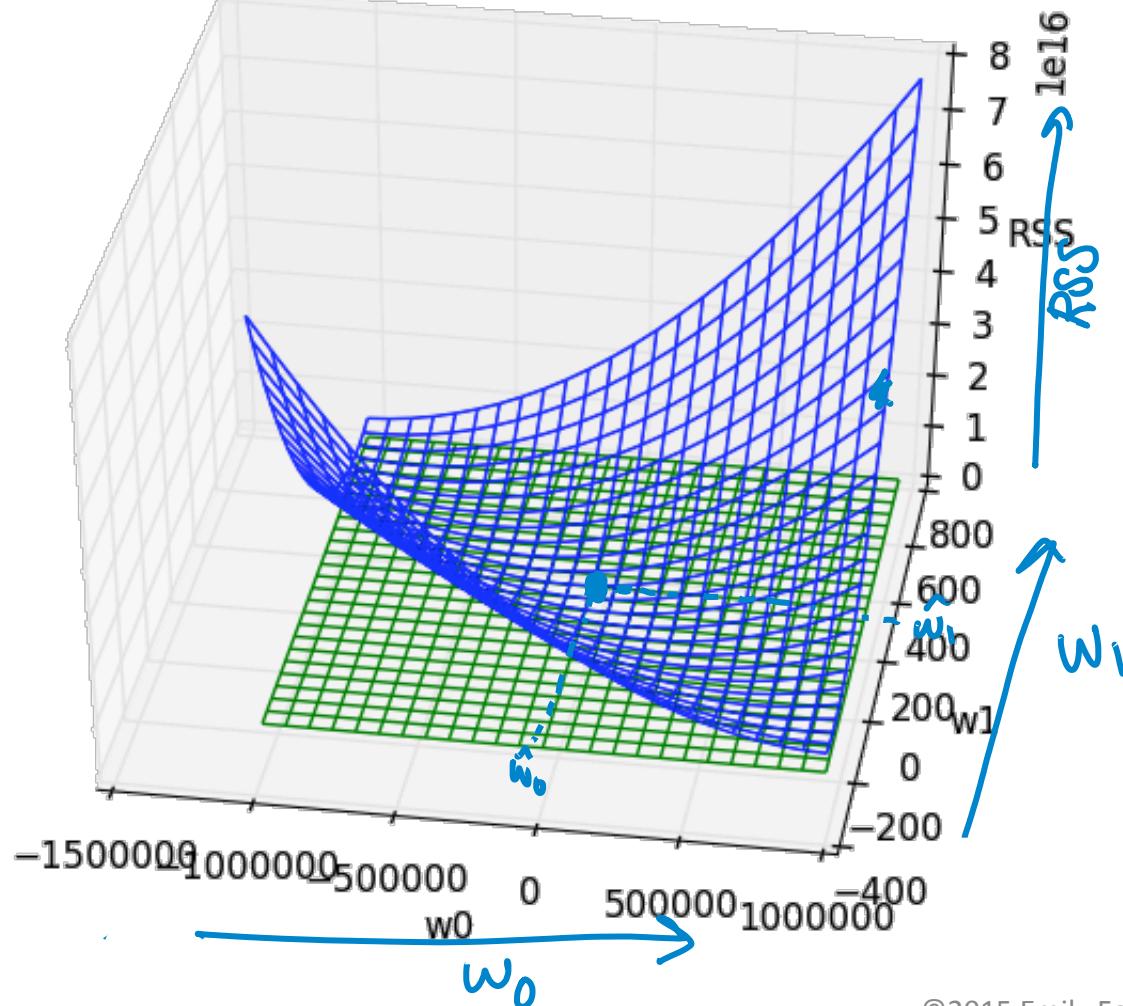


Recall:

$$\text{RSS}(w_0, w_1) = \sum_{i=1}^N (y_i - [w_0 + w_1 x_i])^2$$

# Minimizing the cost

3D plot of RSS with tangent plane at minimum



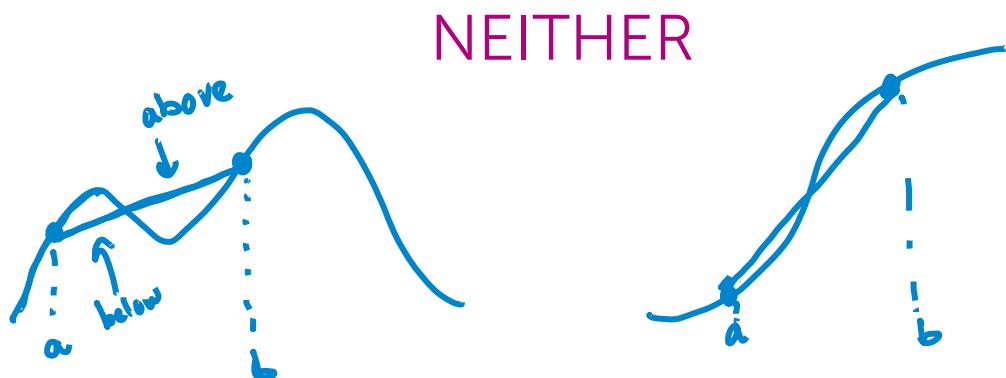
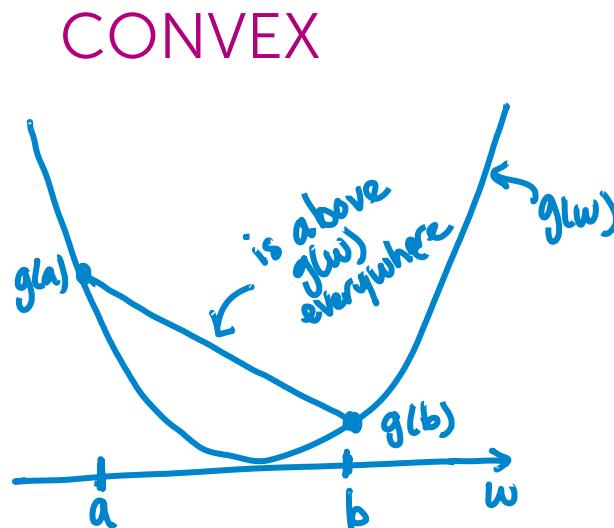
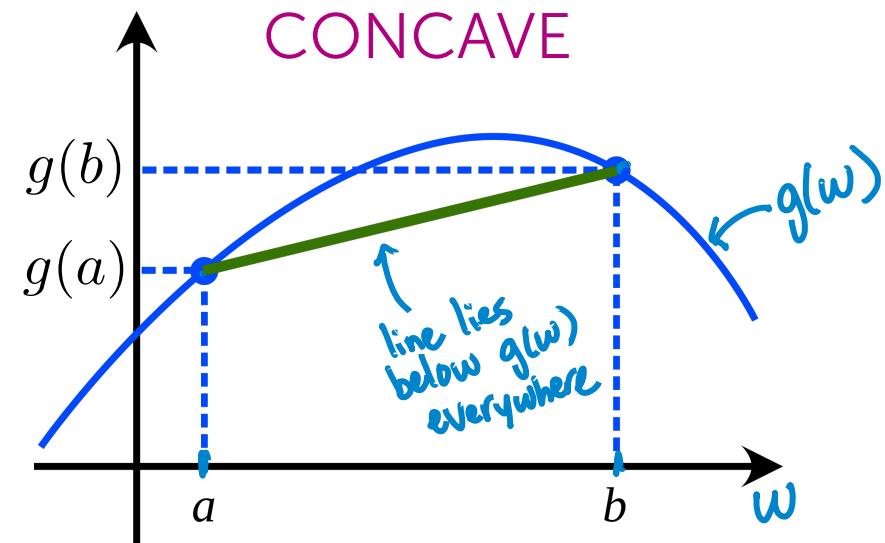
Minimize function  
over all possible  $w_0, w_1$

