Simple Regression:

Linear regression with one input

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*

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

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

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

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

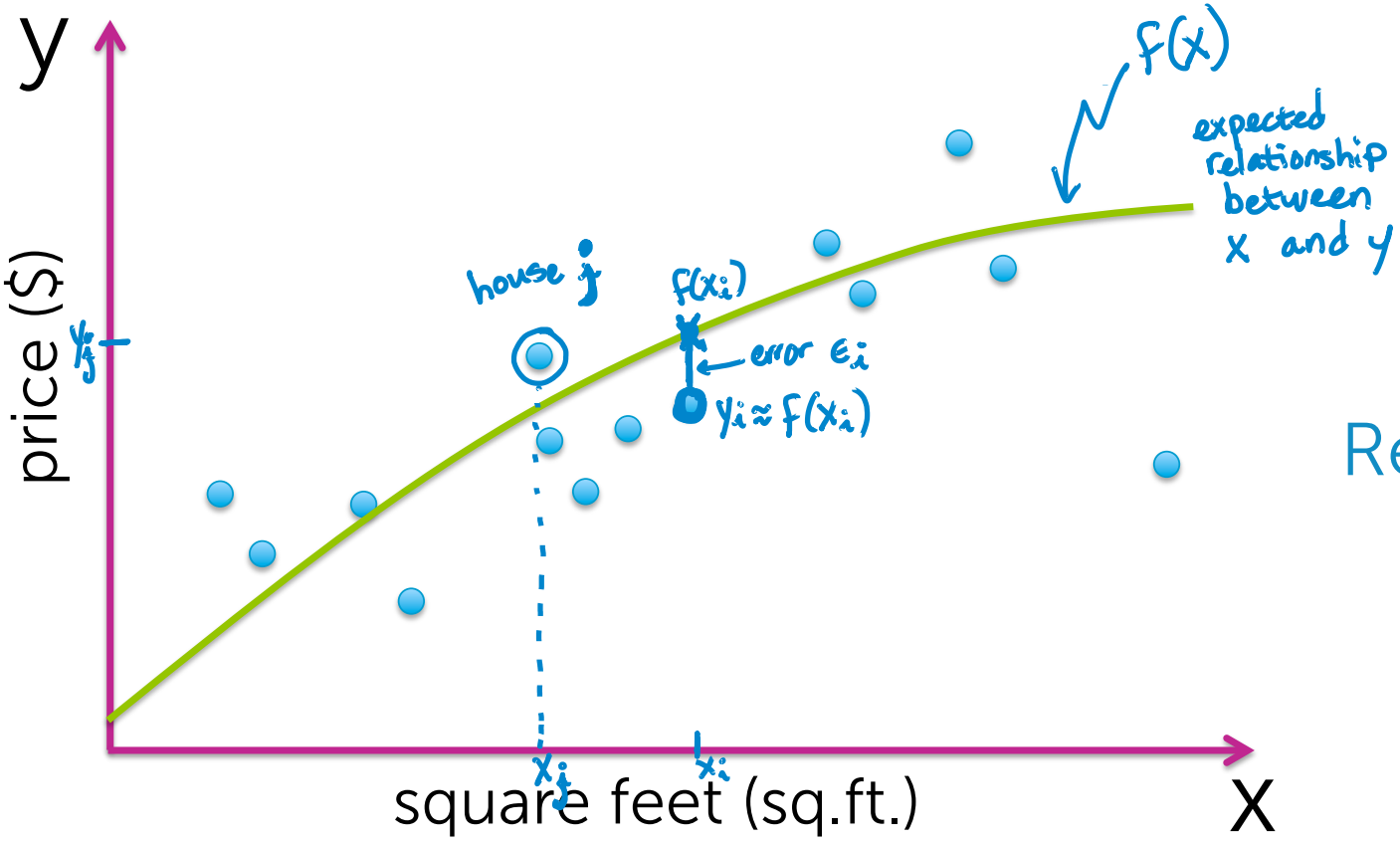
$(x_5 = \text{sq.ft.}, y_5 = \$)$

⋮

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:

$y_i = f(x_i) + \epsilon_i$

$E[\epsilon_i] = 0$ ← equally likely that error is + or −

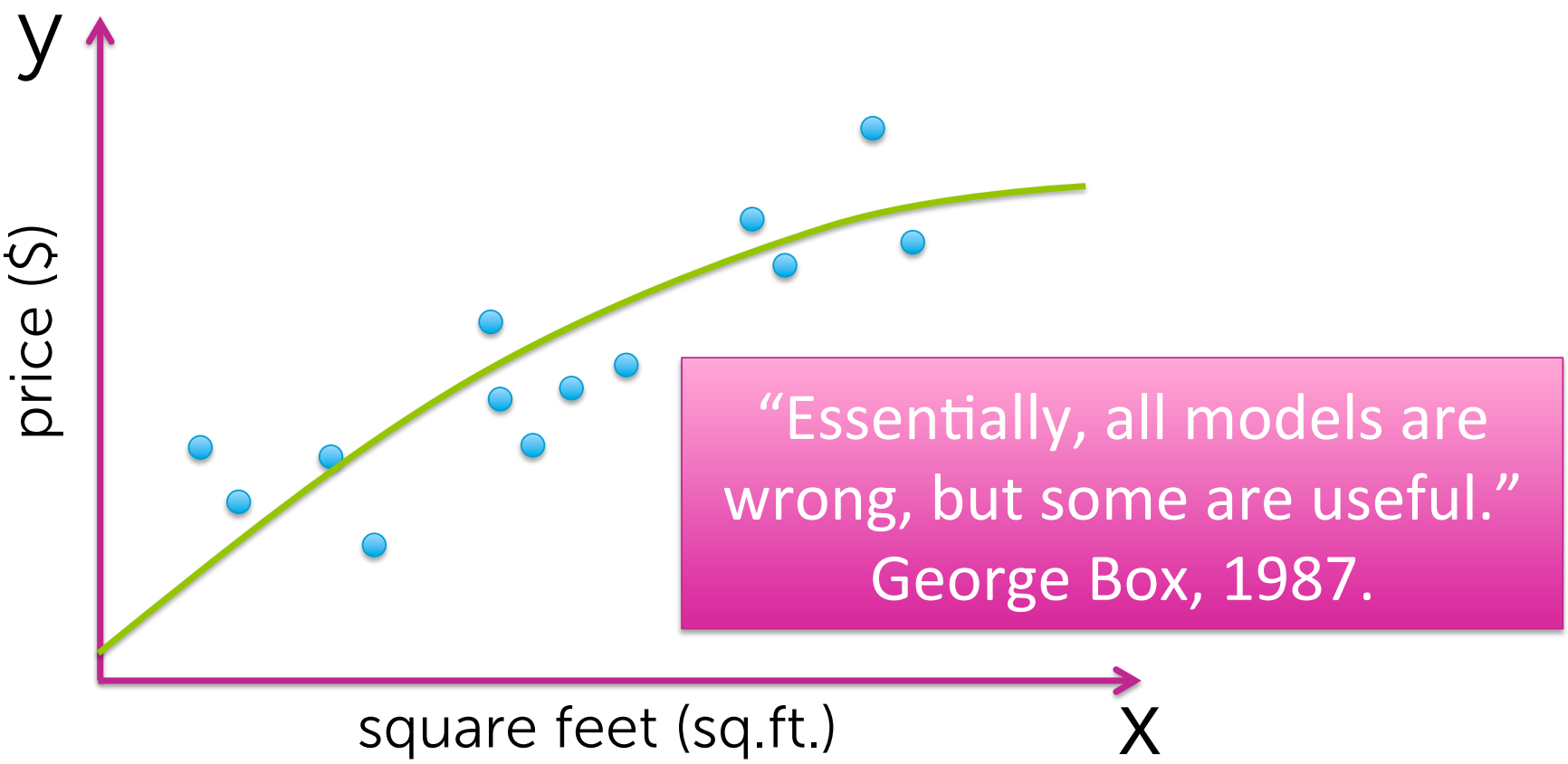
↑ expected value

⇓

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

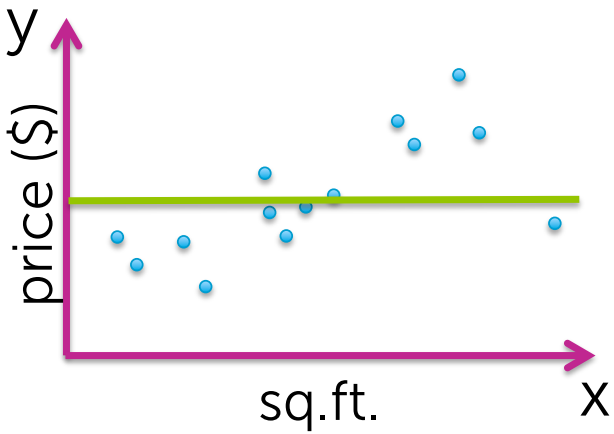
Model –

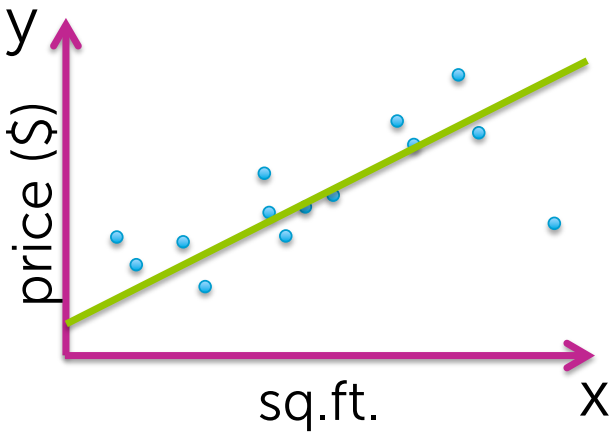
How we *assume* the world works

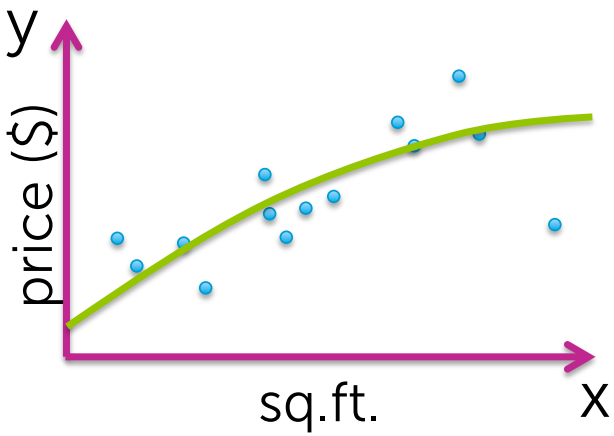


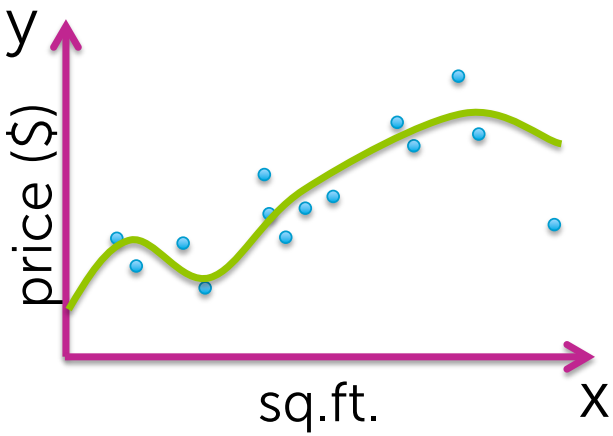
"Essentially, all models are wrong, but some are useful."
George Box, 1987.

Task 1–
Which model f(x)?

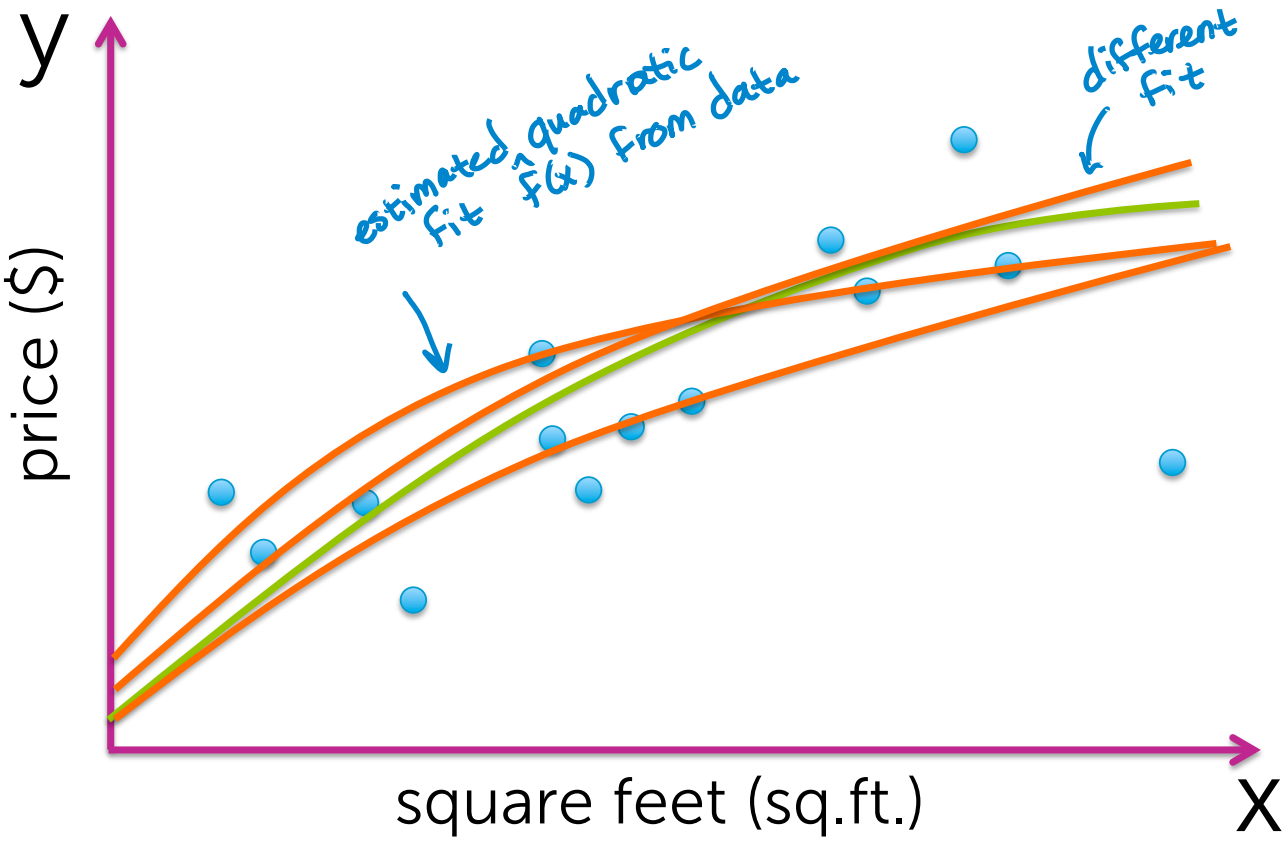




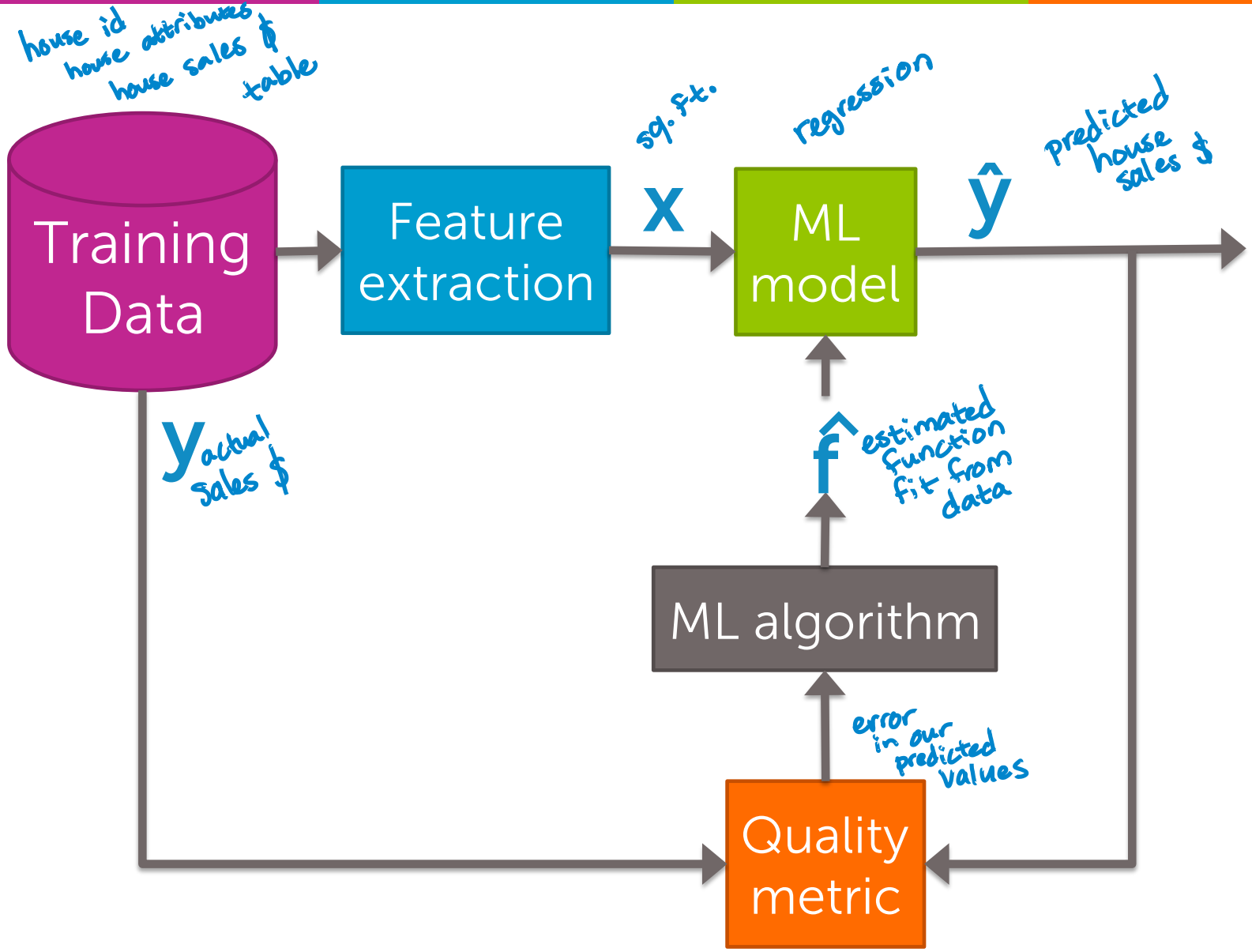




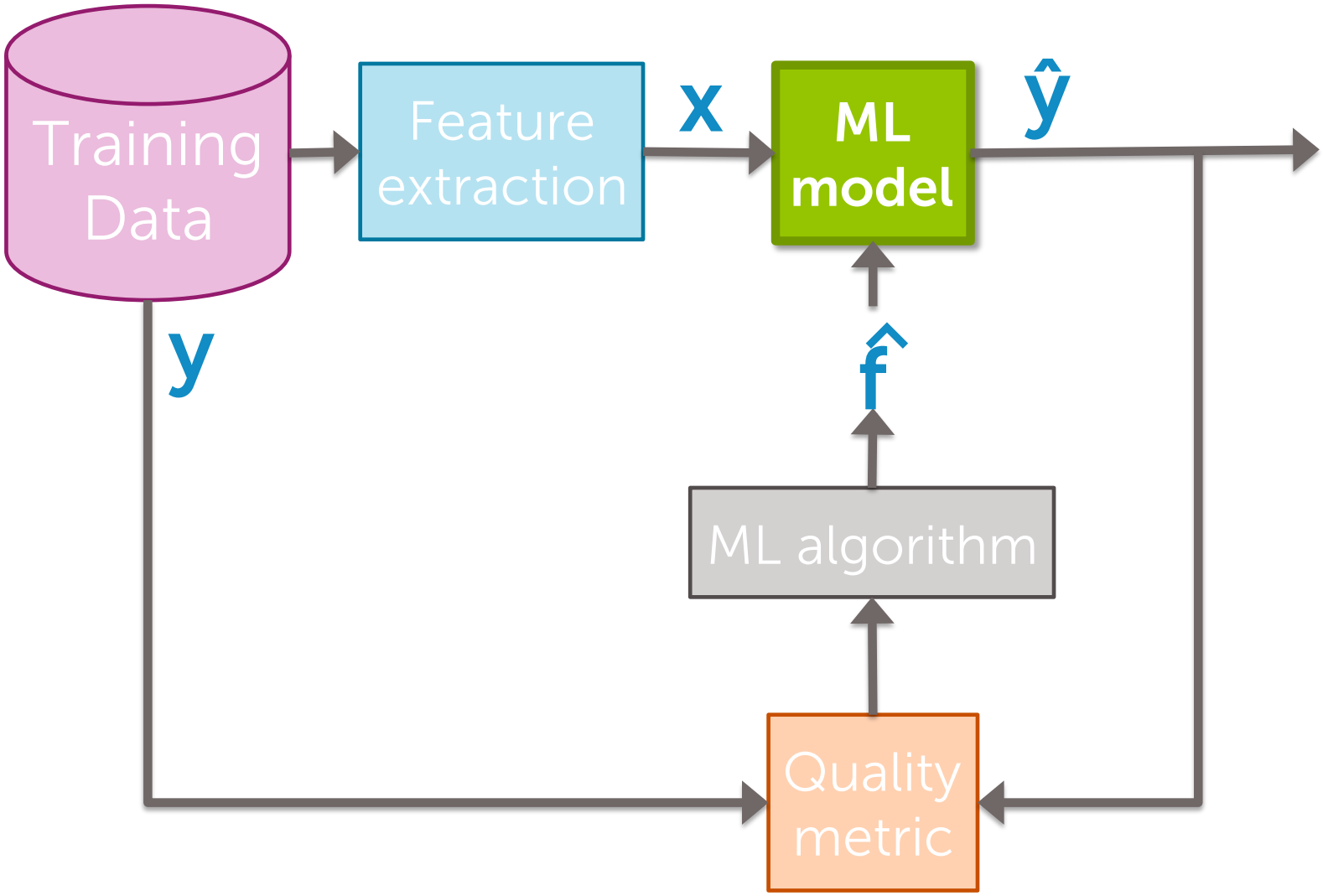
Task 2 – For a given model f(x), estimate function $\hat{f}(x)$ from data