$$\min_{w_0, w_1} \sum_{i=1}^N (y_i - [w_0 + w_1 x_i])^2$$

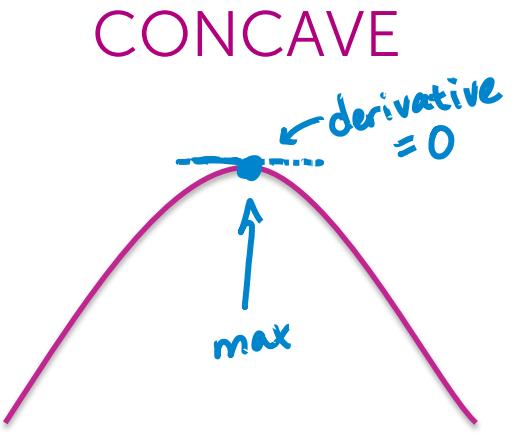
RSS( $w_0, w_1$ ) is a function  
of 2 variables =  $g(w_0, w_1)$

# An aside on optimization

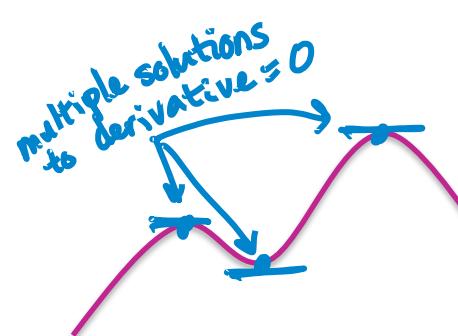
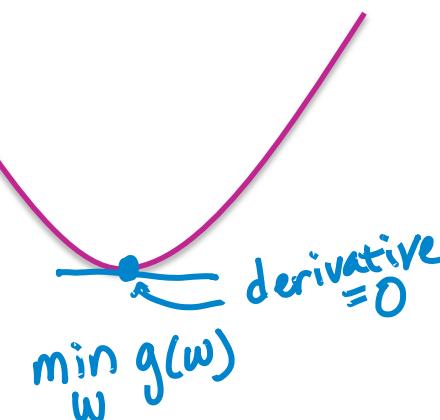
# Convex/concave functions



# Finding the max or min analytically



CONVEX



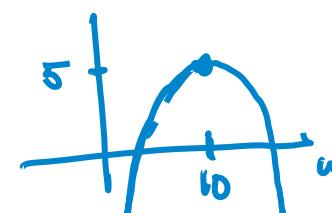
no solution to derivative = 0

Example:

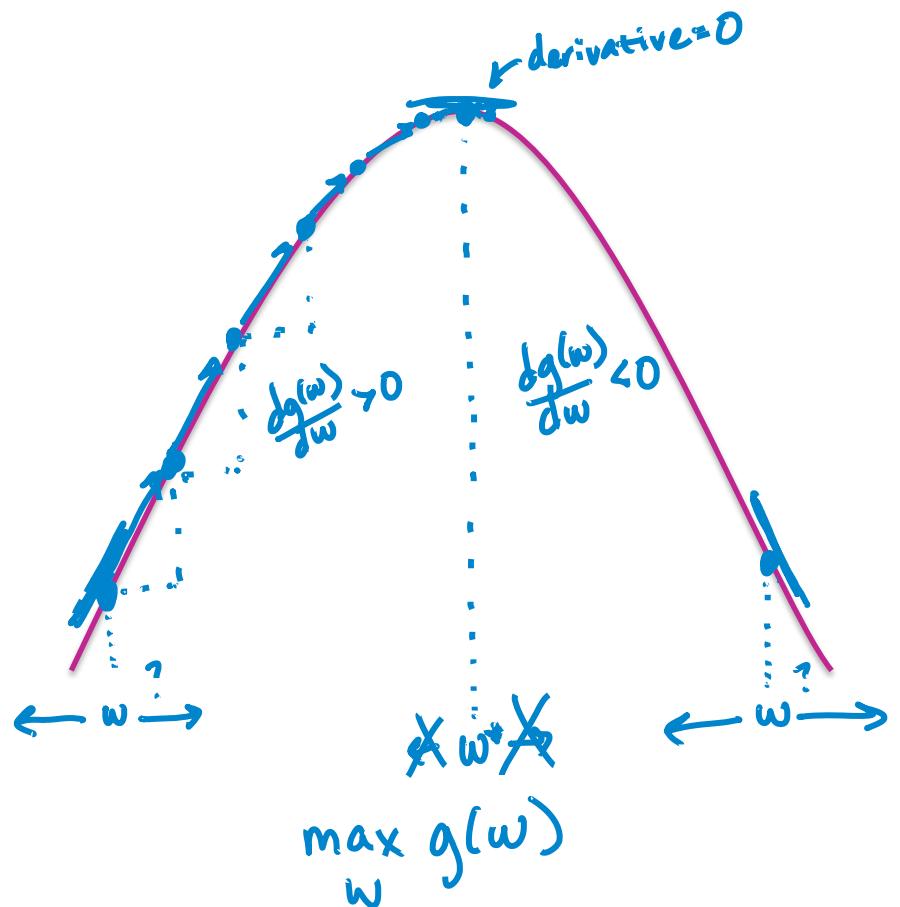
$$g(w) = 5 - (w-10)^2$$

$$\begin{aligned}\frac{dg(w)}{dw} &= 0 - 2(w-10) \cdot 1 \\ &= -2w + 20\end{aligned}$$

$$\begin{aligned}\text{set derivate } &= 0 : \\ -2w + 20 &= 0 \\ w &= 10\end{aligned}$$



# Finding the max via hill climbing



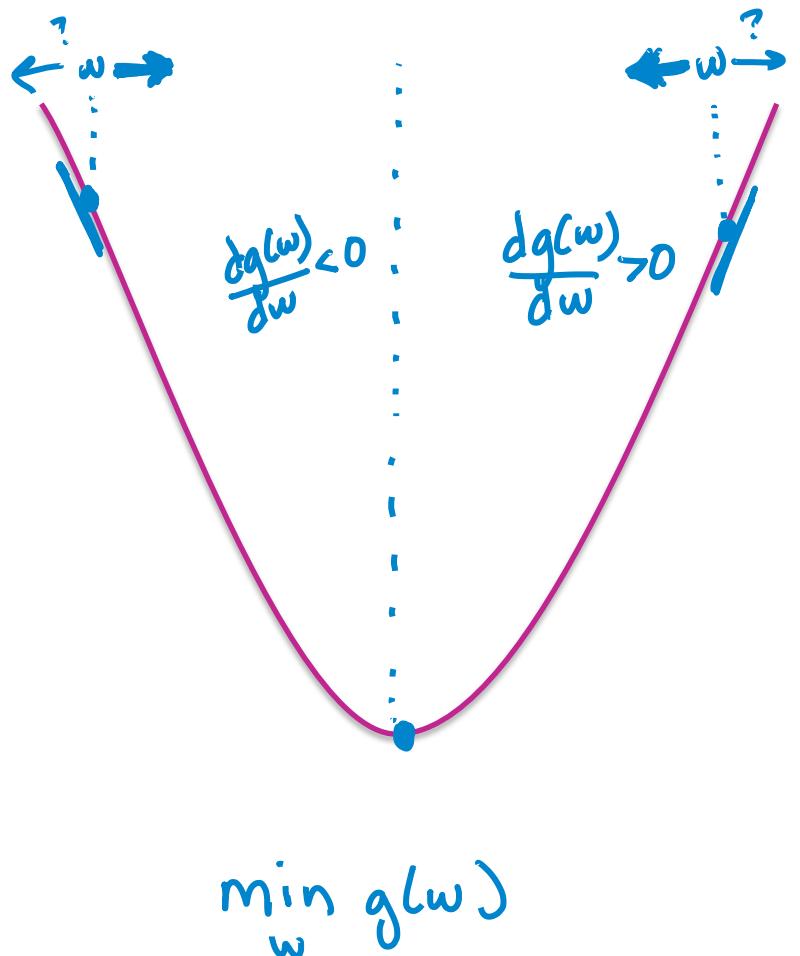
How do we know whether to move  $w$  to right or left?  
(Inc. or dec. the value of  $w$ ?)

while not converged

$$w^{(t+1)} \leftarrow w^{(t)} + \eta \frac{dg(w)}{dw}$$

iteration  $t$       stepsize

# Finding the min via hill descent



when derivative is positive, we want to decrease w  
and when derivative is negative, we want to increase w

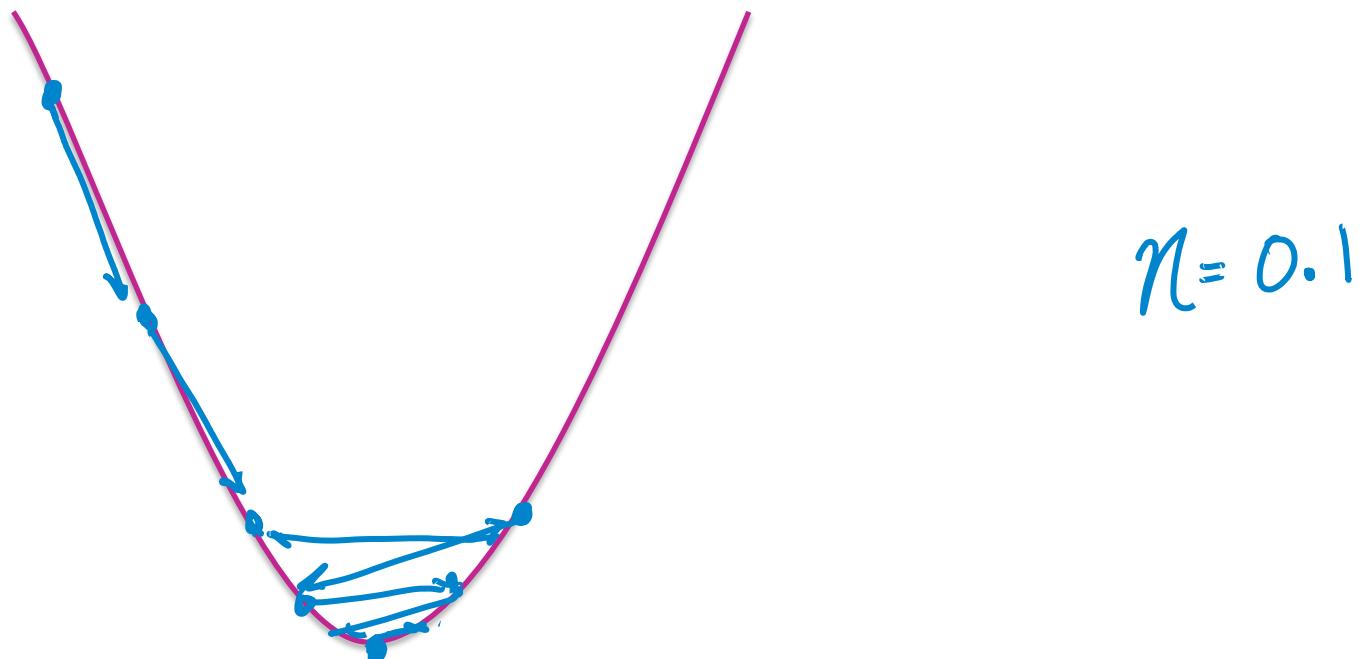
Algorithm:

**while** not converged  
 $w^{(t+1)} \leftarrow w^{(t)} - \eta \frac{dg}{dw}\Big|_{w^{(t)}}$

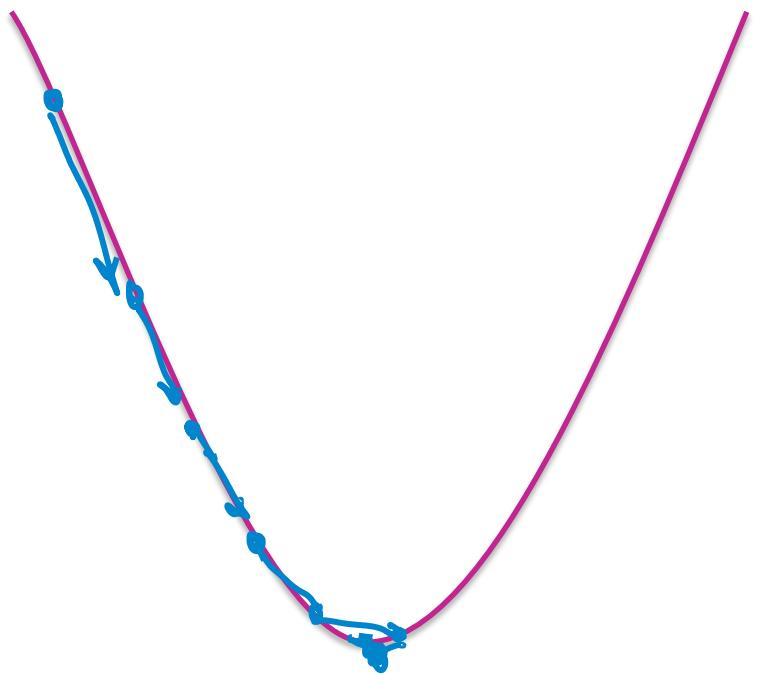
# Choosing the stepsize—

## Fixed stepsize

$\eta$



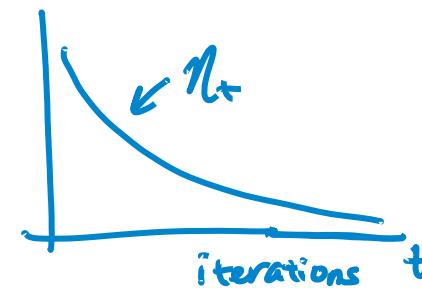
# Choosing the stepsize— Decreasing stepsize or stepsize schedule