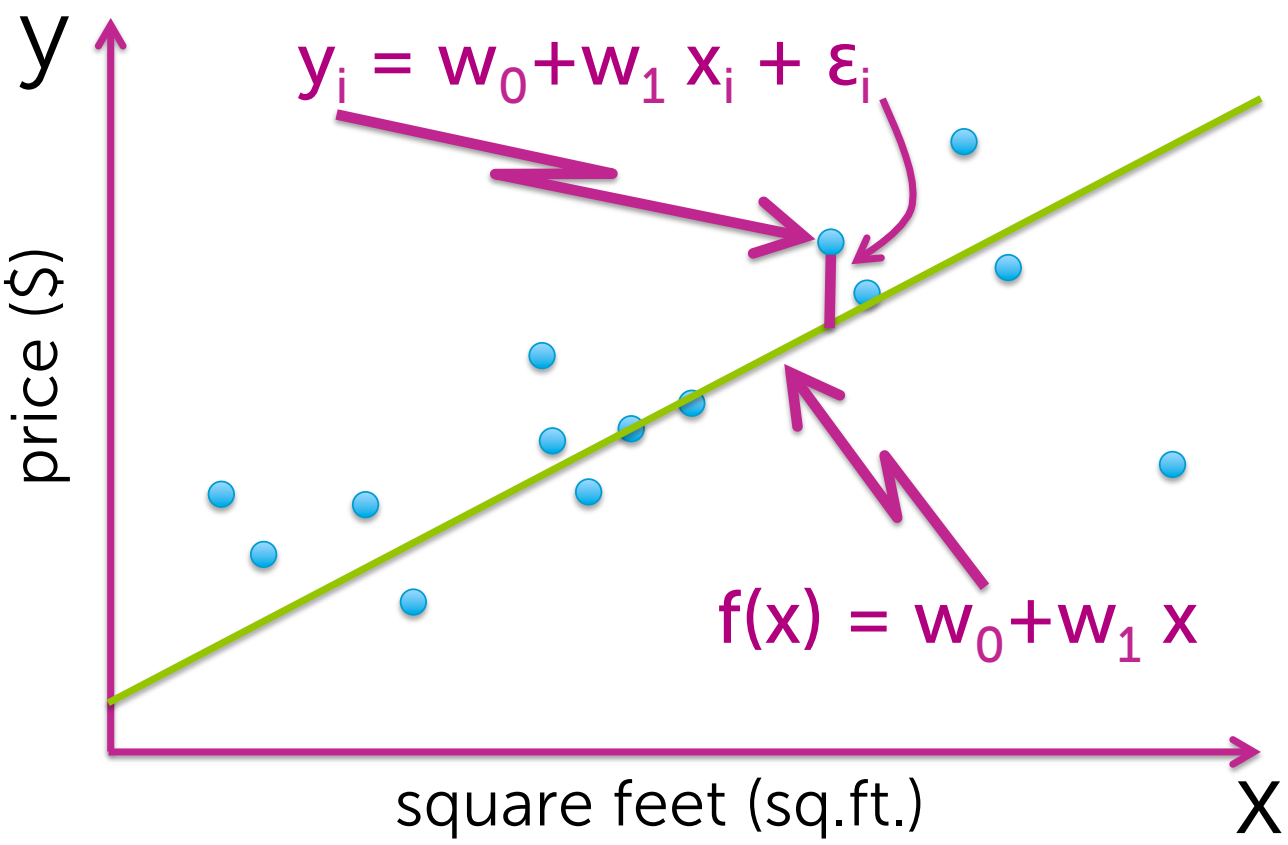
Assume model f(x) is a quadratic function


house id
house attributes
house sales $
table
Training Data
Feature extraction
sq. ft.
x
regression
ML model
ŷ
predicted house sales $
y actual sales $
f̂ estimated function fit from data
ML algorithm
error in our predicted values
Quality metric

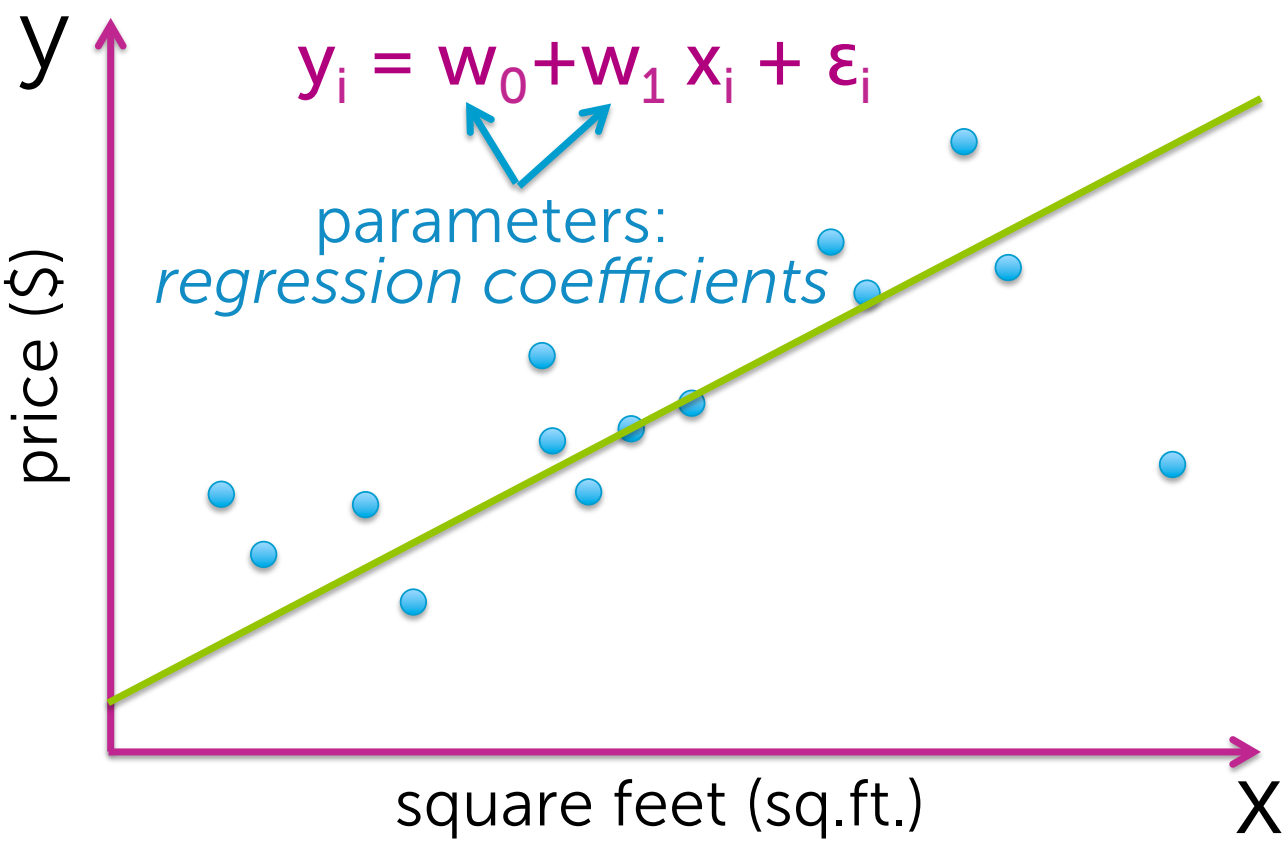
Simple linear regression


Training Data
Feature extraction
x
ML model
ŷ
y
f̂
ML algorithm
Quality metric

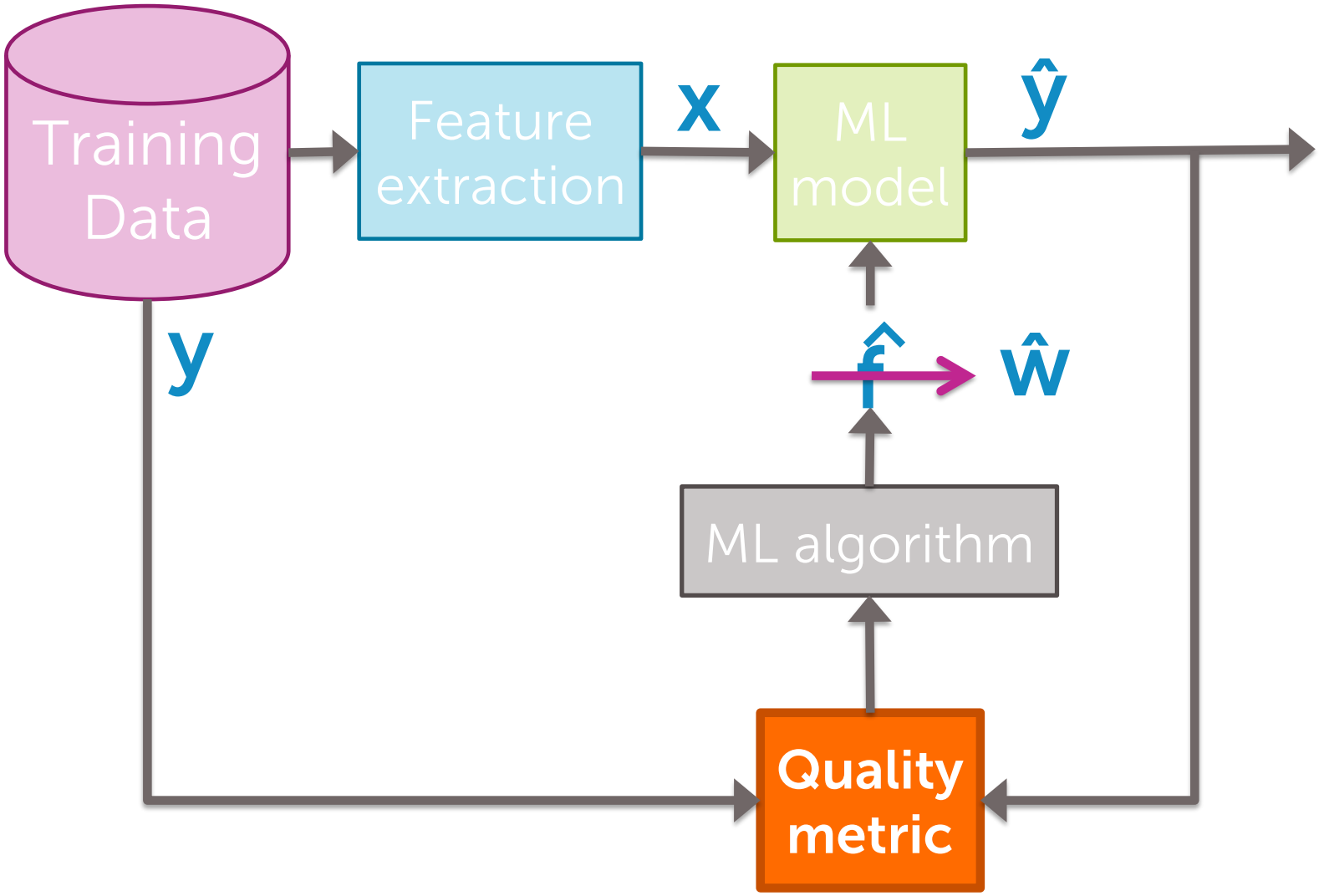
Simple linear regression model



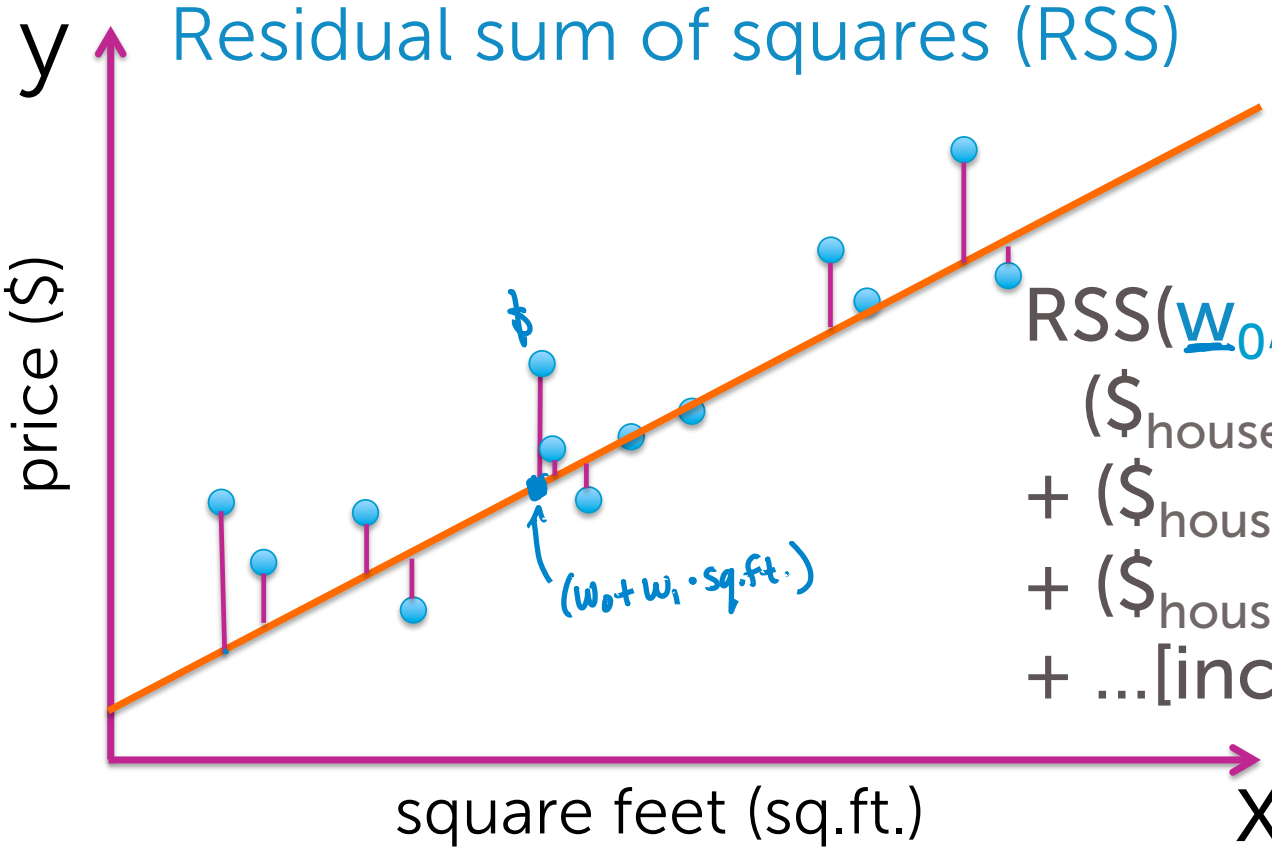
Simple linear regression model



Fitting a line to data

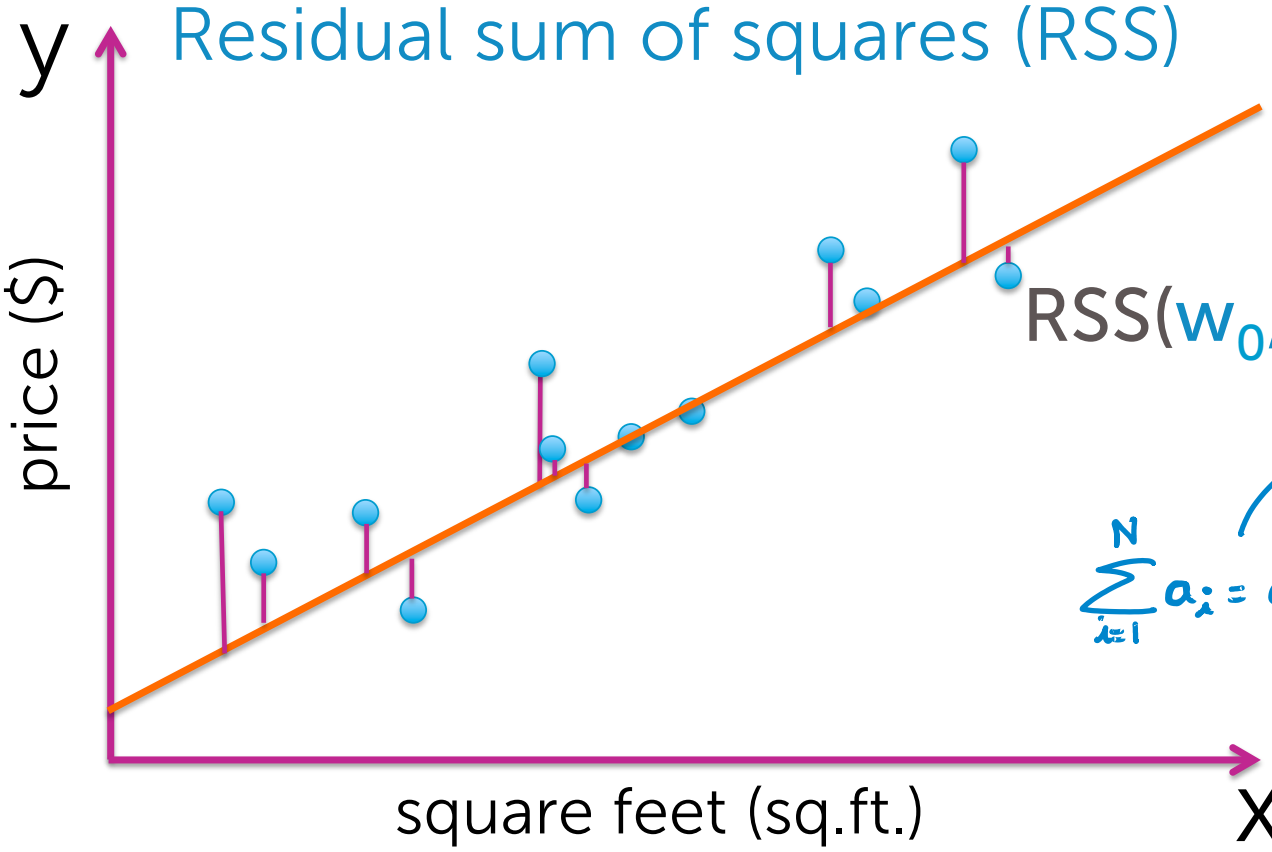

Training Data
Feature extraction
x
ML model
ŷ
y
f̂
ŵ
ML algorithm
Quality metric

"Cost" of using a given line



$$RSS(w_0, w_1) = (\$_{house\ 1} - [w_0 + w_1 sq.ft._{house\ 1}])^2 + (\$_{house\ 2} - [w_0 + w_1 sq.ft._{house\ 2}])^2 + (\$_{house\ 3} - [w_0 + w_1 sq.ft._{house\ 3}])^2 + \ldots [\text{include all training houses}]$$

"Cost" of using a given line



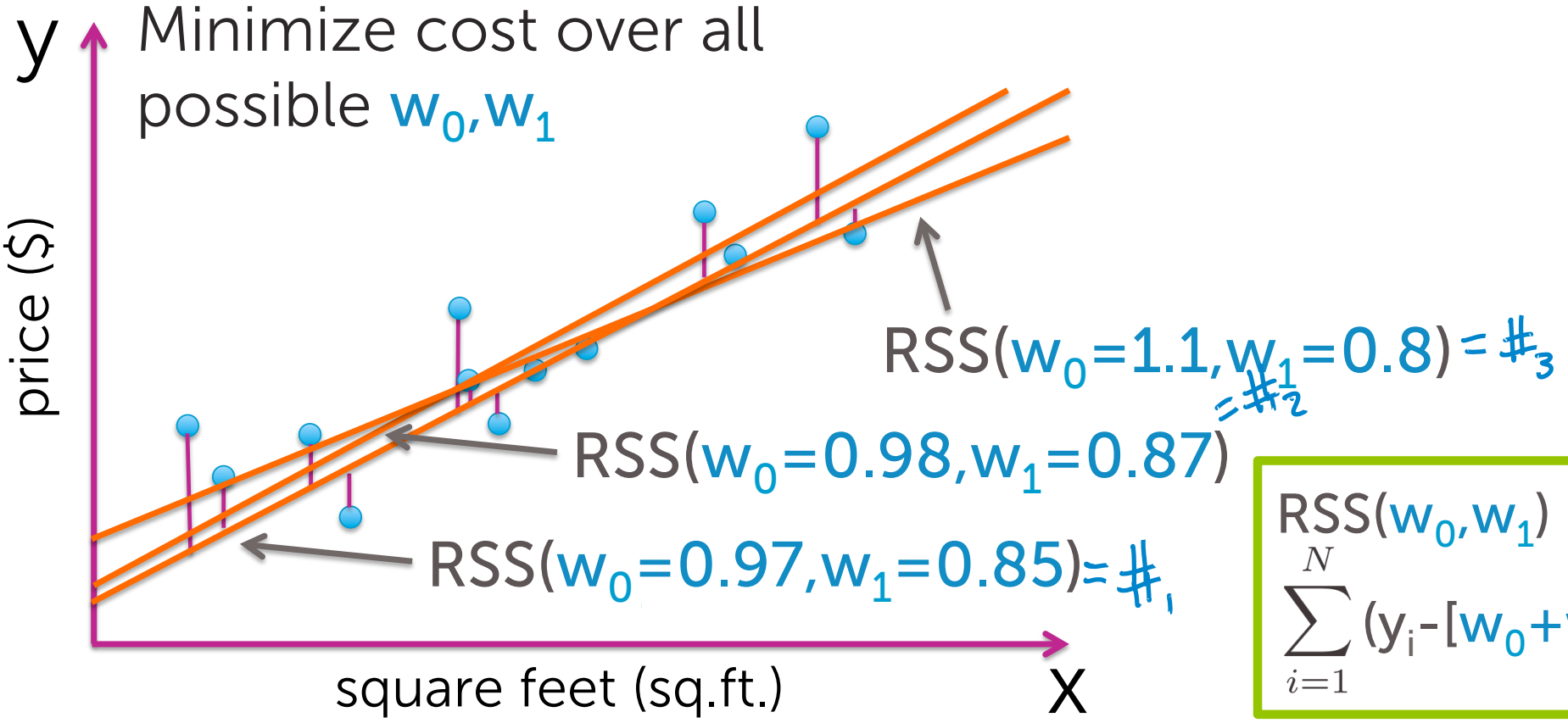
Residual sum of squares (RSS)

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

$\sum_{i=1}^{N} a_i = a_1 + a_2 + \ldots + a_N$

Here, $a_i = (y_i - [w_0 + w_1 x_i])^2$

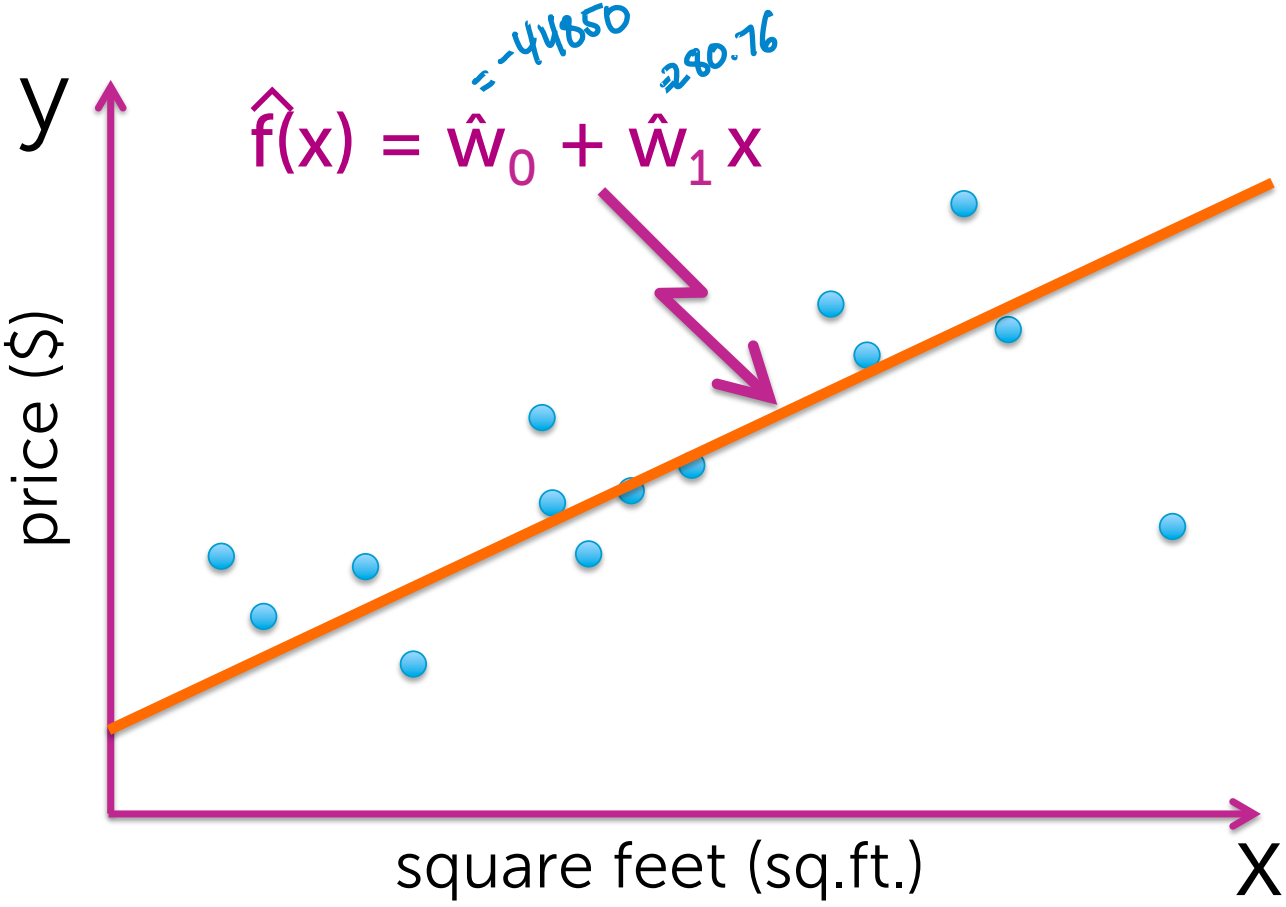
Find "best" line



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

The fitted line: use + interpretation

Model vs. fitted line



Regression model:

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

parameters (unknown variables)

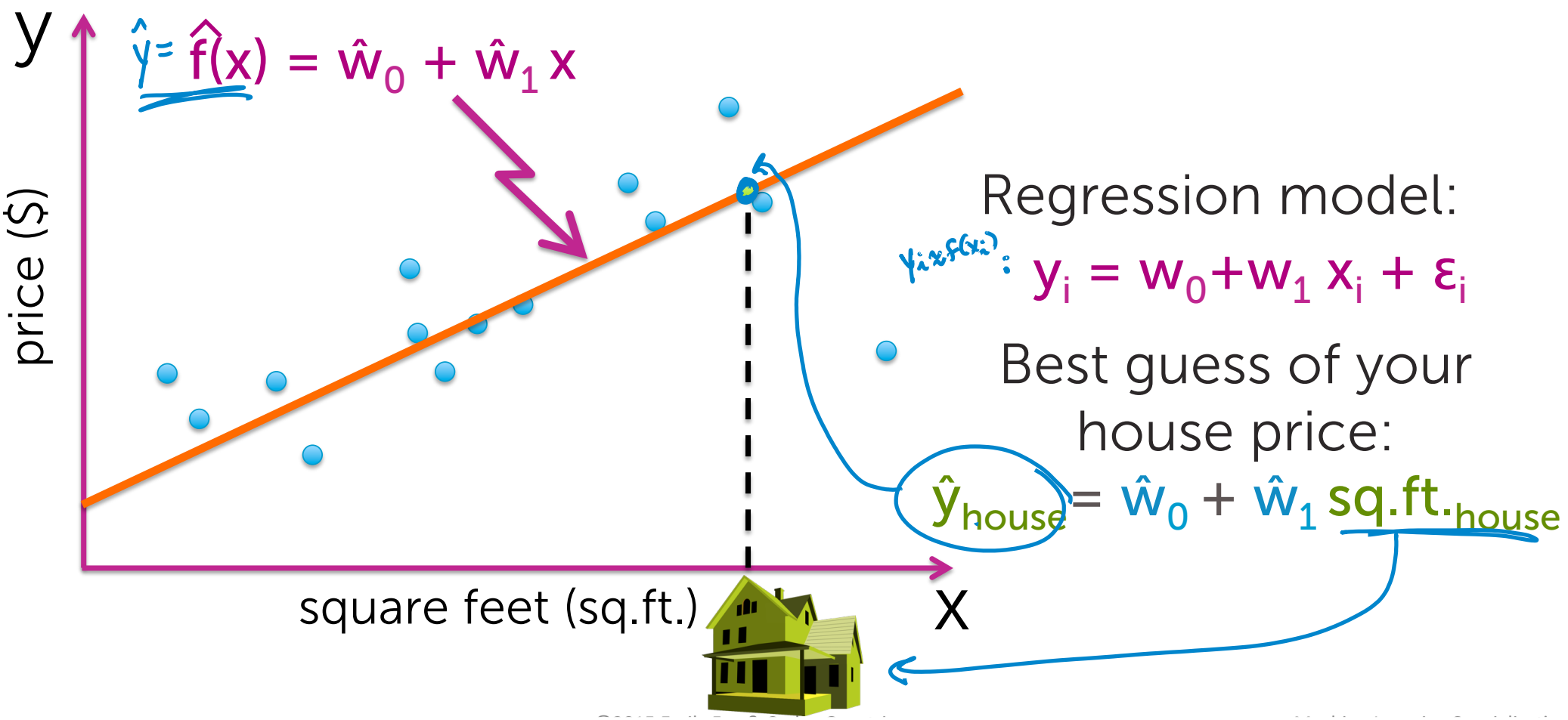
Estimated parameters:

$$\hat{w}_0 = -44850 \quad , \quad \hat{w}_1 = 280.76$$

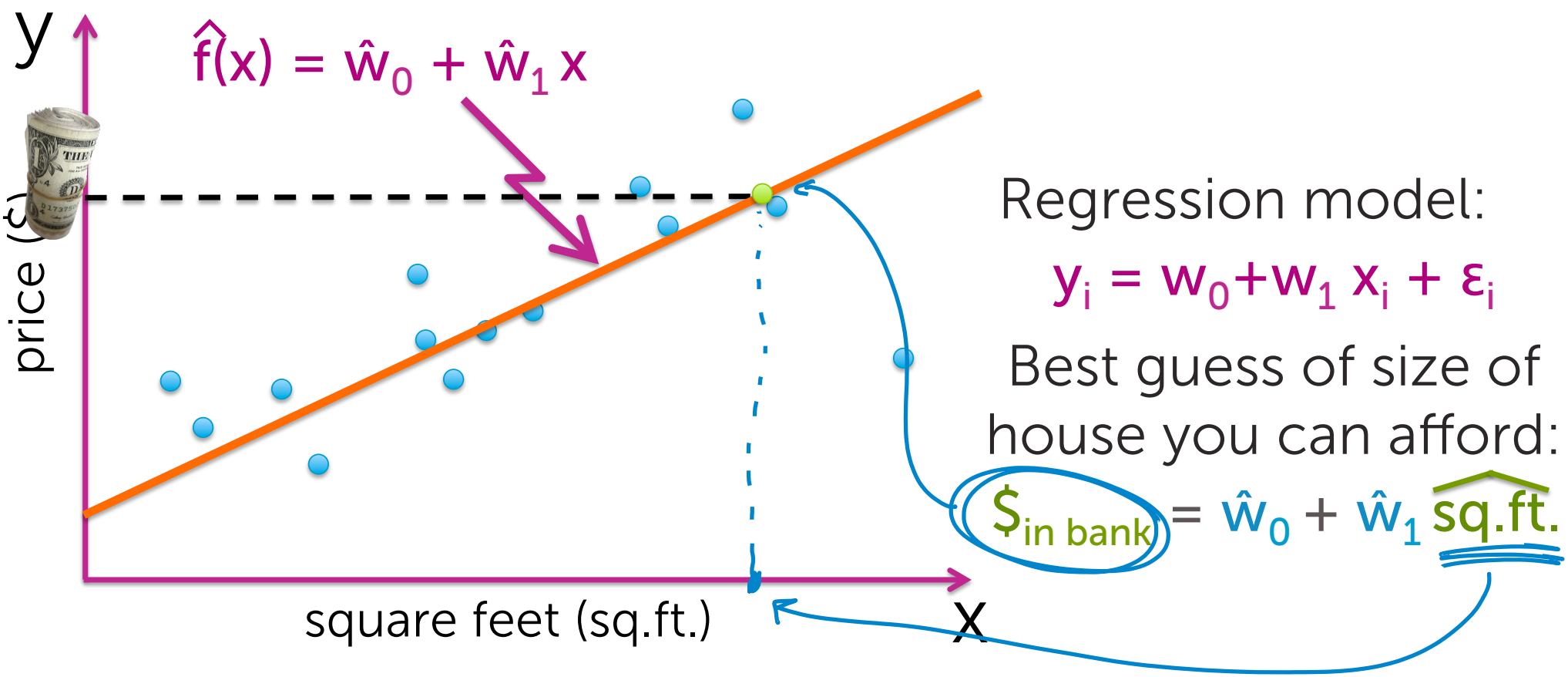
take actual values

Seller:

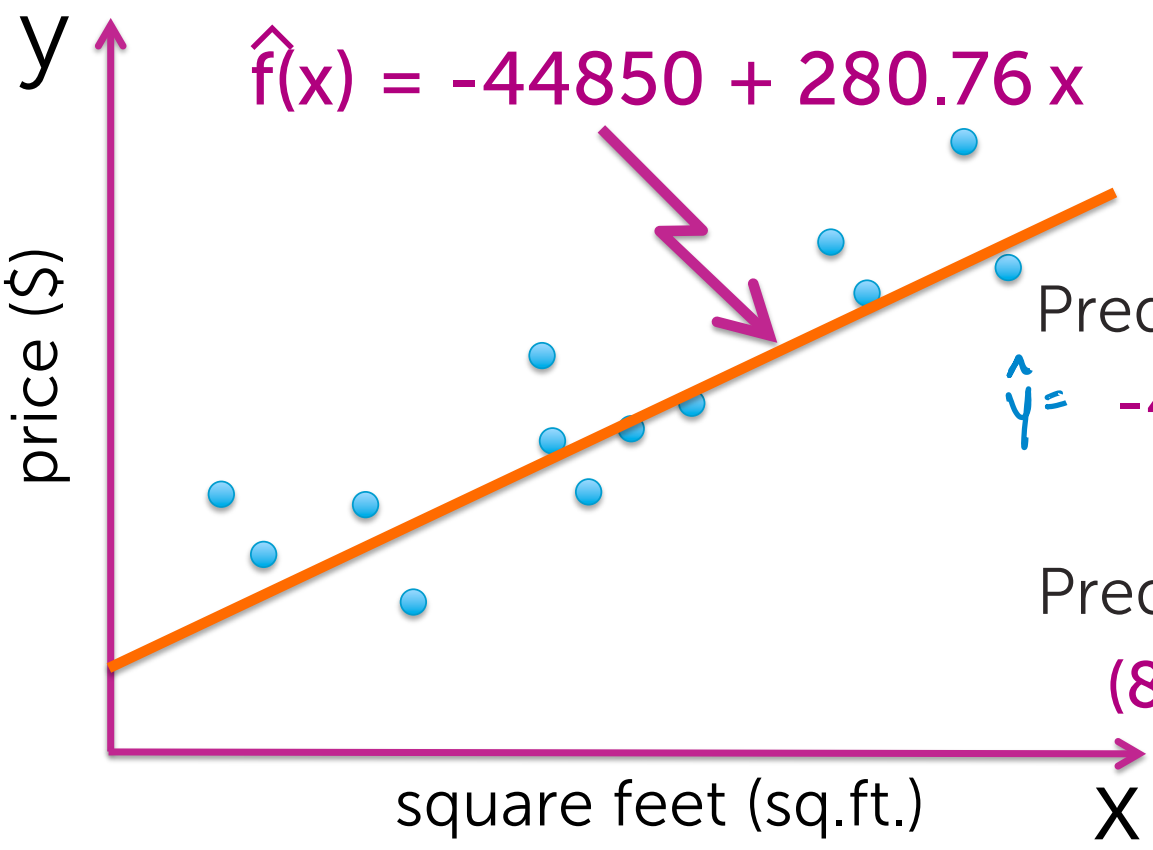
Predicting your house price



Buyer:
Predicting size of house



A concrete example



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

$\hat{y} = -44850 + 280.76 * 2{,}640$

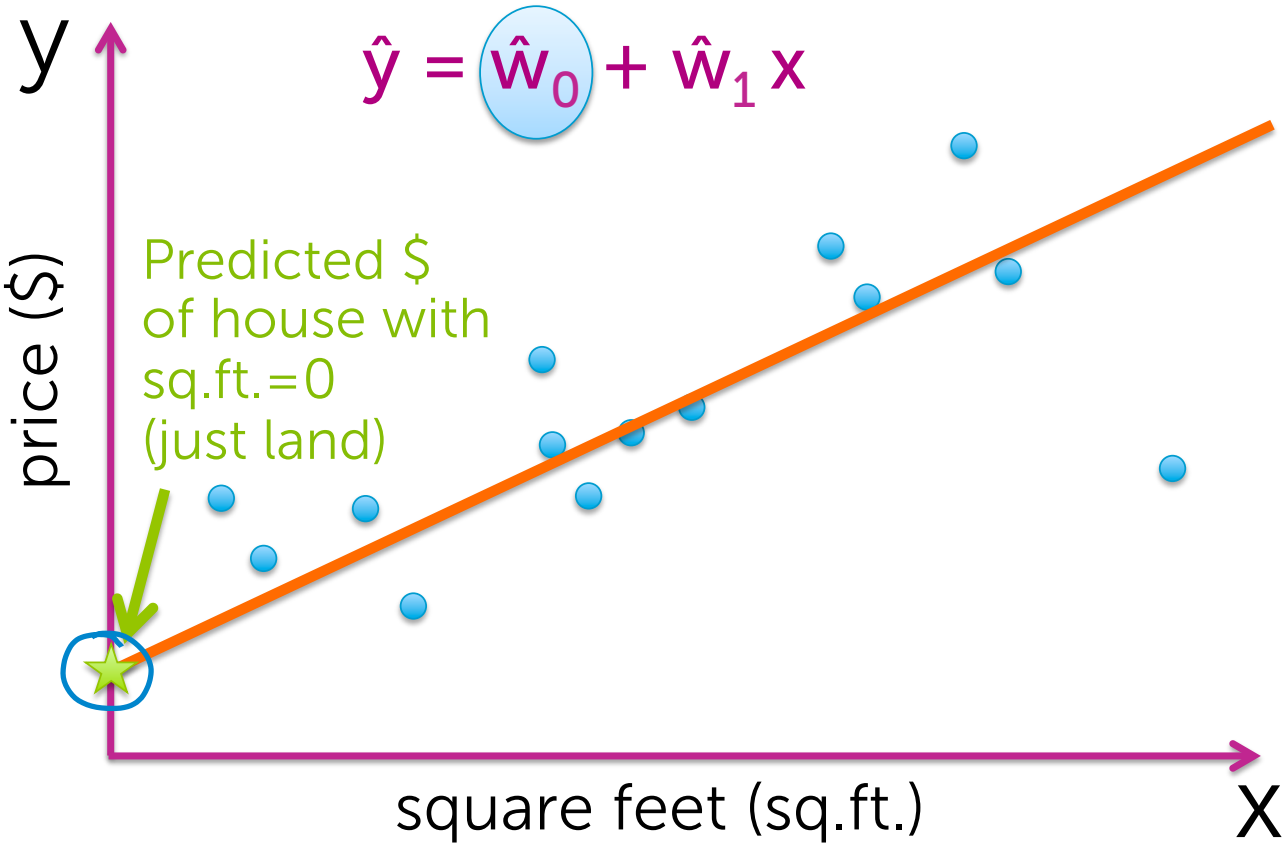
$= \$696{,}356$

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

$(859000+44850)/\ 280.76$

$= 3{,}219$ sq.ft.

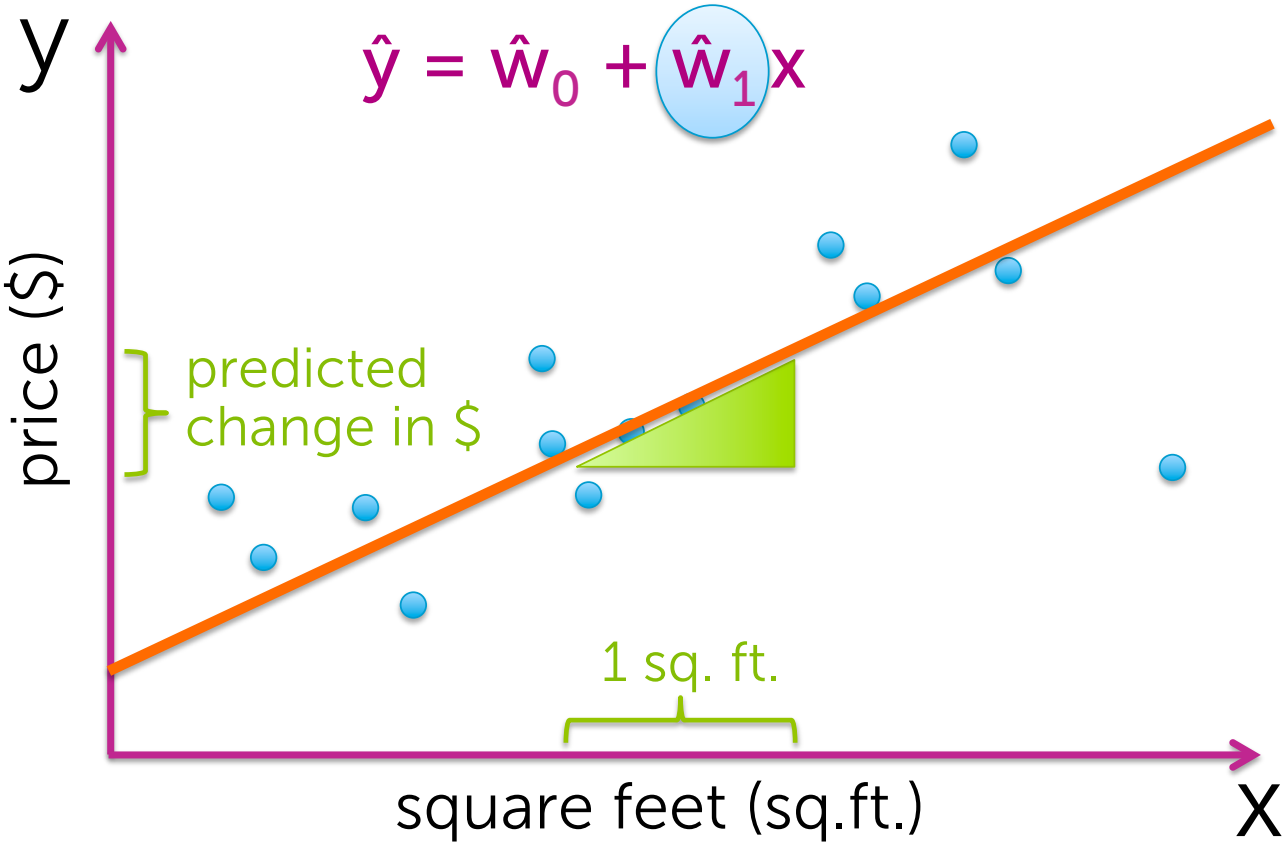
Interpreting the coefficients



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

not very meaningful

Interpreting the coefficients



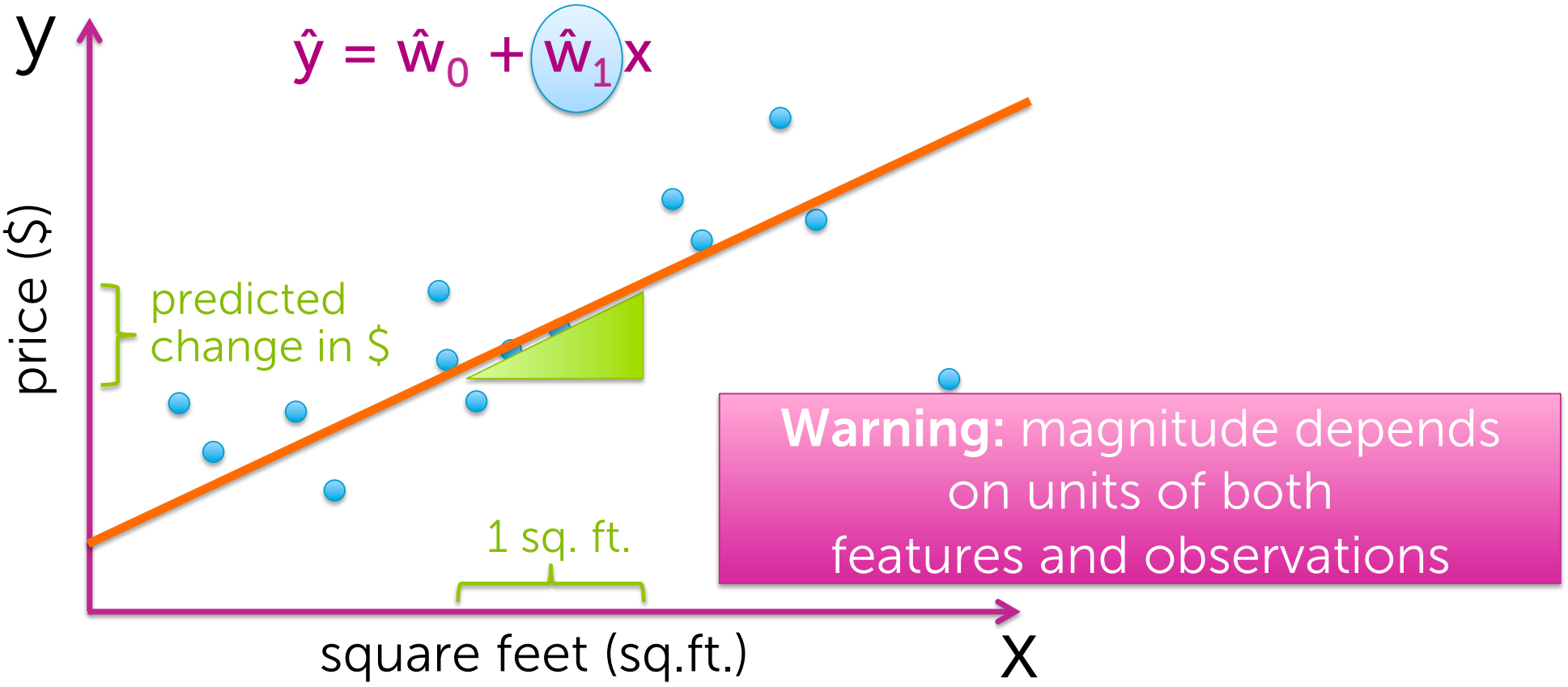
$\hat{\$}_{1001\ sq.ft.} - \hat{\$}_{1000\ sq.ft.}$

$= \hat{w}_0 + \hat{w}_1 \cdot 1001\ sq.ft. - (\hat{w}_0 + \hat{w}_1 1000\ sq.ft.)$

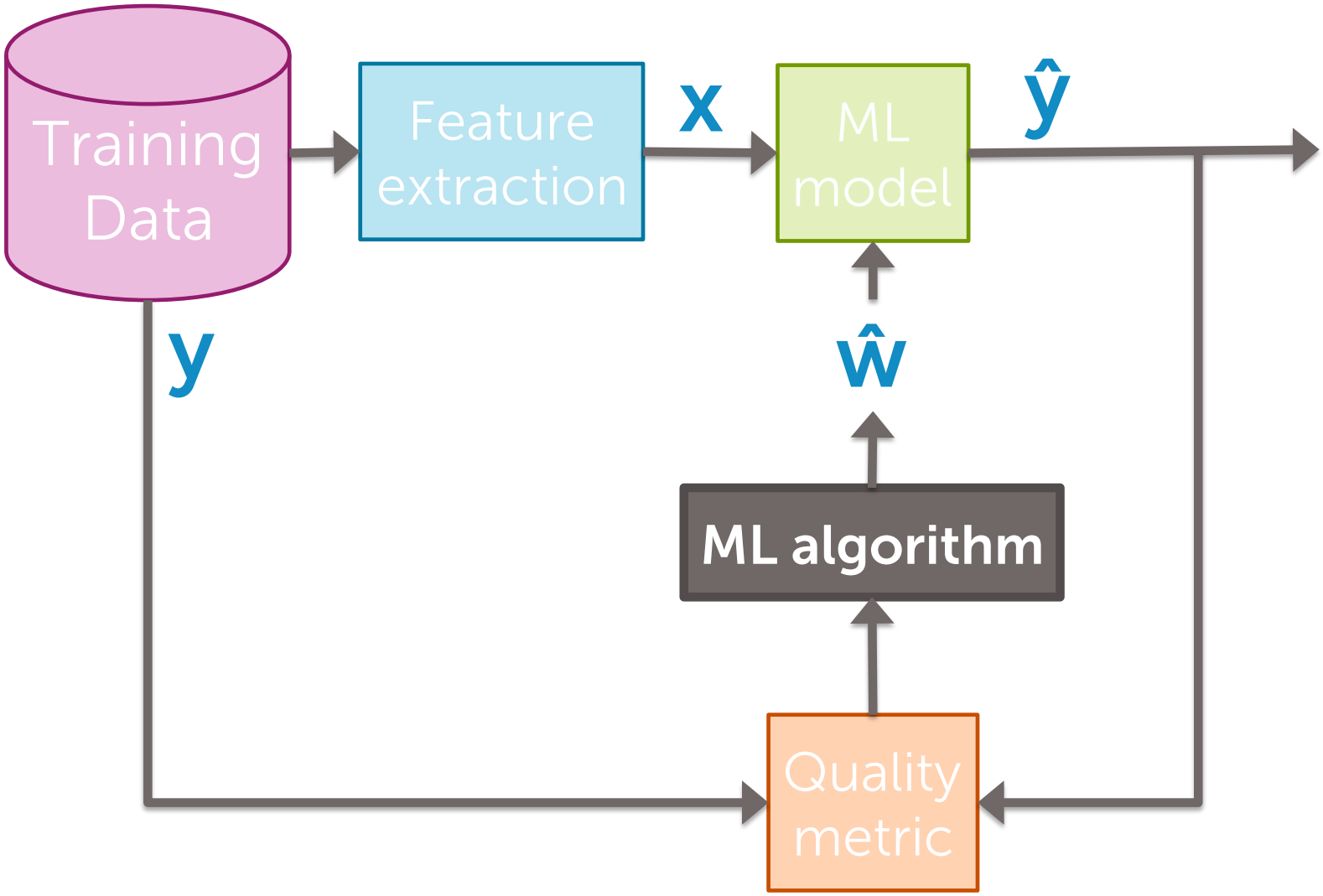
$= \hat{w}_1$

predicted change in the output per unit change in input

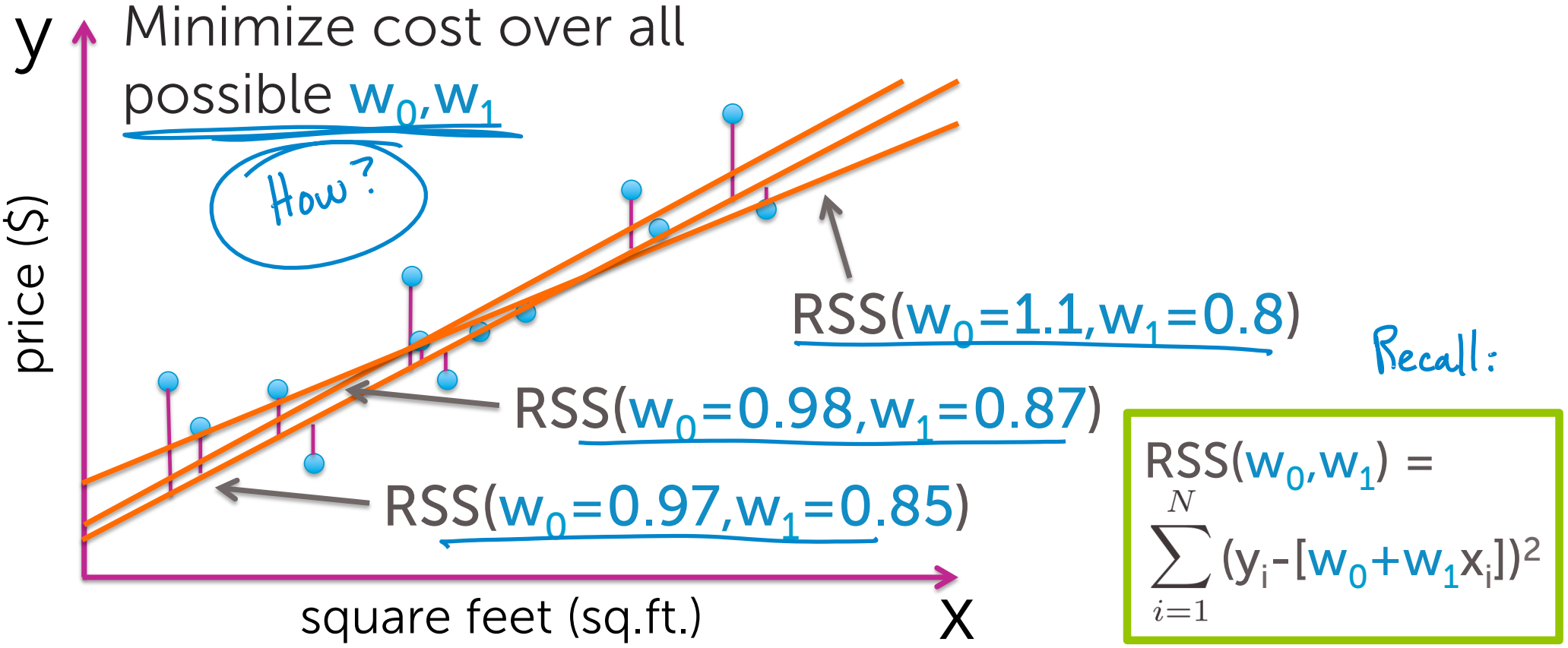
Interpreting the coefficients



Algorithms for fitting the model


Training Data
Feature extraction
x
ML model
ŷ
y
ŵ
ML algorithm
Quality metric

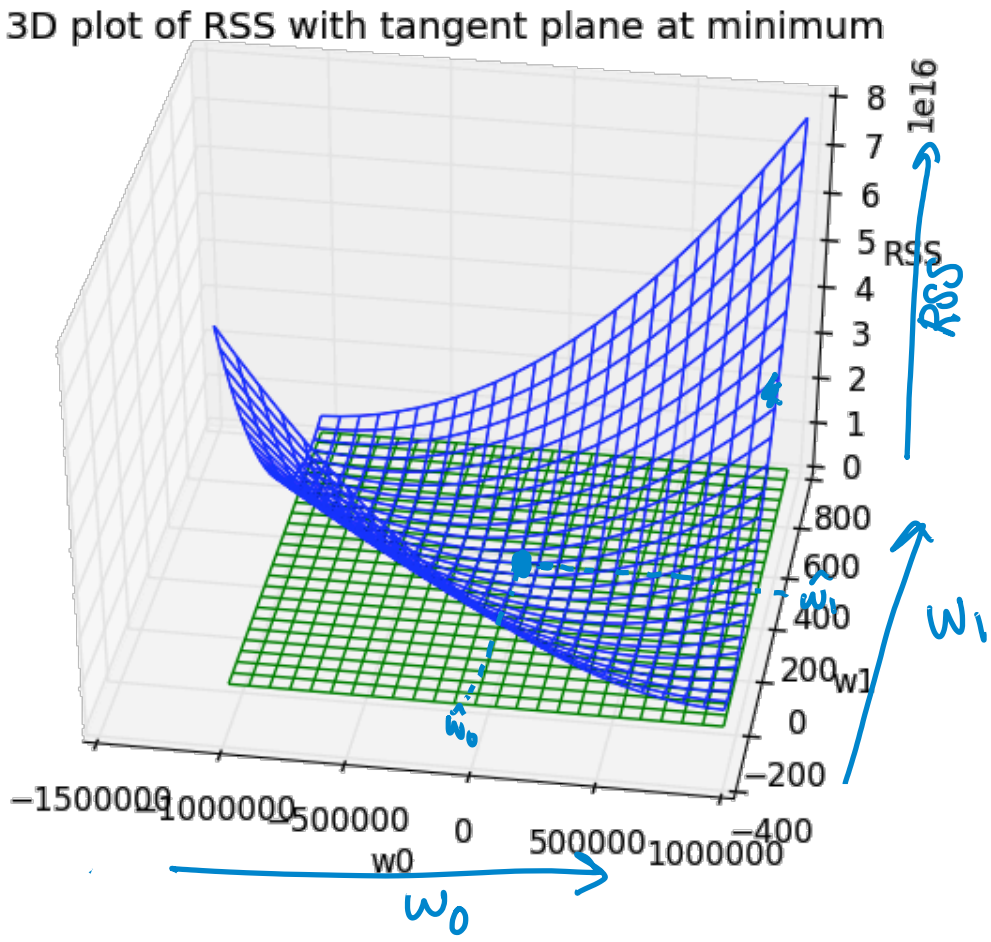
Find "best" line



Recall:

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

Minimizing the cost



Minimize function over all possible w_0, w_1

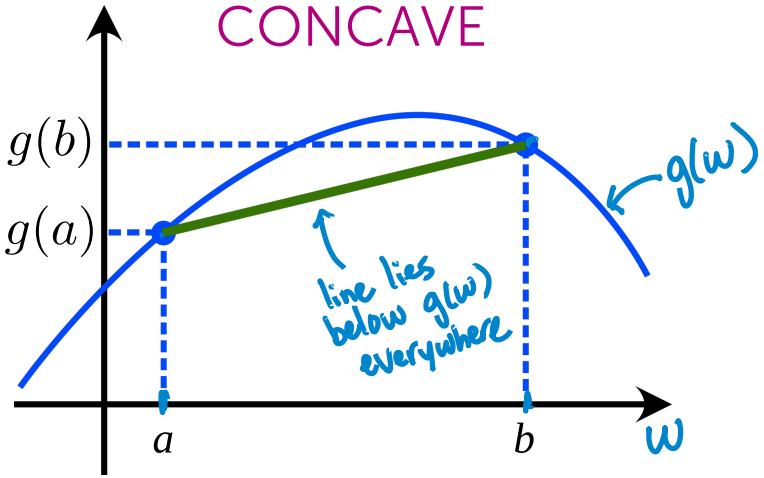
$$\min_{w_0, w_1} \sum_{i=1}^{N} (y_i - [w_0 + w_1 x_i])^2$$

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

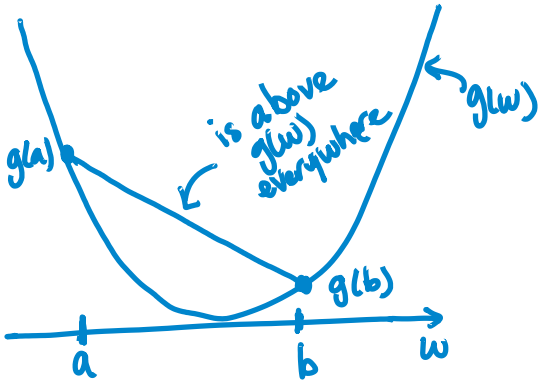
An aside on optimization

Convex/concave functions

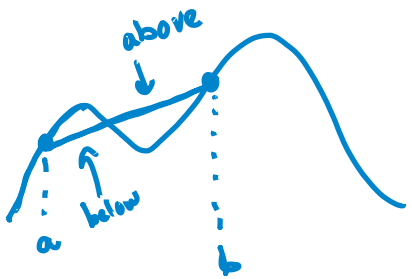
CONCAVE

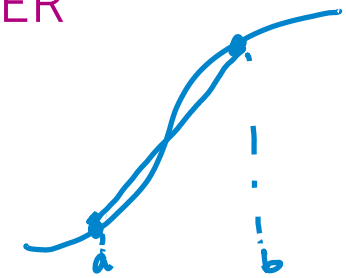


CONVEX



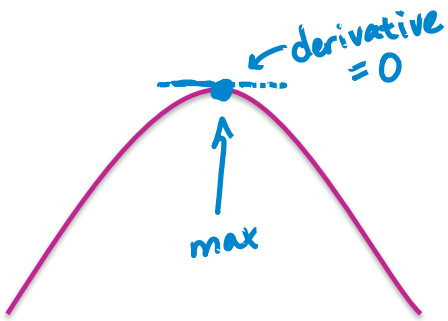
NEITHER





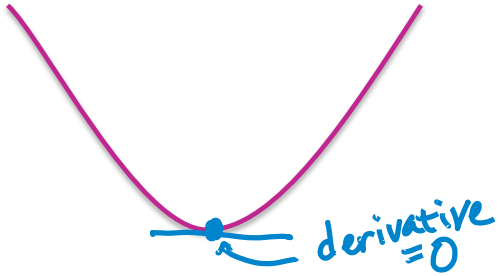
Finding the max or min analytically

CONCAVE



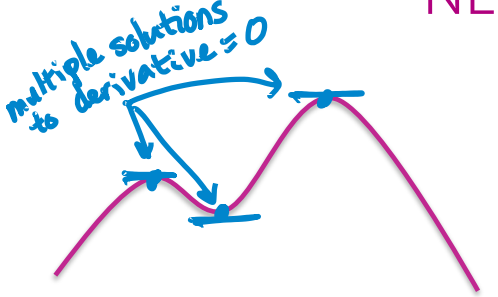
$\max_w g(w)$

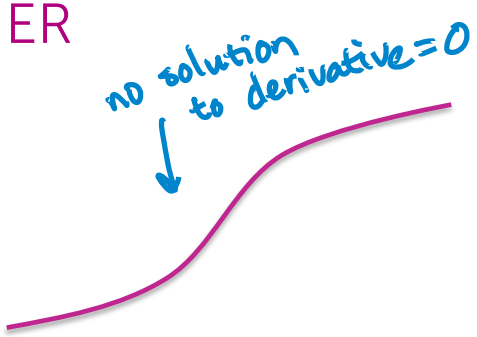
CONVEX



$\min_w g(w)$

NEITHER





Example:

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

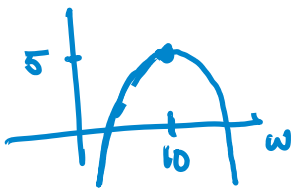
$\frac{dg(w)}{dw} = 0 - 2(w-10)^1 \cdot 1$

$= -2w + 20$

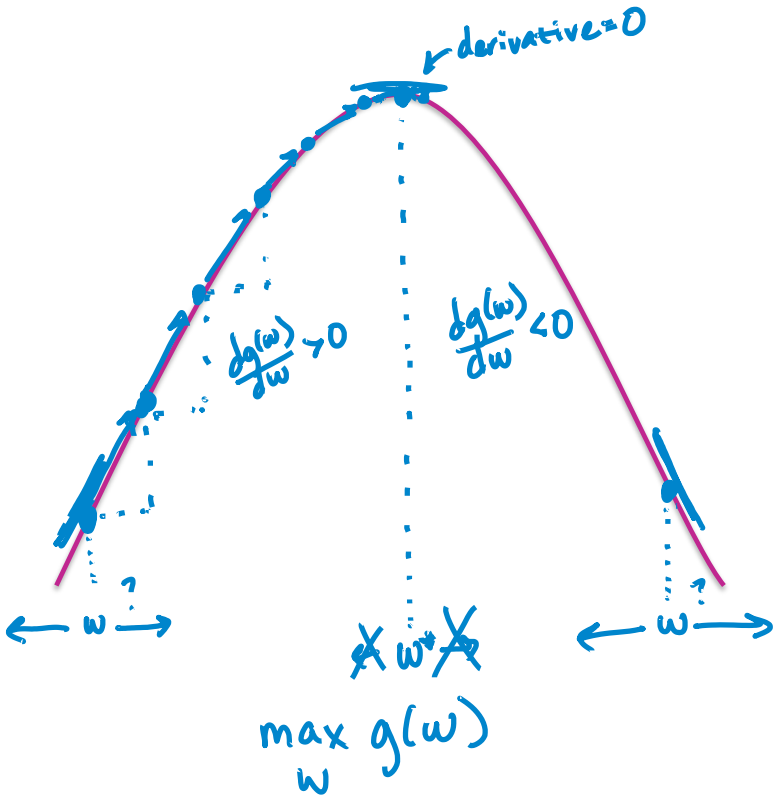
set derivate = 0:

$-2w + 20 = 0$

$w = 10$



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?)

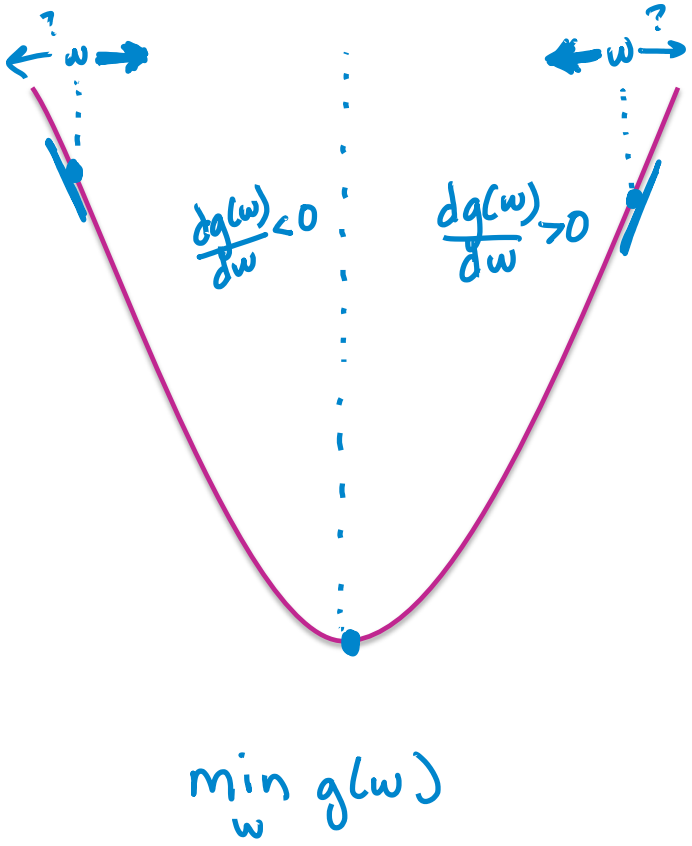
while not converged

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

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:

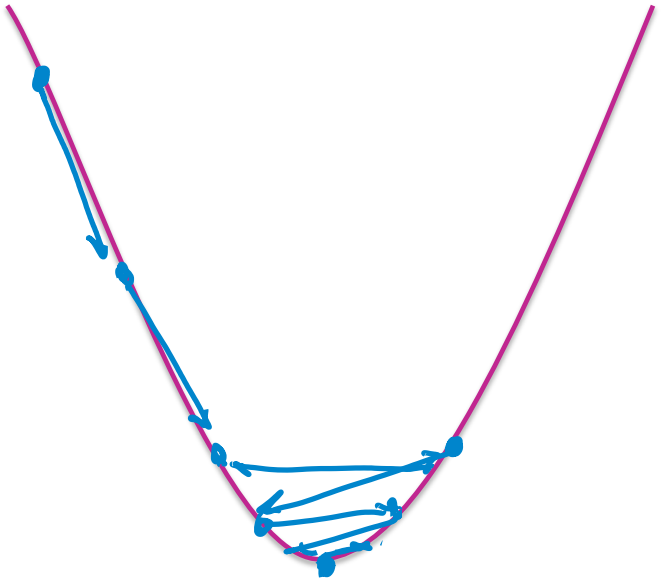
while not converged

$$w^{(t+1)} \leftarrow w^{(t)} - \eta \left.\frac{dg}{dw}\right|_{w^{(t)}}$$

Choosing the stepsize—

Fixed stepsize

η

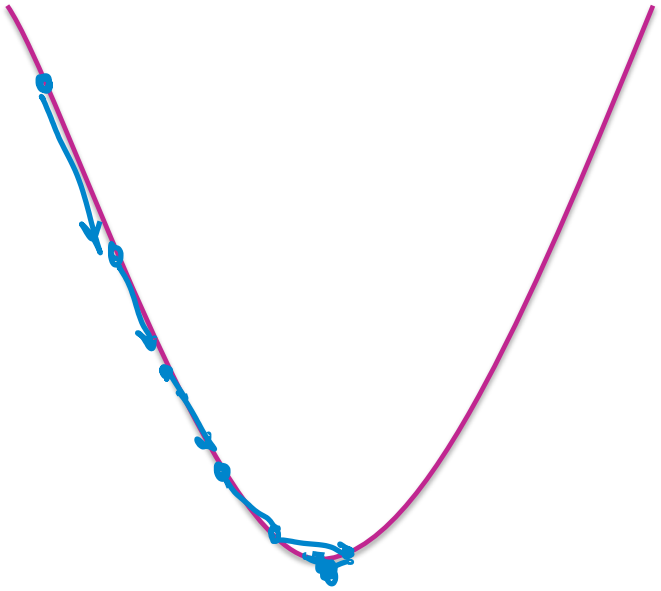


$\eta = 0.1$

Choosing the stepsize—

Decreasing stepsize

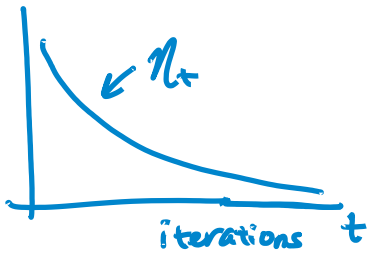
or stepsize schedule



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

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

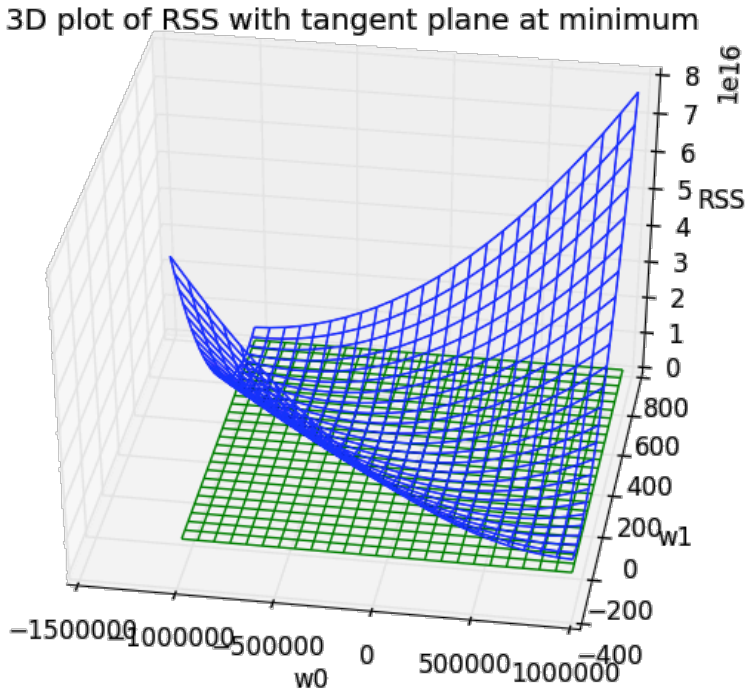
ϵ: threshold to be set

Algorithm:

while not converged

$$w^{(t+1)} \leftarrow w^{(t)} - \eta \left.\frac{dg}{dw}\right|_{w^{(t)}}$$

Moving to multiple dimensions: Gradients



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

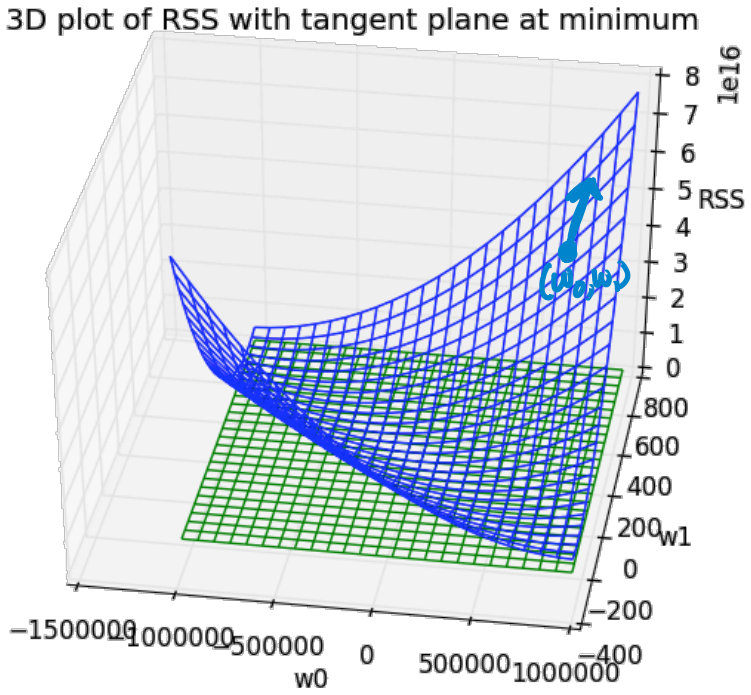
gradient

$[w_0, w_1, \ldots, w_p]$

$(p+1)$-dimensional vector

partial derivative is like a derivate with respect to w_1 treating all other variables as constants

Gradient example



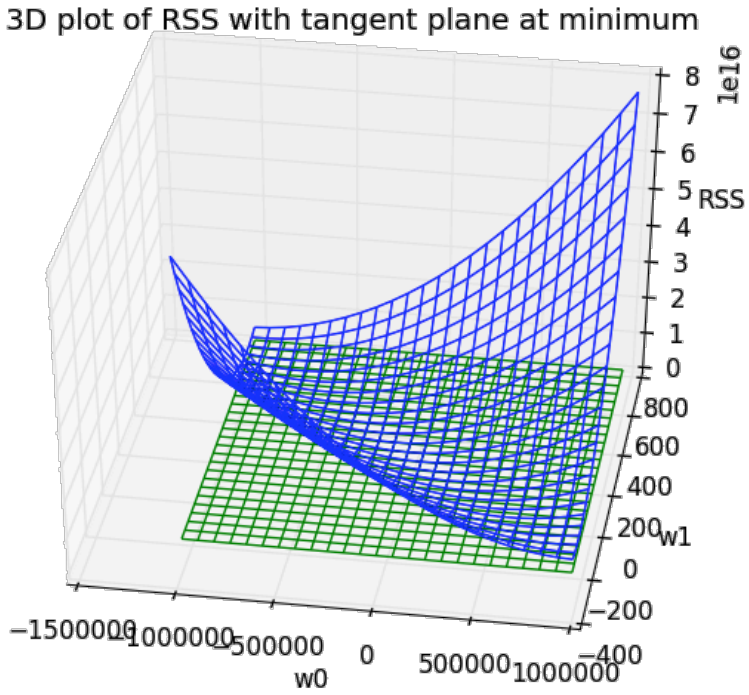
$$g(w) = 5w_0 + 10w_0w_1 + 2w_1^2$$

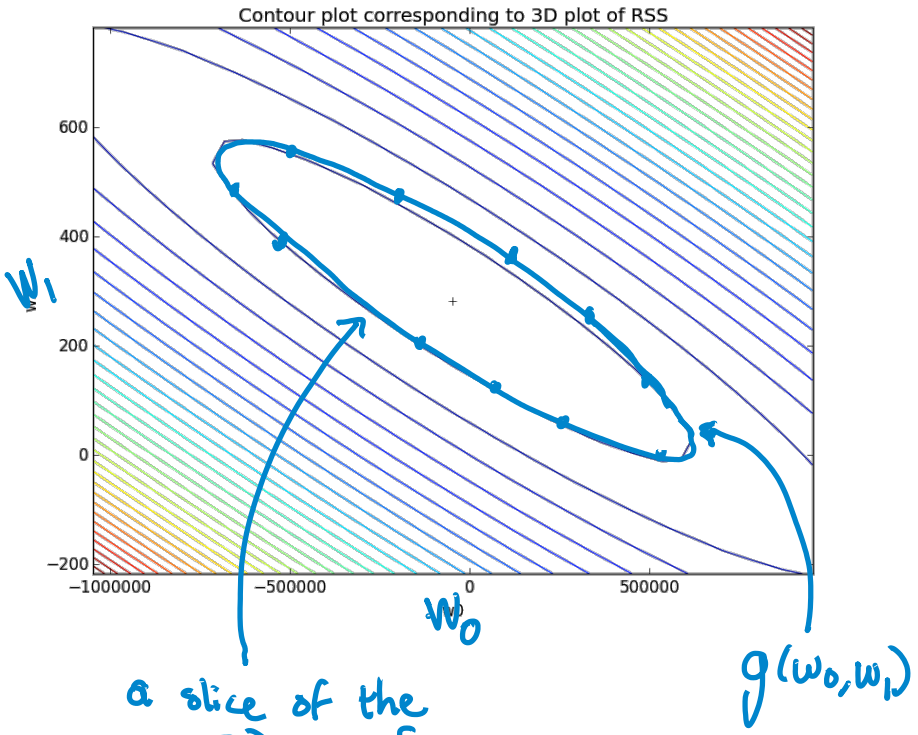
$$\frac{\partial g}{\partial w_0} = 5 + 10w_1$$