Common choices:

$$\eta_t = \frac{\alpha}{t}$$

$$\eta_t = \frac{\alpha}{\sqrt{t}}$$



# Convergence criteria

For convex functions,  
optimum occurs when

$$\frac{dg(w)}{dw} = 0$$

In practice, stop when

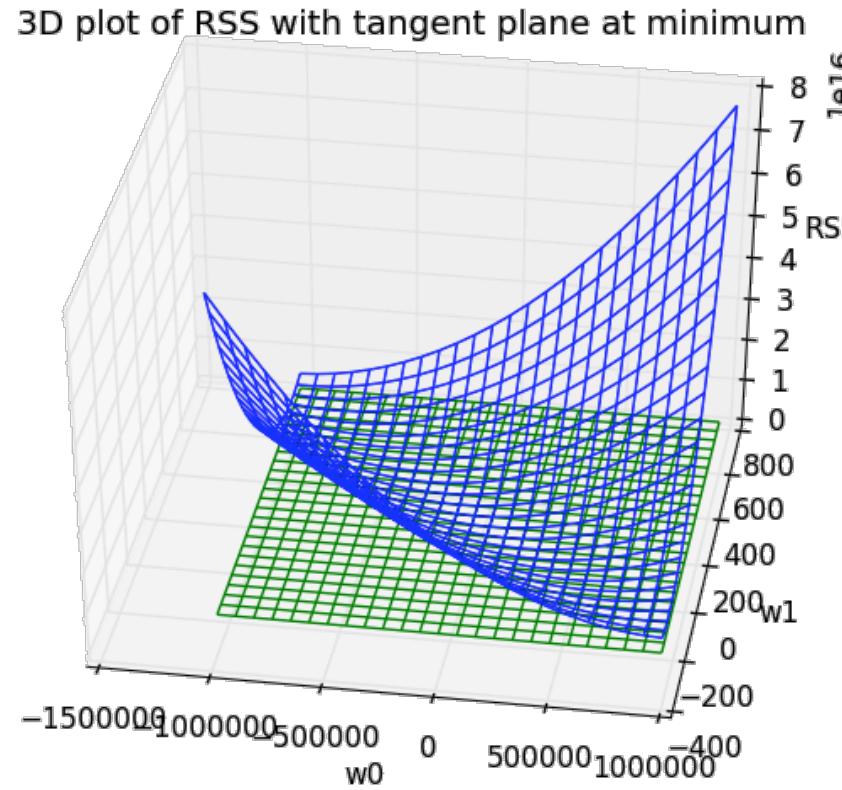
$$\left| \frac{dg(w)}{dw} \right| < \epsilon$$

*threshold  
to be set*

Algorithm:

**while** not converged  
 $w^{(t+1)} \leftarrow w^{(t)} - \eta \left. \frac{dg}{dw} \right|_{w^{(t)}}$

# Moving to multiple dimensions: Gradients



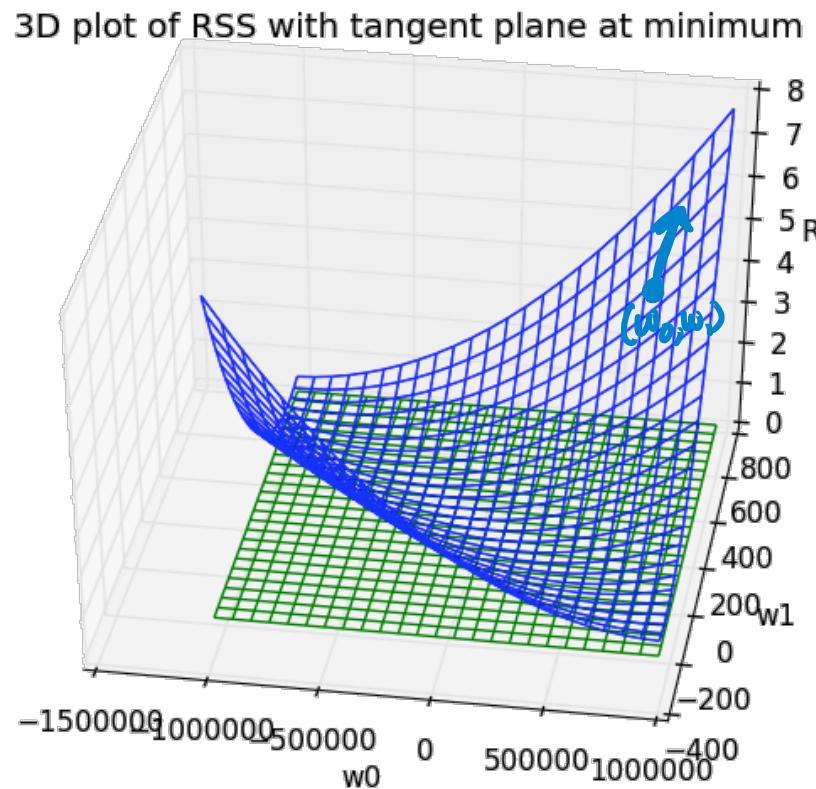
$$\nabla g(\mathbf{w}) = \begin{bmatrix} \frac{\partial g}{\partial w_0} \\ \frac{\partial g}{\partial w_1} \\ \vdots \\ \frac{\partial g}{\partial w_p} \end{bmatrix}$$

gradient     $\uparrow$      $[w_0, w_1, \dots, w_p]$

( $p+1$ ) -dimensional vector

partial derivative  
is like a derivate  
with respect to  $w_i$ ,  
treating all other  
variables as constants