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

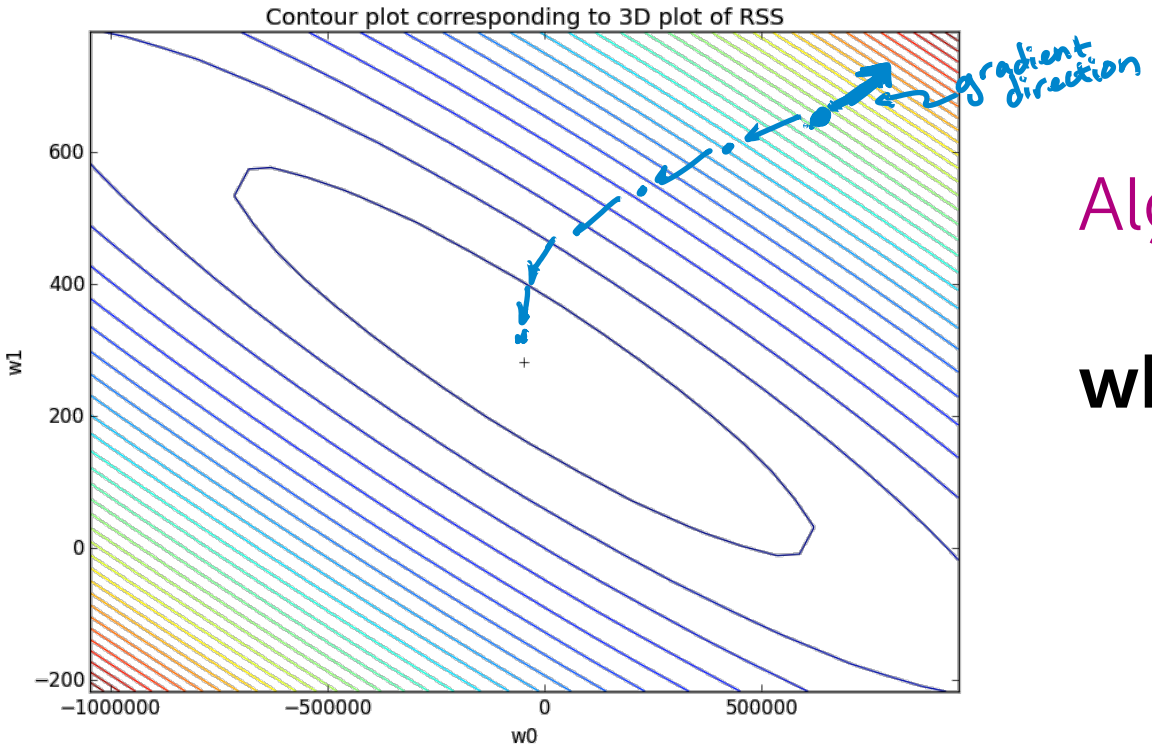
$$\nabla g(w) = \begin{bmatrix} 5 + 10w_1 \\ 10w_0 + 4w_1 \end{bmatrix}$$

Contour plots





Gradient descent



Algorithm:

while not converged

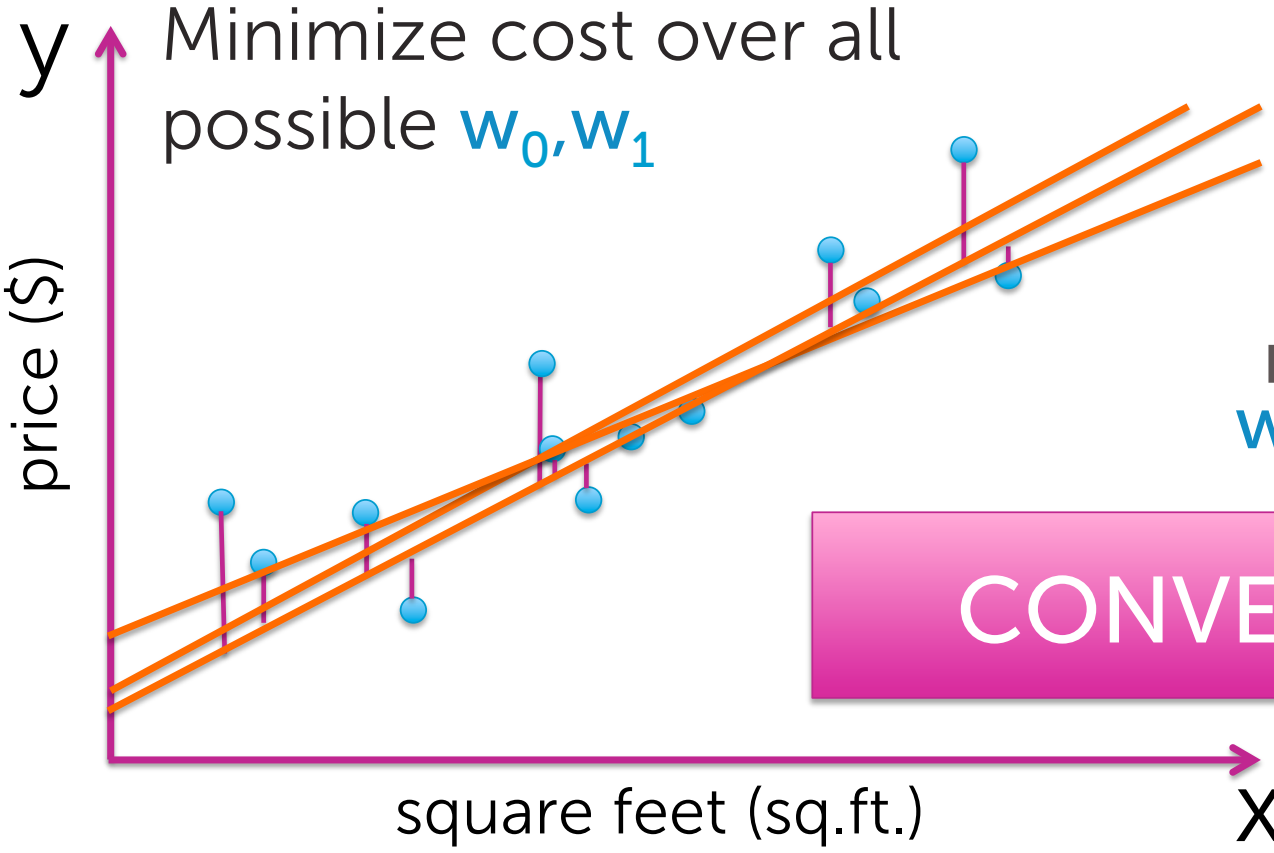
$$w^{(t+1)} \leftarrow w^{(t)} - \eta \nabla g(w^{(t)})$$

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

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

Finding the least squares line

Find "best" line



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

CONVEX

⇒ solution is unique

+ gradient descent alg. will converge to minimum

Compute the gradient

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

Aside:

$$\frac{d}{dw} \sum_{i=1}^{N} g_i(w) = \frac{d}{dw} (g_1(w) + g_2(w) + \ldots g_N(w))$$
$$= \frac{d}{dw} g_1(w) + \frac{d}{dw} g_2(w) + \ldots \frac{d}{dw} g_N(w)$$
$$= \sum_{i=1}^{N} \frac{d}{dw} g_i(w)$$

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

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

Taking the derivative w.r.t. w_0

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

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

Compute the gradient

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

Taking the derivative w.r.t. w_1

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

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

Compute the gradient

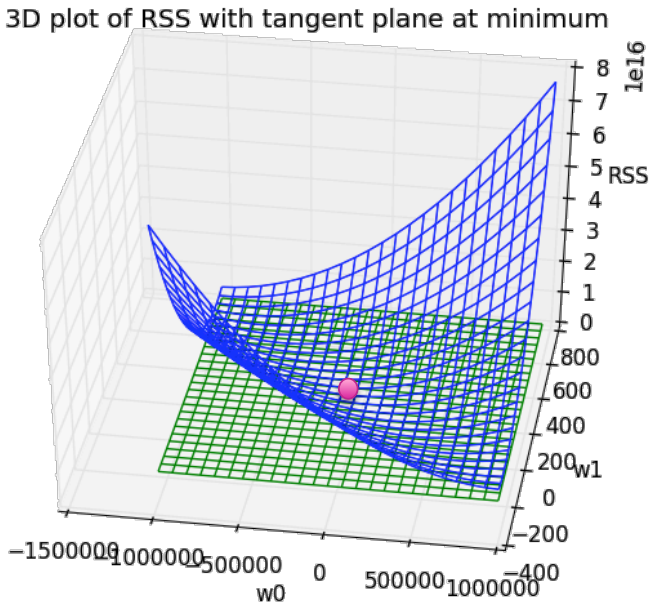
$$RSS(w_0,w_1) = \sum_{i=1}^{N} (y_i-[w_0+w_1x_i])^2$$

Putting it together:

$$\nabla RSS(w_0,w_1) = \begin{bmatrix} -2\sum_{i=1}^{N} [y_i - (w_0+w_1x_i)] \\ -2\sum_{i=1}^{N} [y_i - (w_0+w_1x_i)]x_i \end{bmatrix}$$

Approach 1: Set gradient = 0

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



top term:

$$\hat{w}_0 = \frac{\sum_{i=1}^{N} y_i}{N} - \hat{w}_1 \frac{\sum_{i=1}^{N} x_i}{N}$$

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

bottom term:

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

plug in

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

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:

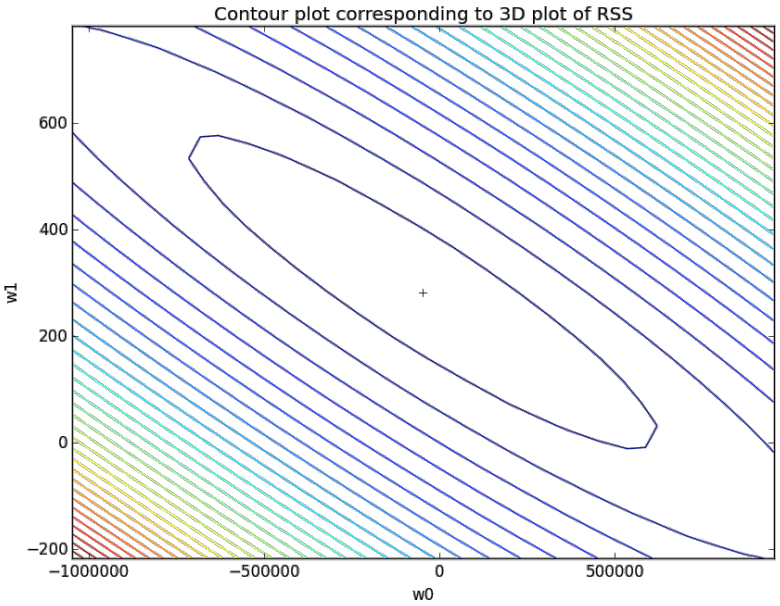
actual house sales observation

predicted value $\hat{y}_i(w_0, w_1)$

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

Approach 2: Gradient descent

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



while not converged $(-2)\cdot(-\eta)$

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

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

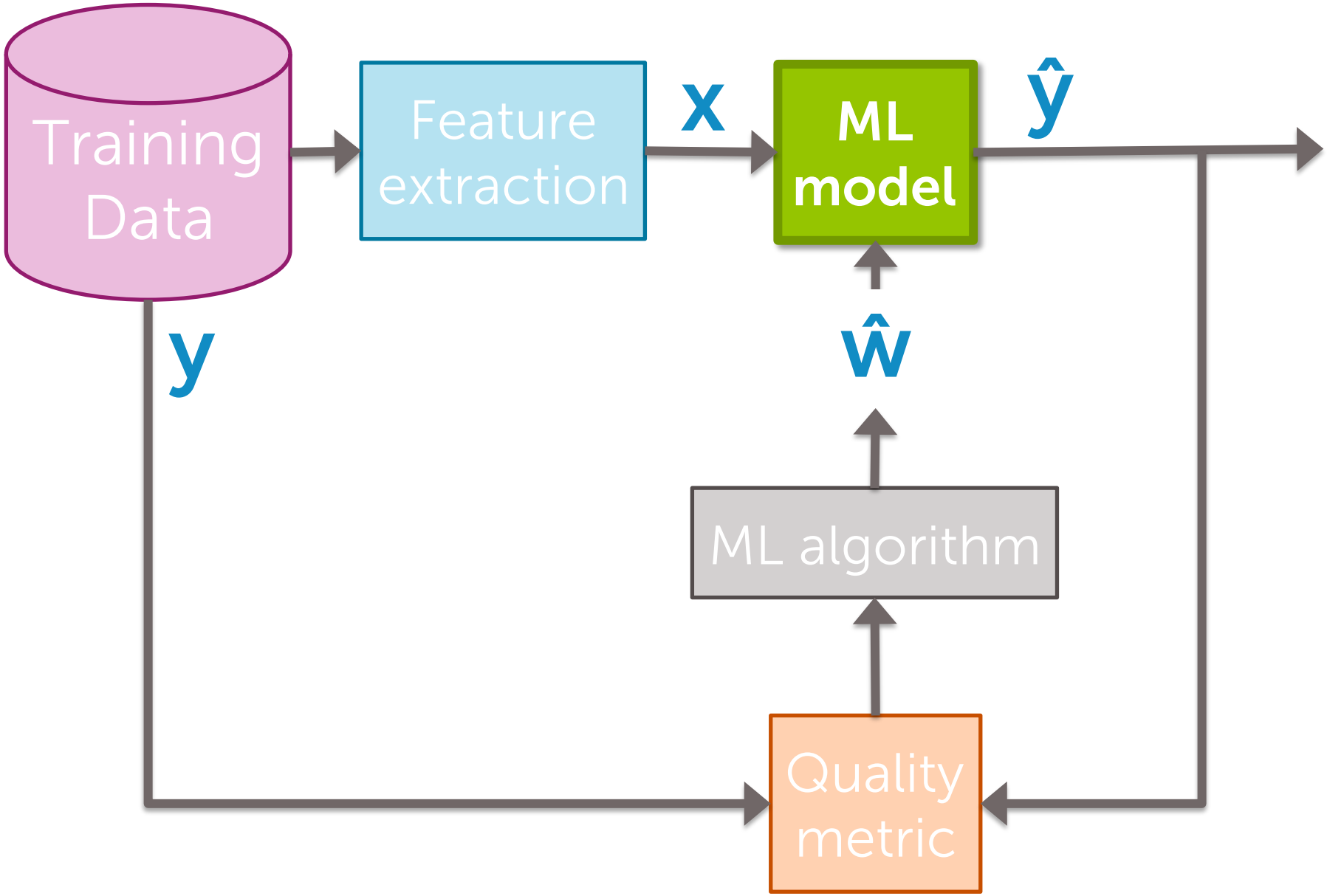
Summary for simple linear regression

What you can do now...

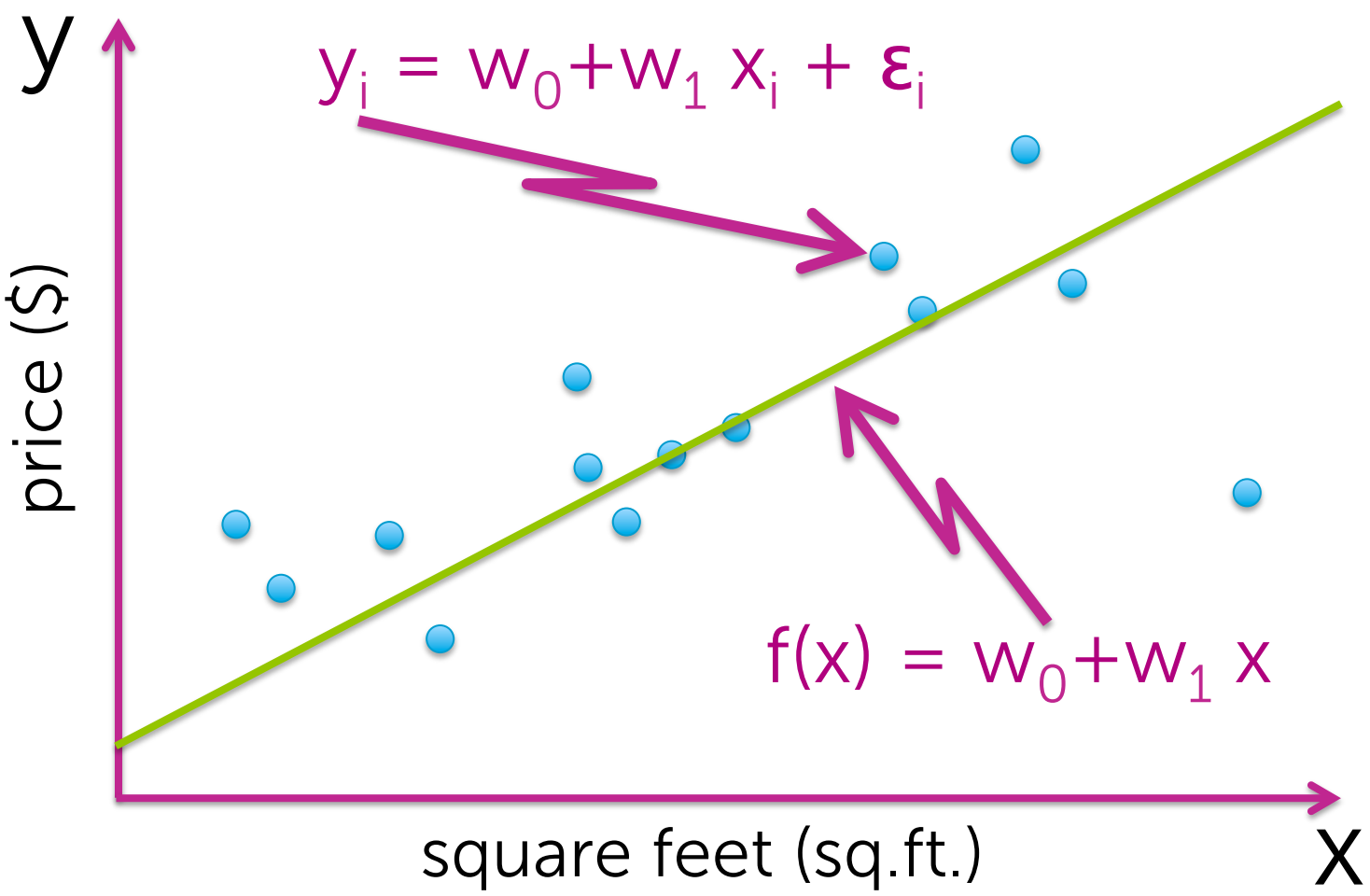
- Describe the input (features) and output (real-valued predictions) of a regression model
- Calculate a goodness-of-fit metric (e.g., RSS)
- Estimate model parameters to minimize RSS using gradient descent
- Interpret estimated model parameters
- Exploit the estimated model to form predictions
- Discuss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

Multiple Regression:

Linear regression with multiple features


Training Data
Feature extraction
x
ML model
ŷ
y
ŵ
ML algorithm
Quality metric

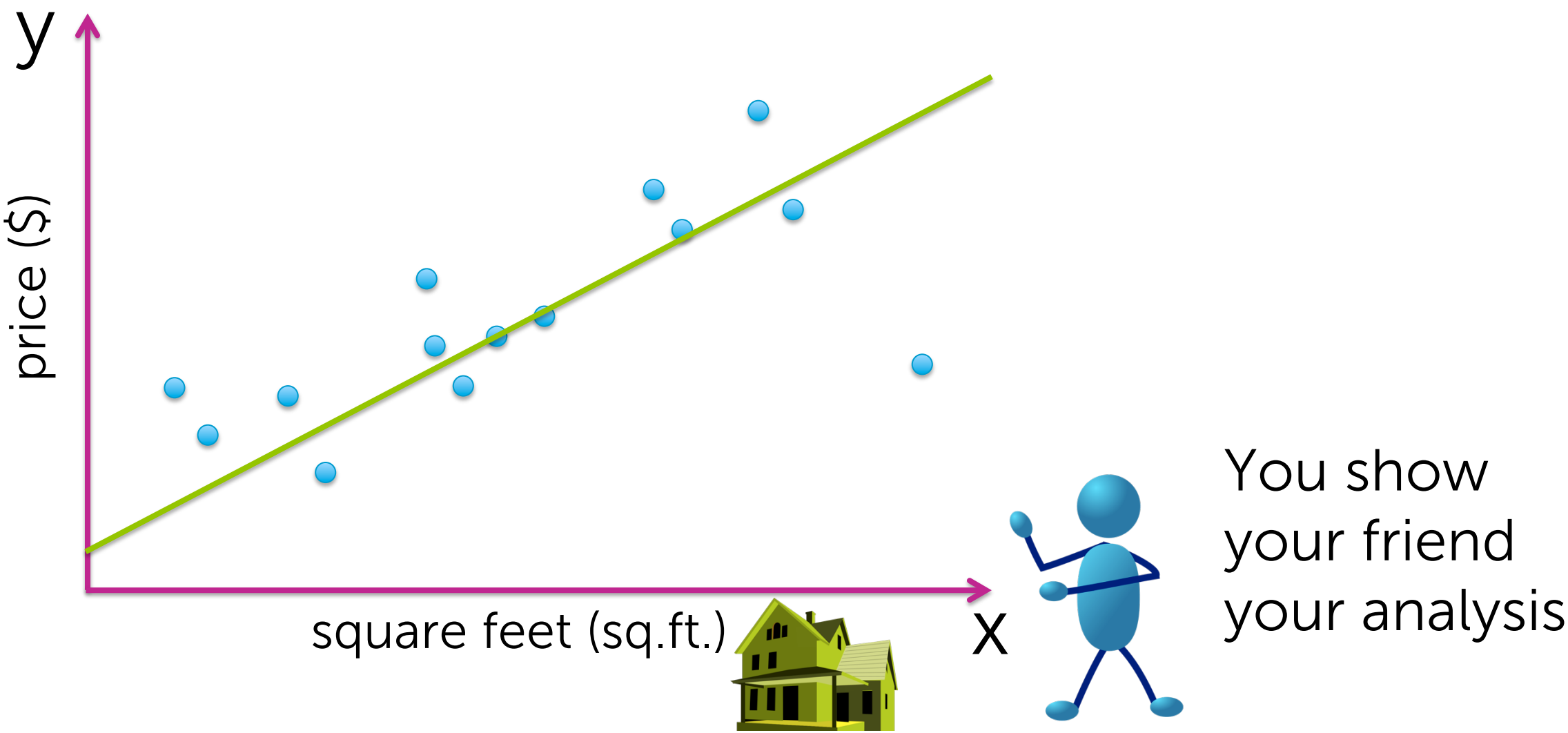
Simple linear regression model



More complex functions of a single input

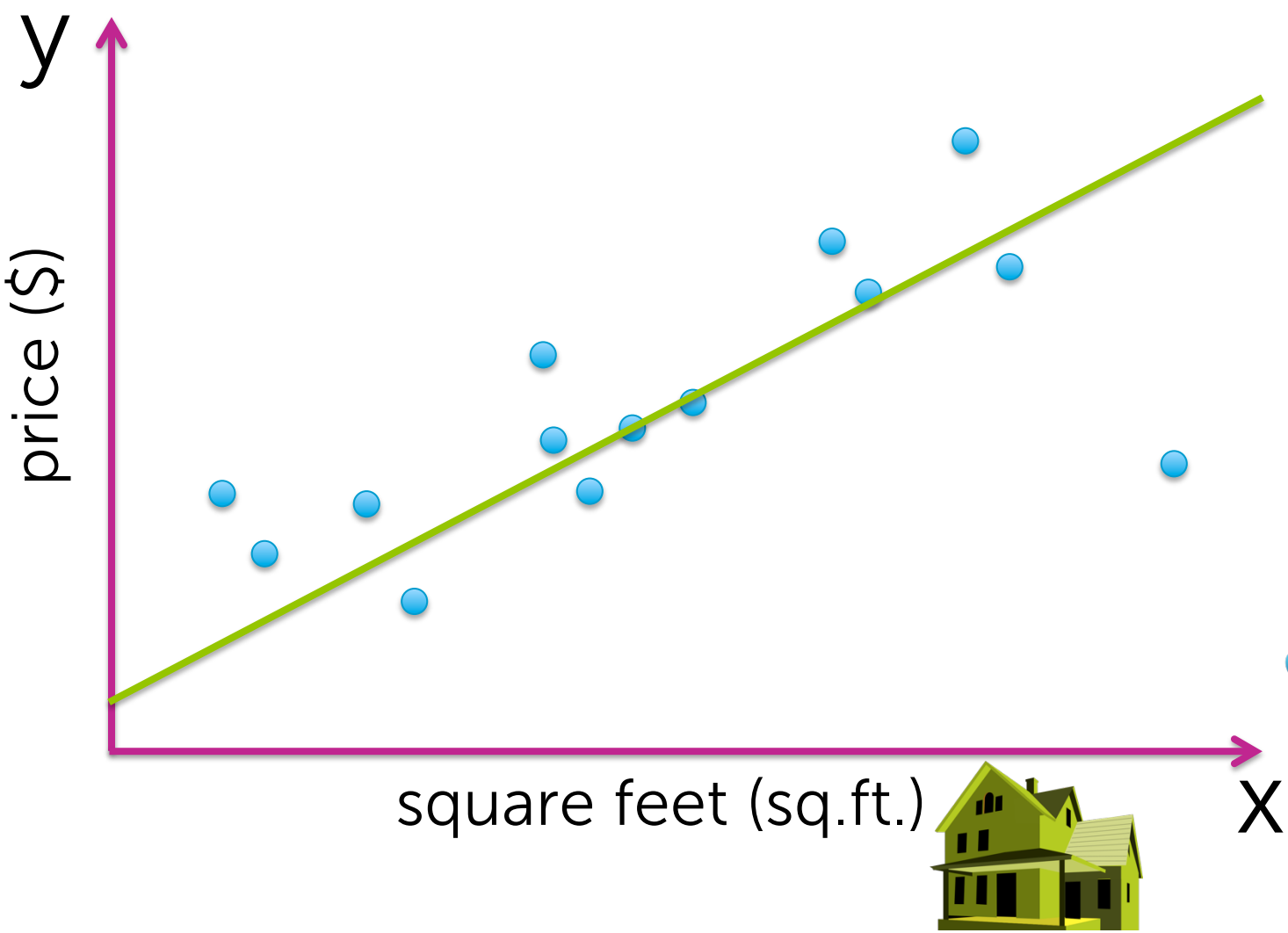
Polynomial regression

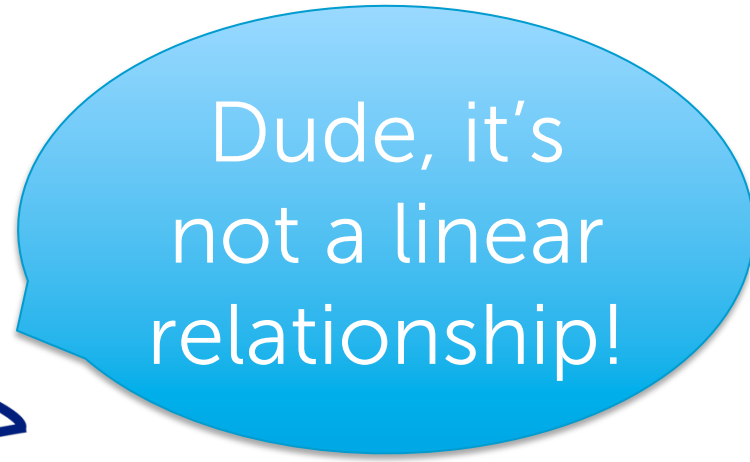
Fit data with a line or … ?



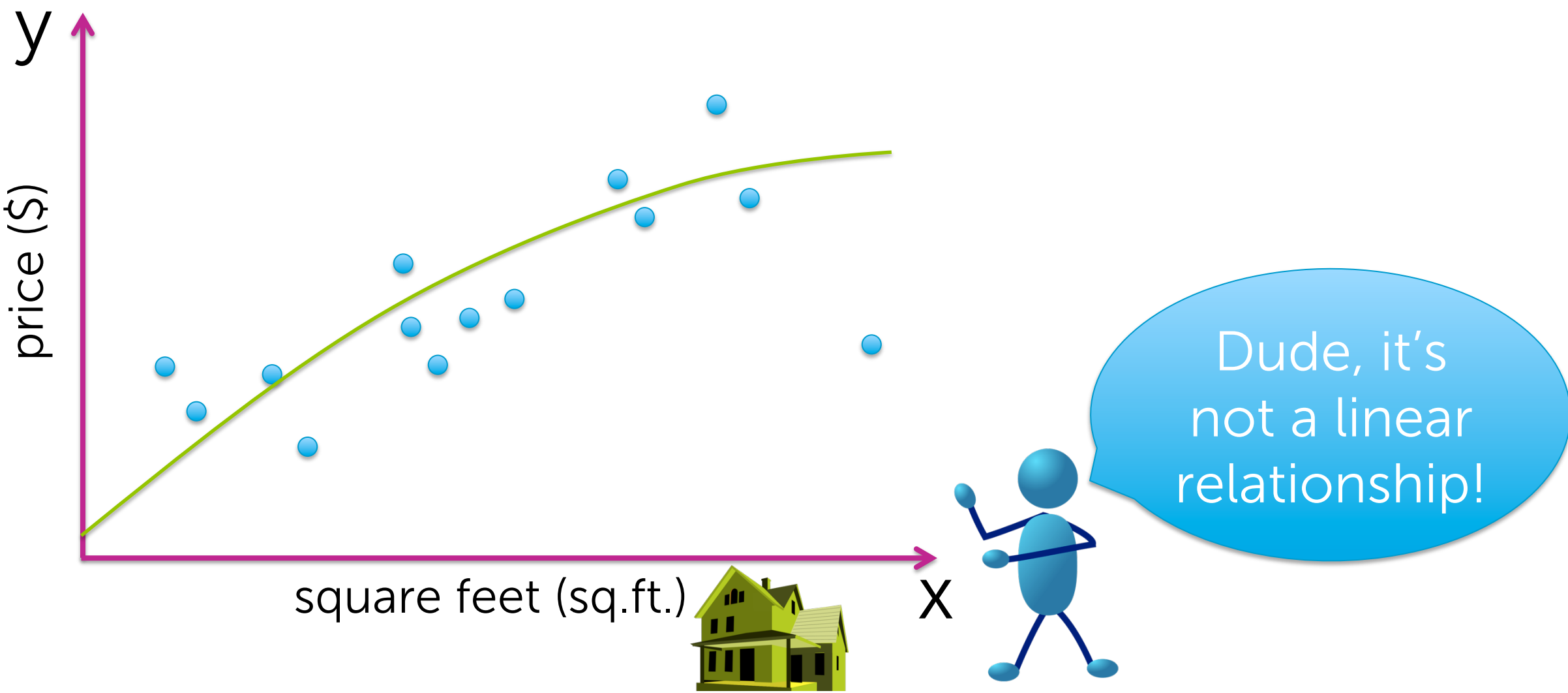
You show your friend your analysis

Fit data with a line or … ?

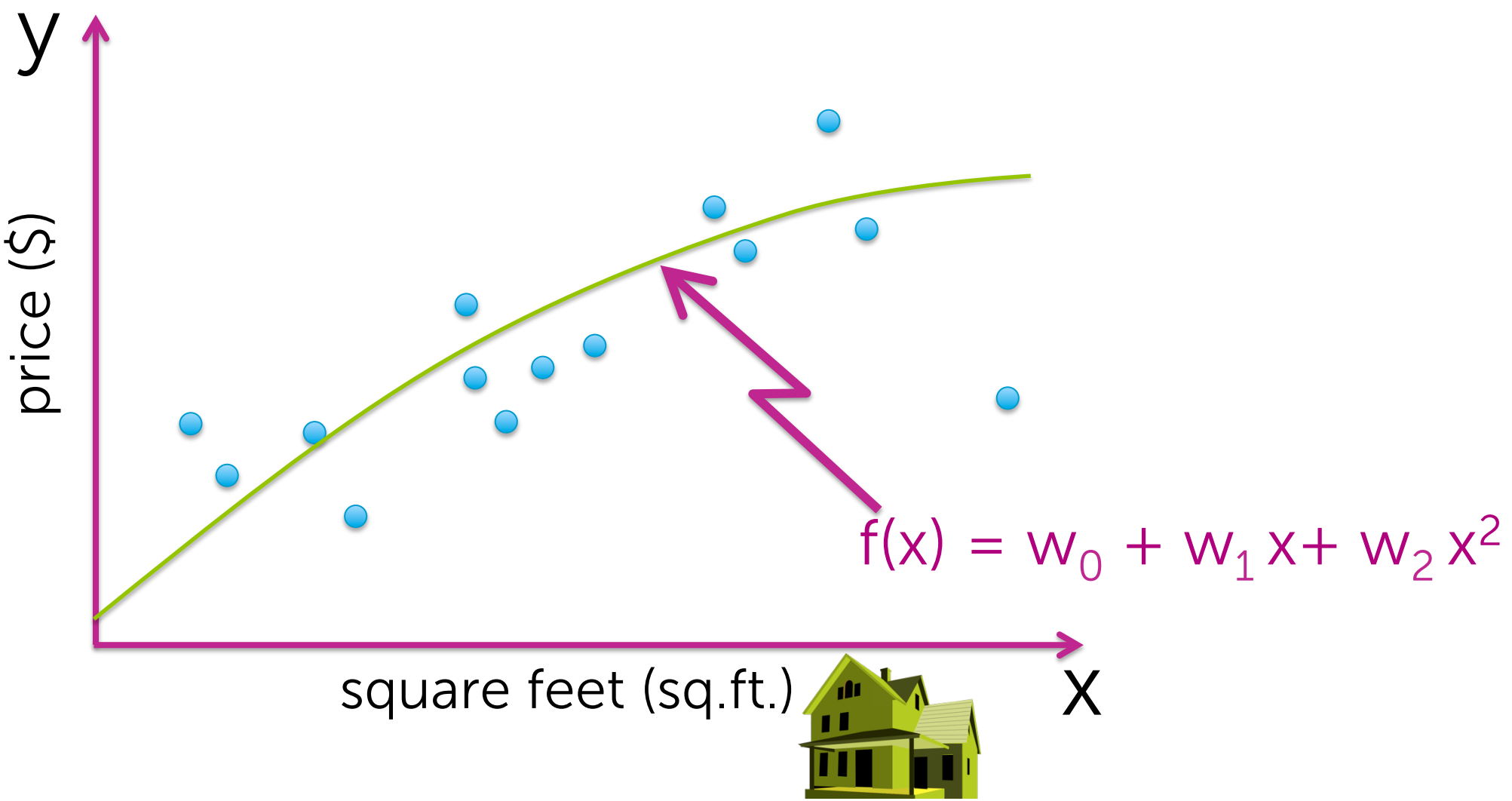




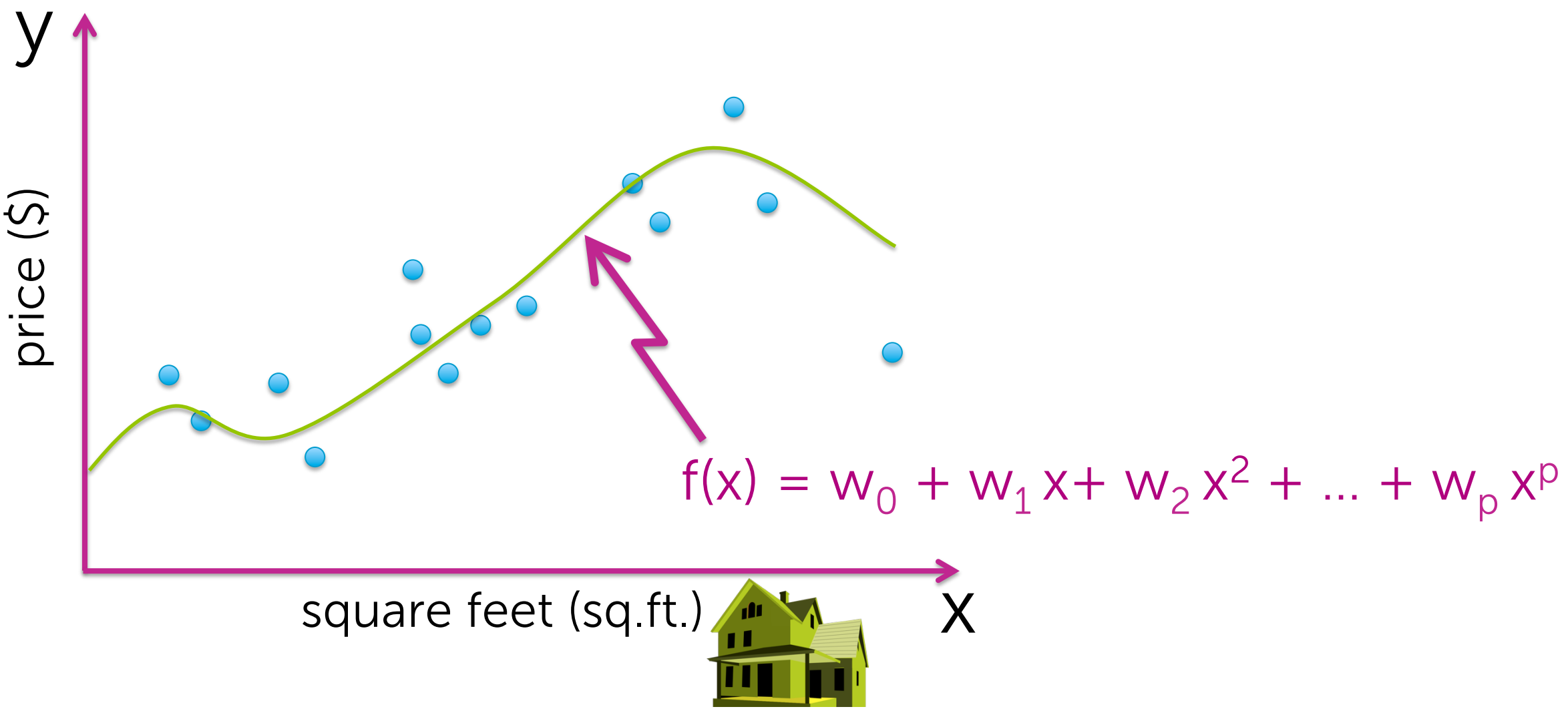
What about a quadratic function?