# Gradient example



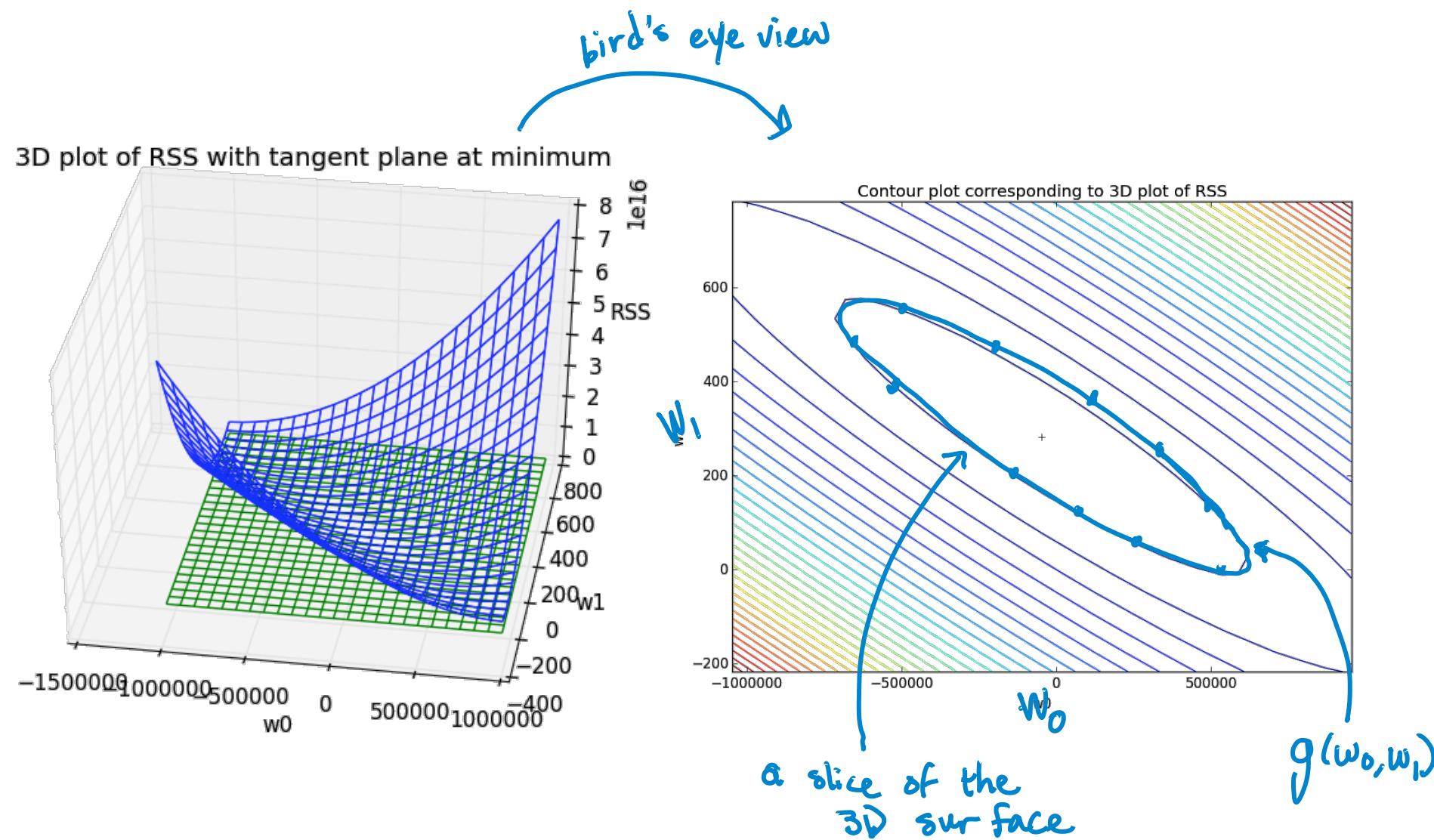
$$g(\mathbf{w}) = 5w_0 + 10w_0w_1 + 2w_1^2$$

$$\frac{\partial g}{\partial w_0} = 5 + 10w_1$$

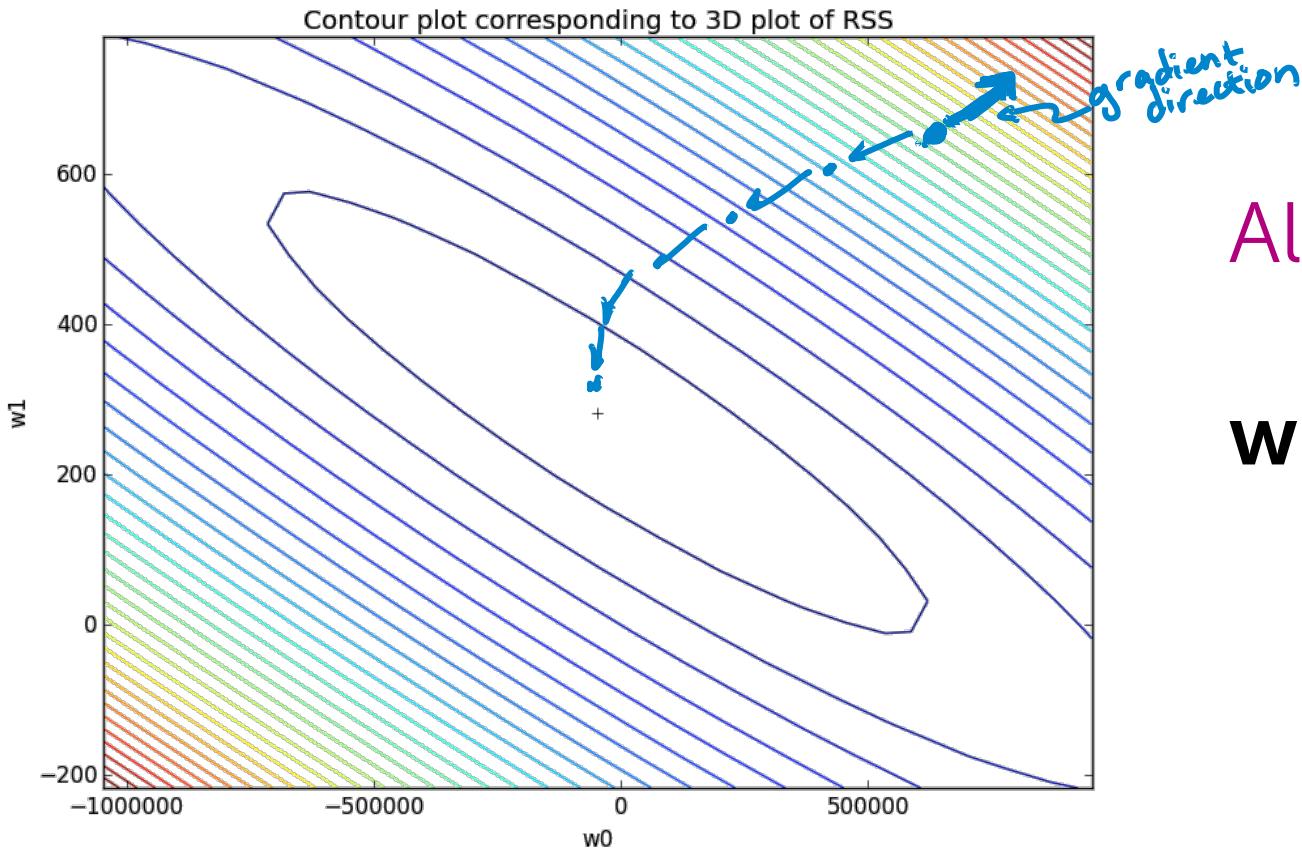
$$\frac{\partial g}{\partial w_1} = 10w_0 + 4w_1$$

$$\nabla g(\mathbf{w}) = \begin{bmatrix} 5 + 10w_1 \\ 10w_0 + 4w_1 \end{bmatrix}$$

# Contour plots



# Gradient descent



Algorithm:

**while** not converged  
 $w^{(t+1)} \leftarrow w^{(t)} - \eta \nabla g(w^{(t)})$

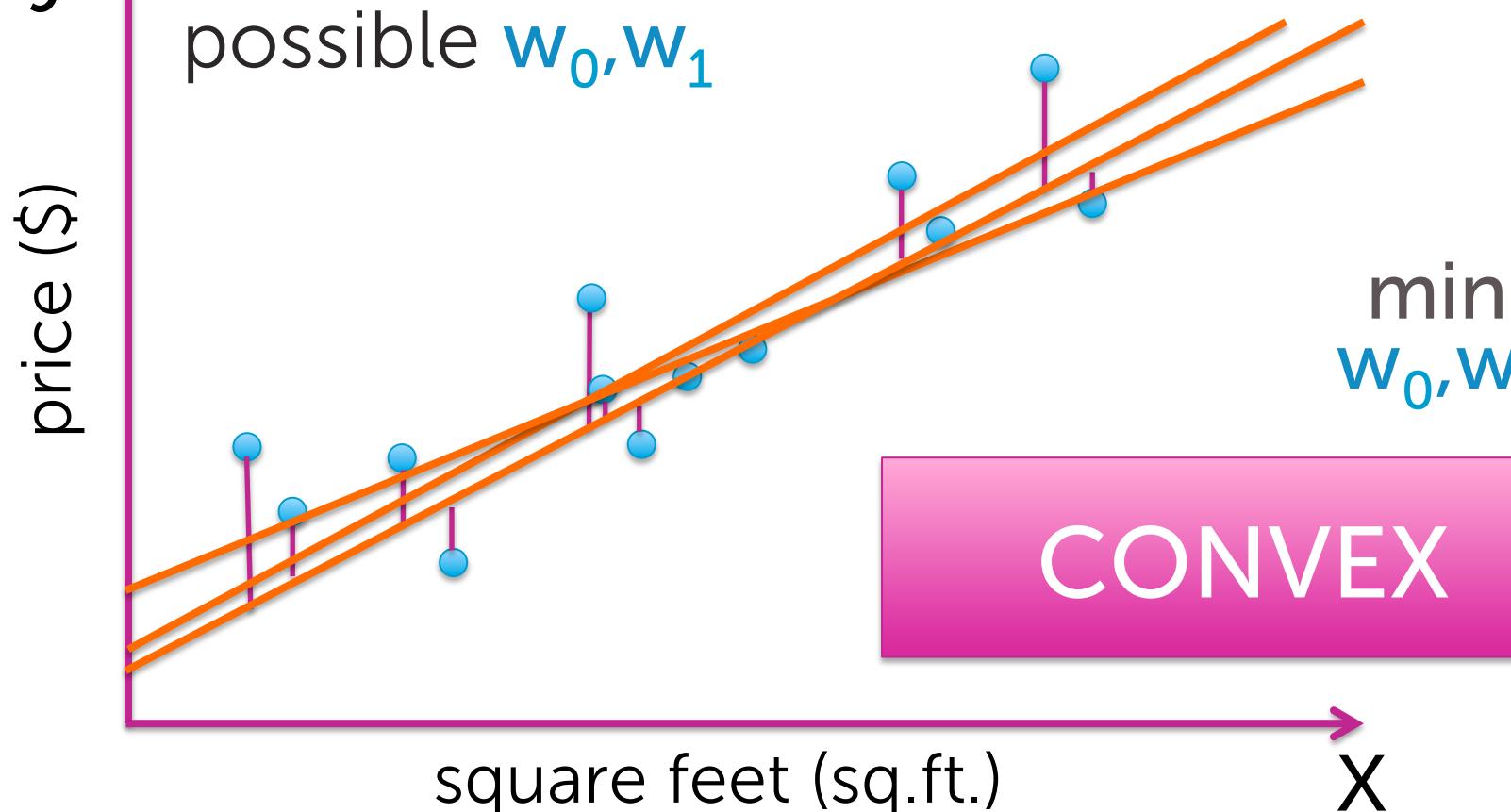
$\begin{bmatrix} \cdot \\ \vdots \\ \cdot \end{bmatrix} \leftarrow \begin{bmatrix} \cdot \\ \vdots \\ \cdot \end{bmatrix} - \eta \begin{bmatrix} \cdot \\ \vdots \\ \cdot \end{bmatrix}$

Convergence:  
 $\|\nabla g(w)\| < \epsilon$

# Finding the least squares line

# Find “best” line

Minimize cost over all possible  $w_0, w_1$



$$\min_{w_0, w_1} \sum_{i=1}^N (y_i - [w_0 + w_1 x_i])^2$$

⇒ solution is unique  
+ gradient descent alg. will converge to minimum

# Compute the gradient

$$\text{RSS}(\mathbf{w}_0, \mathbf{w}_1) = \sum_{i=1}^N (y_i - [\mathbf{w}_0 + \mathbf{w}_1 x_i])^2$$

Aside:

$$\begin{aligned}\frac{d}{dw} \sum_{i=1}^N g_i(w) &= \frac{d}{dw} \underbrace{(g_1(w) + g_2(w) + \dots + g_N(w))}_{\cdot \cdot \cdot} \\ &= \frac{d}{dw} g_1(w) + \frac{d}{dw} g_2(w) + \dots + \frac{d}{dw} g_N(w) \\ &= \sum_{i=1}^N \frac{d}{dw} g_i(w)\end{aligned}$$

In our case

$$g_i(w) = (y_i - [w_0 + w_1 x_i])^2$$
$$\frac{\partial \text{RSS}(w)}{\partial w_0} = \sum_{i=1}^N \frac{\partial}{\partial w_0} (y_i - [w_0 + w_1 x_i])^2$$

same for  $w_1$

# Compute the gradient

$$\text{RSS}(\mathbf{w}_0, \mathbf{w}_1) = \sum_{i=1}^N (y_i - [\mathbf{w}_0 + \mathbf{w}_1 x_i])^2$$

Taking the derivative w.r.t.  $\mathbf{w}_0$

$$\sum_{i=1}^N 2(y_i - [\mathbf{w}_0 + \mathbf{w}_1 x_i])^1 \cdot (-1)$$

$$= -2 \sum_{i=1}^N (y_i - [\mathbf{w}_0 + \mathbf{w}_1 x_i])$$

# Compute the gradient

$$\text{RSS}(\mathbf{w}_0, \mathbf{w}_1) = \sum_{i=1}^N (y_i - [\mathbf{w}_0 + \mathbf{w}_1 x_i])^2$$

Taking the derivative w.r.t.  $\mathbf{w}_1$

$$\sum_{i=1}^N 2(y_i - [\mathbf{w}_0 + \mathbf{w}_1 x_i]) \cdot (-x_i)$$

$$= -2 \sum_{i=1}^N (y_i - [\mathbf{w}_0 + \mathbf{w}_1 x_i]) \underline{x_i}$$

# Compute the gradient

$$\text{RSS}(\mathbf{w}_0, \mathbf{w}_1) = \sum_{i=1}^N (y_i - (\mathbf{w}_0 + \mathbf{w}_1 x_i))^2$$

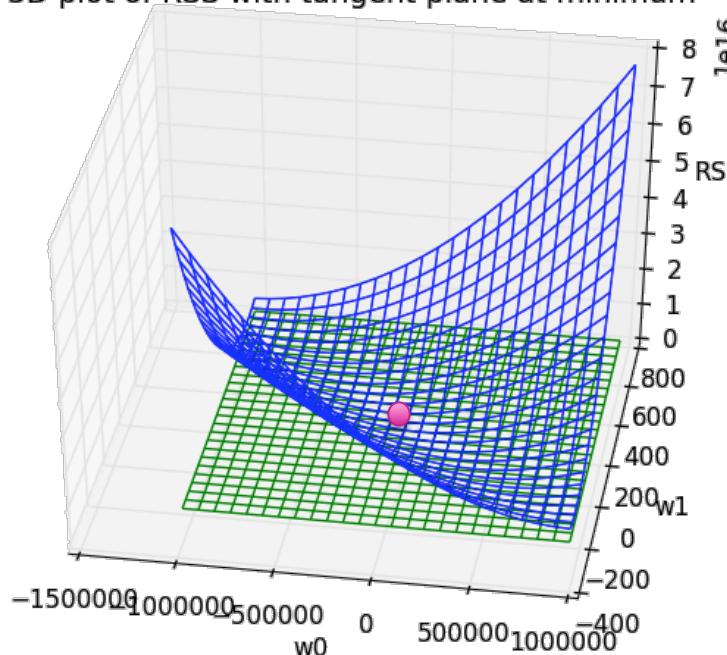
Putting it together:

$$\nabla \text{RSS}(\mathbf{w}_0, \mathbf{w}_1) = \begin{bmatrix} -2 \sum_{i=1}^N [y_i - (\mathbf{w}_0 + \mathbf{w}_1 x_i)] \\ -2 \sum_{i=1}^N [y_i - (\mathbf{w}_0 + \mathbf{w}_1 x_i)] x_i \end{bmatrix}$$

# Approach 1: Set gradient = 0

$$\nabla \text{RSS}(w_0, w_1) = \left[ -2 \sum_{i=1}^N [y_i - (w_0 + w_1 x_i)] \right. \\ \left. -2 \sum_{i=1}^N [y_i - (w_0 + w_1 x_i)] x_i \right]$$

3D plot of RSS with tangent plane at minimum



Plug in:

top term:  $\hat{w}_0 = \frac{\sum y_i}{N} - \hat{w}_1 \frac{\sum x_i}{N}$

average house price  
average sq-ft.  
estimate of the slope

bottom term:  
 $\sum y_i x_i - \hat{w}_0 \sum x_i - \hat{w}_1 \sum x_i^2 = 0$

$\hat{w}_1 = \frac{\sum y_i x_i - \frac{\sum y_i \sum x_i}{N}}{\sum x_i^2 - \frac{\sum x_i \sum x_i}{N}}$

Note:

$$\sum_{i=1}^N y_i$$
$$\sum_{i=1}^N x_i$$
$$\sum_{i=1}^N y_i x_i$$
$$\sum_{i=1}^N x_i^2$$

# Approach 2: Gradient descent

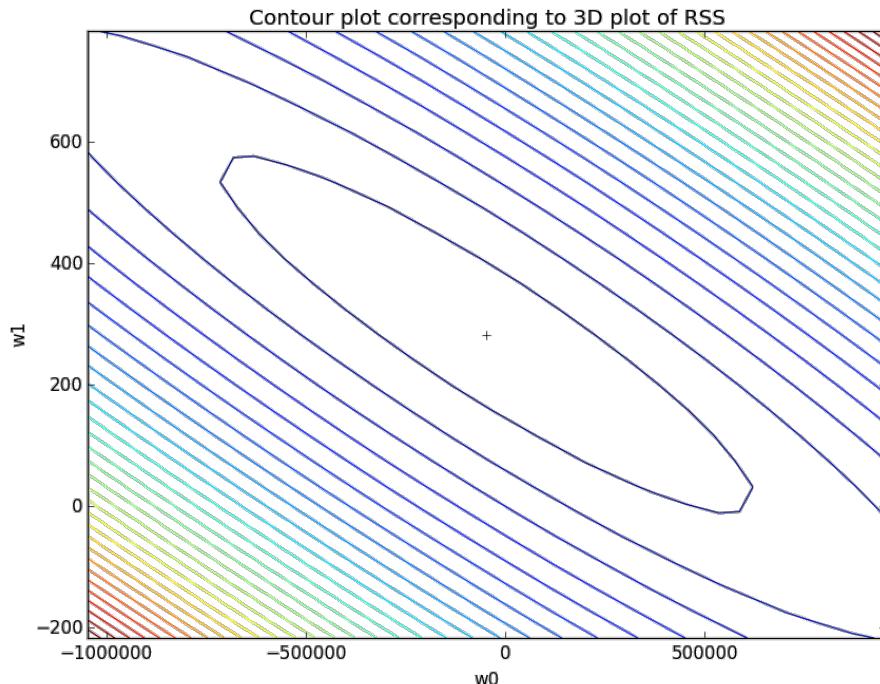
Interpreting the gradient:

$$\nabla \text{RSS}(w_0, w_1) = \begin{bmatrix} -2 \sum_{i=1}^N [y_i - (\underline{w_0 + w_1 x_i})] \\ -2 \sum_{i=1}^N [y_i - (\underline{w_0 + w_1 x_i})]x_i \end{bmatrix} = \begin{bmatrix} -2 \sum_{i=1}^N [y_i - \hat{y}_i(w_0, w_1)] \\ -2 \sum_{i=1}^N [y_i - \hat{y}_i(w_0, w_1)]x_i \end{bmatrix}$$

Annotations for the first term:  
actual house sales observation  
 $y_i$  predicted value  
 $\hat{y}_i(w_0, w_1)$

# Approach 2: Gradient descent

$$\nabla \text{RSS}(w_0, w_1) = \begin{bmatrix} -2 \sum_{i=1}^N [y_i - \hat{y}_i(w_0, w_1)] \\ -2 \sum_{i=1}^N [y_i - \hat{y}_i(w_0, w_1)]x_i \end{bmatrix}$$



while not converged

$$\begin{bmatrix} w_0^{(t+1)} \\ w_1^{(t+1)} \end{bmatrix} \leftarrow \begin{bmatrix} w_0^{(t)} \\ w_1^{(t)} \end{bmatrix} + 2\eta \begin{bmatrix} -2 \cdot (-\nabla) \\ \sum_{i=1}^N [y_i - \hat{y}_i(w_0^{(t)}, w_1^{(t)})] \\ \sum_{i=1}^N [y_i - \hat{y}_i(w_0^{(t)}, w_1^{(t)})]x_i \end{bmatrix}$$

If overall, underpredicting  $\hat{y}_i$ , then  $\sum [y_i - \hat{y}_i]$  is positive  
→  $w_0$  is going to increase  
similar intuition for  $w_1$ , but multiply by  $x_i$

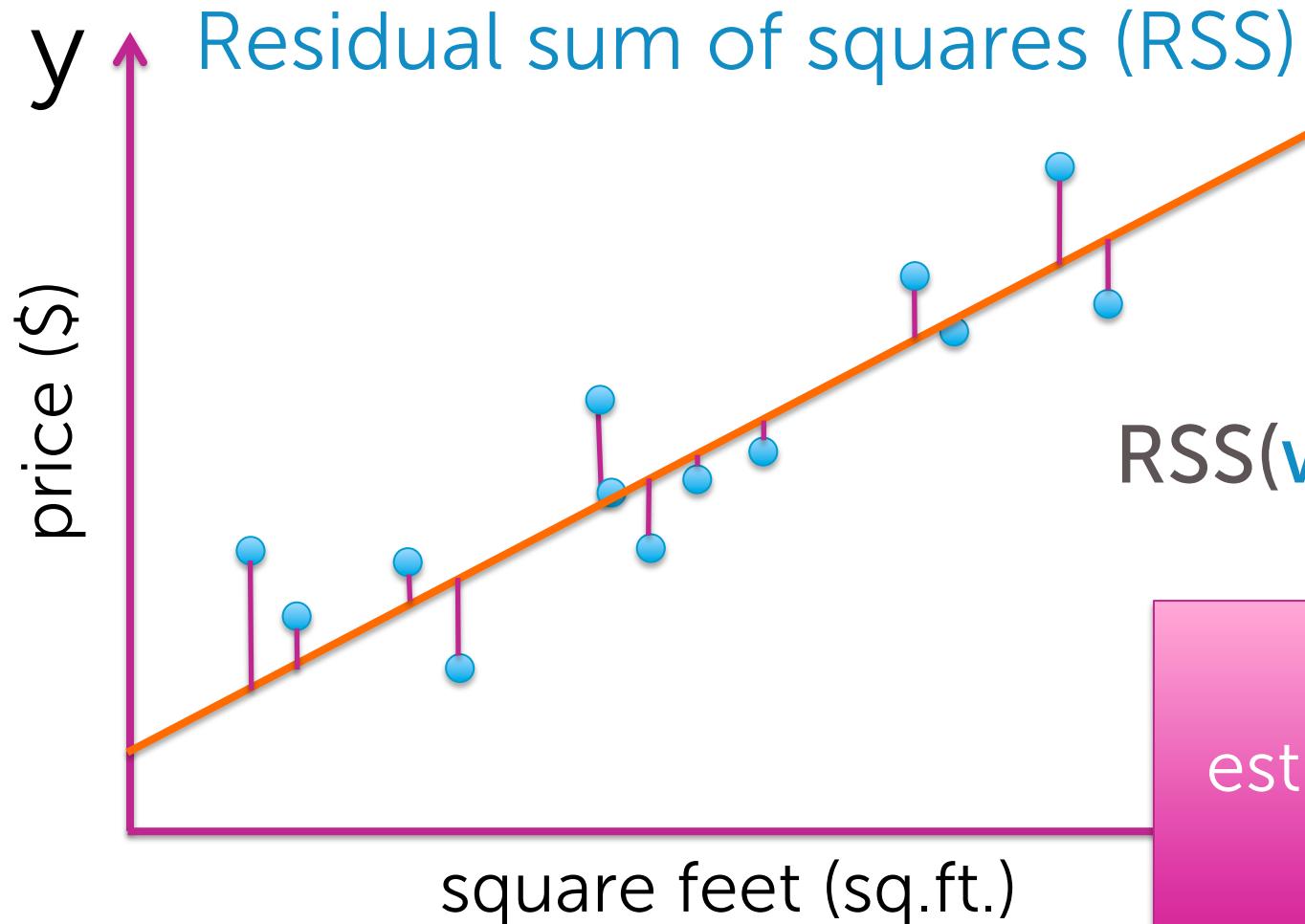
# Comparing the approaches

- For most ML problems, cannot solve **gradient** = 0
- Even if solving **gradient** = 0 is feasible, **gradient descent** can be more efficient
- **Gradient descent** relies on choosing **stepsize** and **convergence** criteria

# Influence of high leverage points

# Asymmetric errors

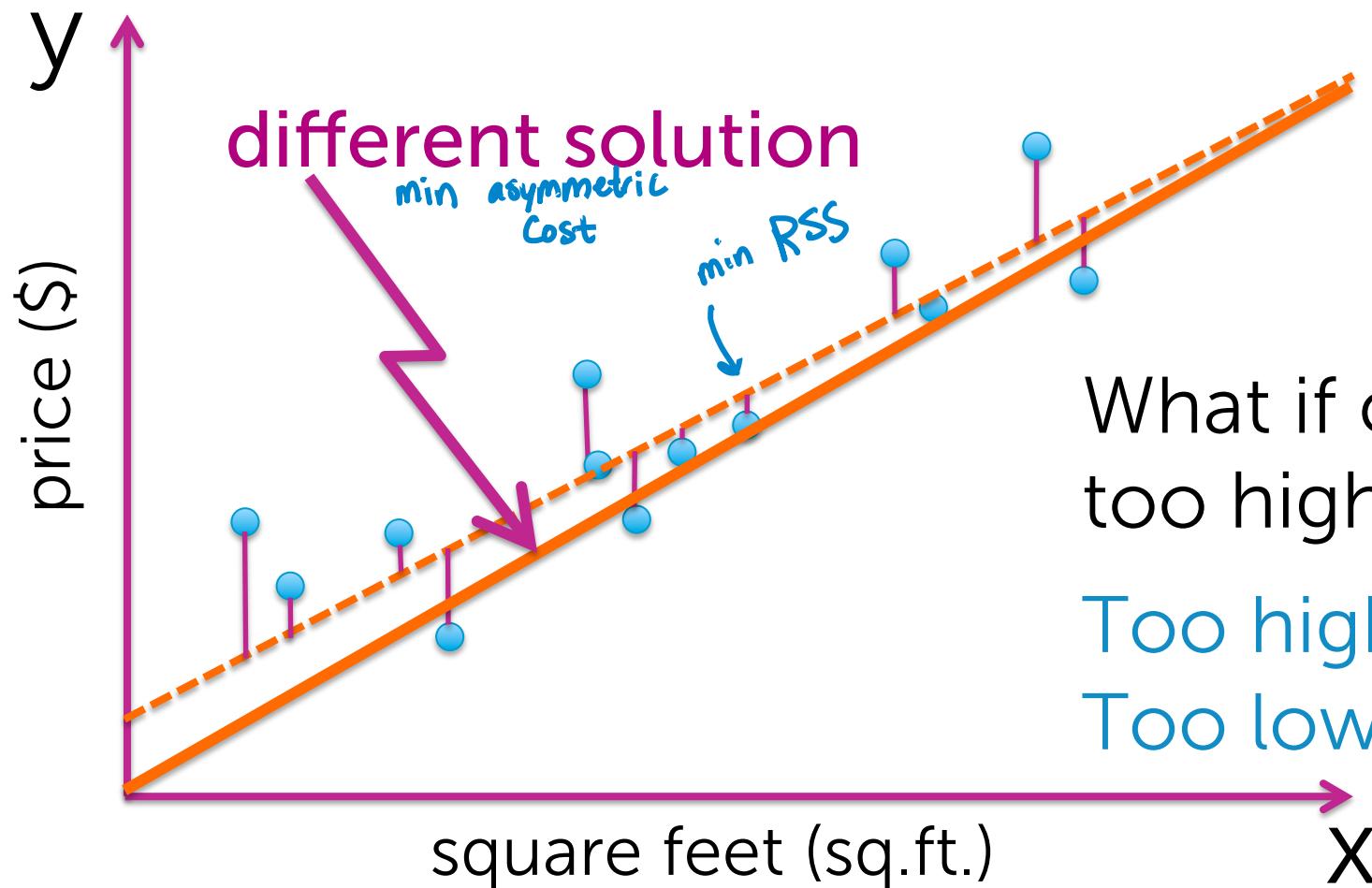
# Symmetric cost functions



$$RSS(w_0, w_1) = \sum_{i=1}^N (y_i - [w_0 + w_1 x_i])^2$$

Assumes cost of over-estimating sales price is same as under-estimating

# Asymmetric cost functions



What if cost of listing house too high has **bigger cost**?

Too high  $\rightarrow$  no offers ( $\$=0$ )

Too low  $\rightarrow$  offers for lower  $\$$

# Summary for simple linear regression

# What you can do now...

- Describe the input (features) and output (real-valued predictions) of a regression model
- Calculate a goodness-of-fit metric (e.g., RSS)
- Estimate model parameters to minimize RSS using gradient descent
- Interpret estimated model parameters
- Exploit the estimated model to form predictions
- Discuss the possible influence of high leverage points
- Describe intuitively how fitted line might change when assuming different goodness-of-fit metrics