What about a quadratic function?



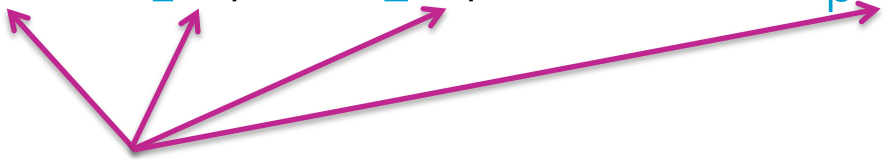
Even higher order polynomial



Polynomial regression

Model:

$$y_i = w_0 + w_1 x_i + w_2 x_i^2 + \ldots + w_p x_i^p + \varepsilon_i$$



treat as different **features**

feature 1 = 1 (constant) *parameter* 1 = w_0

feature 2 = x *parameter* 2 = w_1

feature 3 = x^2 *parameter* 3 = w_2

... ...

feature p+1 = x^p *parameter* p+1 = w_p

More generally…

Generic basis expansion

Model:

$$y_i = w_0 h_0(x_i) + w_1 h_1(x_i) + \ldots + w_D h_D(x_i) + \varepsilon_i$$

$$= \sum_{j=0}^{D} w_j h_j(x_i) + \varepsilon_i$$

j^{th} *feature*

j^{th} *regression coefficient*
or *weight*

Generic basis expansion

Model:

$$y_i = w_0 h_0(x_i) + w_1 h_1(x_i) + \ldots + w_D h_D(x_i) + \varepsilon_i$$

$$= \sum_{j=0}^{D} w_j h_j(x_i) + \varepsilon_i$$

feature 1 = $h_0(x)$…often 1 (constant)

feature 2 = $h_1(x)$… e.g., x

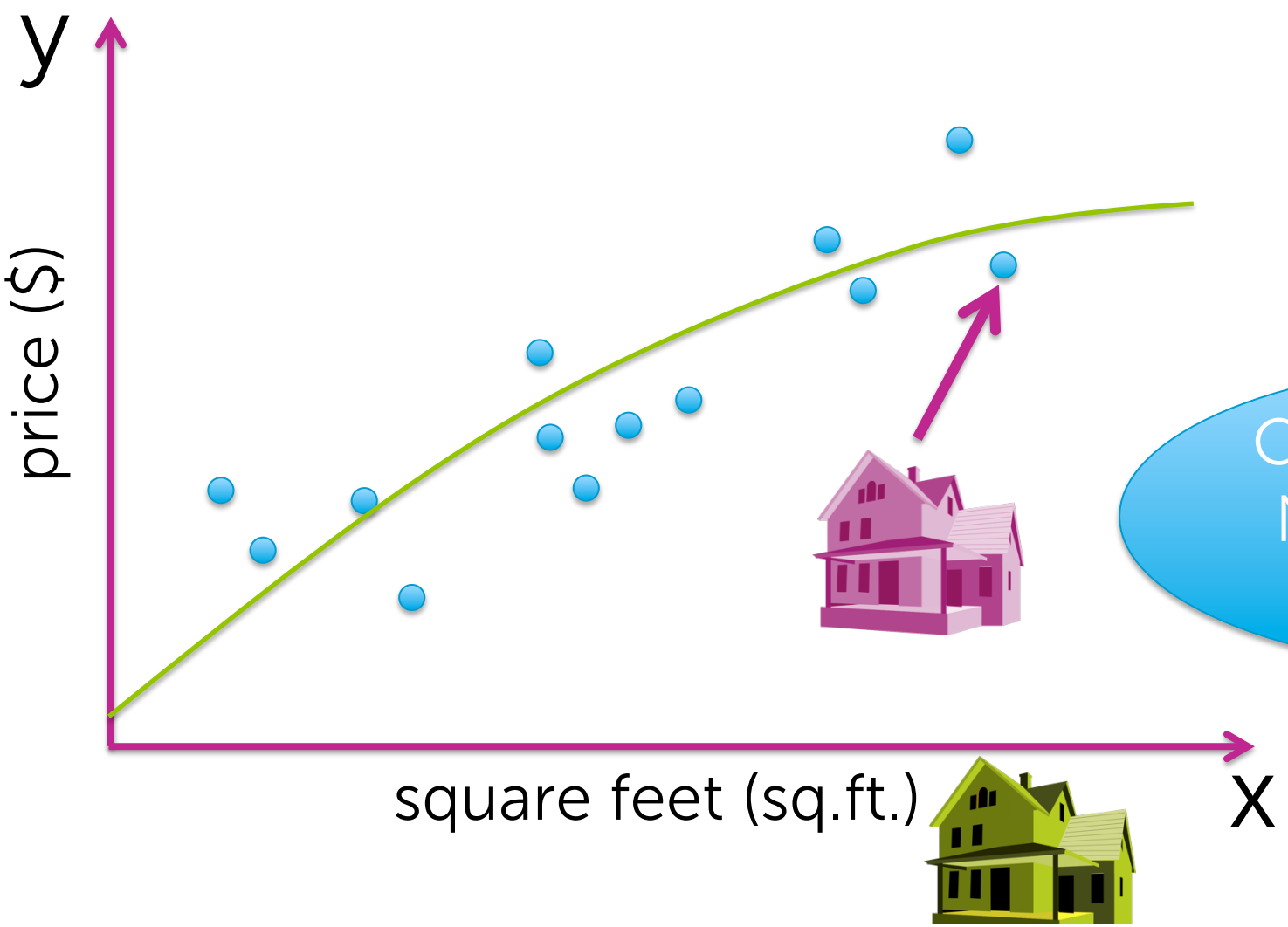
feature 3 = $h_2(x)$… e.g., x^2 or $\sin(2\pi x/12)$

…

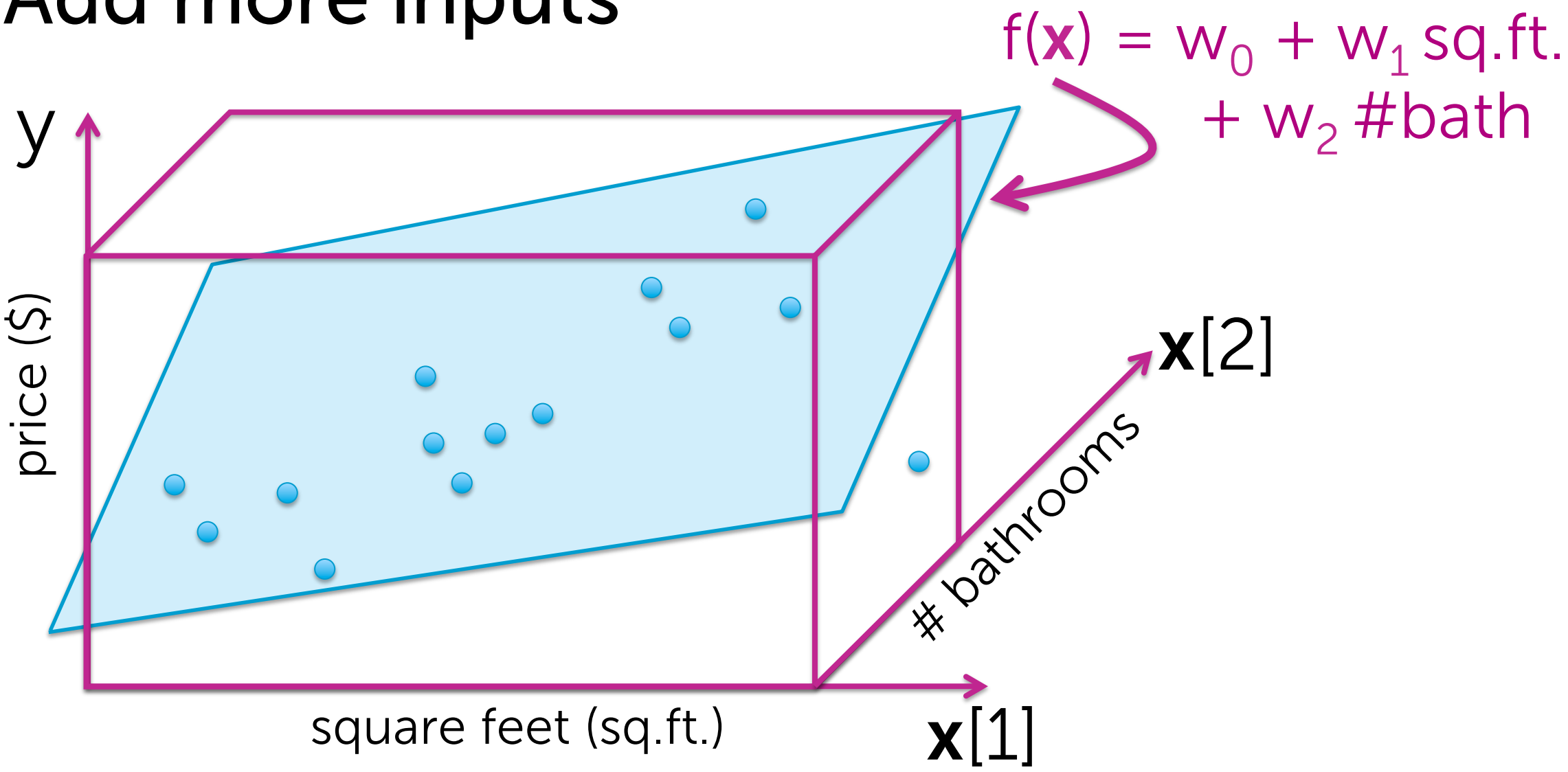
feature D+1 = $h_D(x)$… e.g., x^p

Incorporating multiple inputs

Predictions just based on house size



Add more inputs



Many possible inputs

- Square feet
- # bathrooms
- # bedrooms
- Lot size
- Year built
- …

General notation

Output: y ← scalar

Inputs: $\mathbf{x} = (\mathbf{x}[1], \mathbf{x}[2], \ldots, \mathbf{x}[d])$

↖ d-dim vector

Notational conventions:

- $\mathbf{x}[j]$ = j^{th} input (*scalar*)
- $h_j(\mathbf{x})$ = j^{th} feature (*scalar*)
- $\mathbf{x}_i$ = input of i^{th} data point (*vector*)
- $\mathbf{x}_i[j]$ = j^{th} input of i^{th} data point (*scalar*)

Simple hyperplane

Model:

$y_i = w_0 + w_1 \mathbf{x}_i[1] + \ldots + w_d \mathbf{x}_i[d] + \varepsilon_i$

feature 1 = 1

feature 2 = **x**[1] … e.g., sq. ft.

feature 3 = **x**[2] … e.g., #bath

…

feature d+1 = **x**[d] … e.g., lot size

More generically…
D-dimensional curve

Model:

$$y_i = w_0\, h_0(\mathbf{x}_i) + w_1\, h_1(\mathbf{x}_i) + \ldots + w_D\, h_D(\mathbf{x}_i) + \varepsilon_i$$

$$= \sum_{j=0}^{D} w_j\, h_j(\mathbf{x}_i) + \varepsilon_i$$

feature 1 = $h_0(\mathbf{x})$ … e.g., 1
feature 2 = $h_1(\mathbf{x})$ … e.g., $\mathbf{x}$[1] = sq. ft.
feature 3 = $h_2(\mathbf{x})$ … e.g., $\mathbf{x}$[2] = #bath
or, log($\mathbf{x}$[7]) $\mathbf{x}$[2] = log(#bed) x #bath

…
feature D+1 = $h_D(\mathbf{x})$ … some other function of $\mathbf{x}$[1],…, $\mathbf{x}$[d]

More on notation

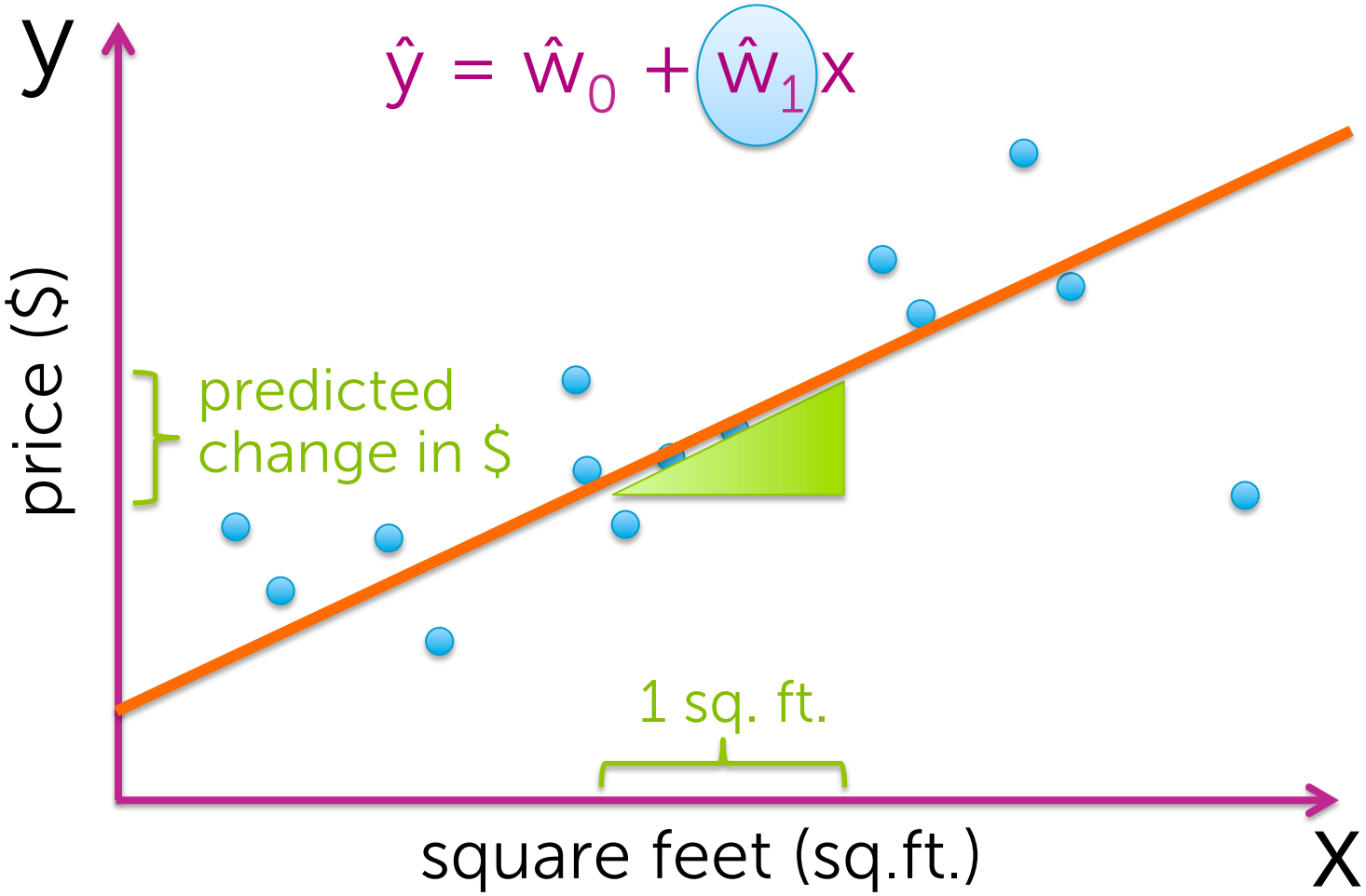
observations $(\mathbf{x}_i, y_i)$: N

inputs $\mathbf{x}[j]$: d

features $h_j(\mathbf{x})$: D

Interpreting the fitted function

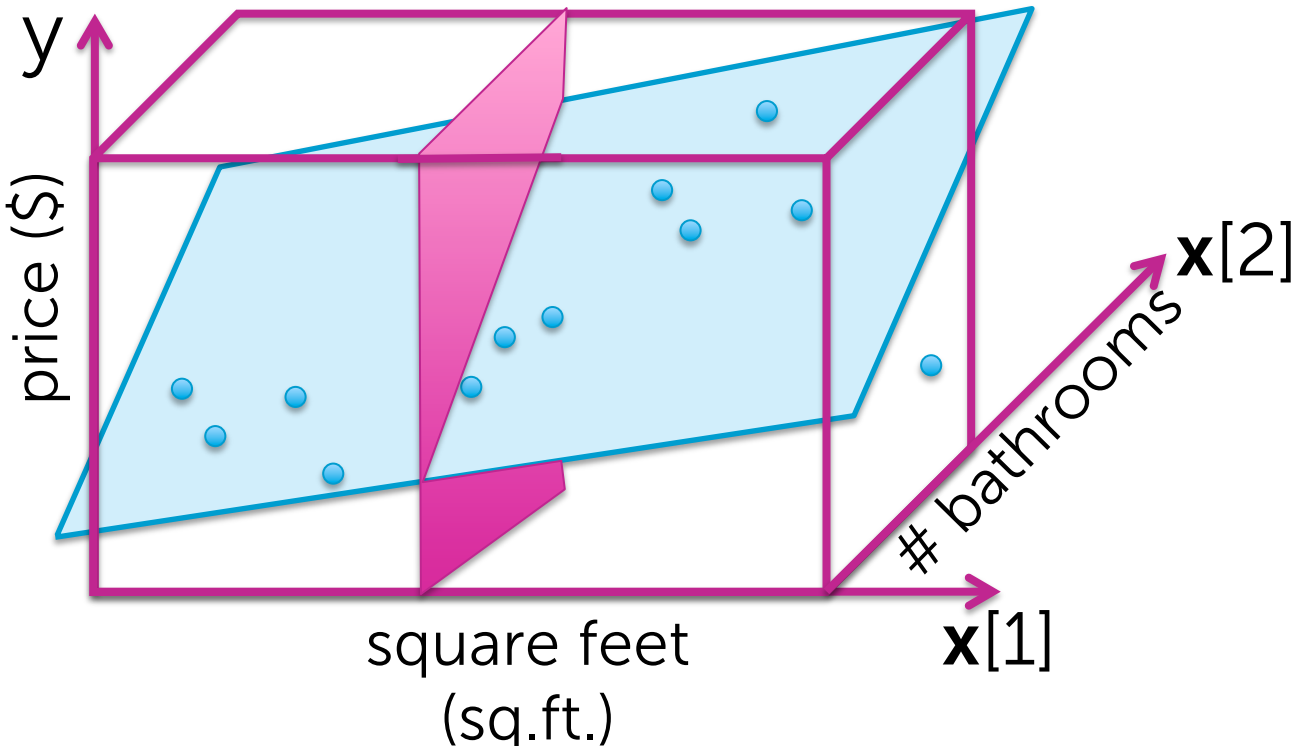
Interpreting the coefficients – Simple linear regression



Interpreting the coefficients – Two linear features

$$\hat{y} = \hat{w}_0 + \hat{w}_1 \mathbf{x}[1] + \hat{w}_2 \mathbf{x}[2]$$

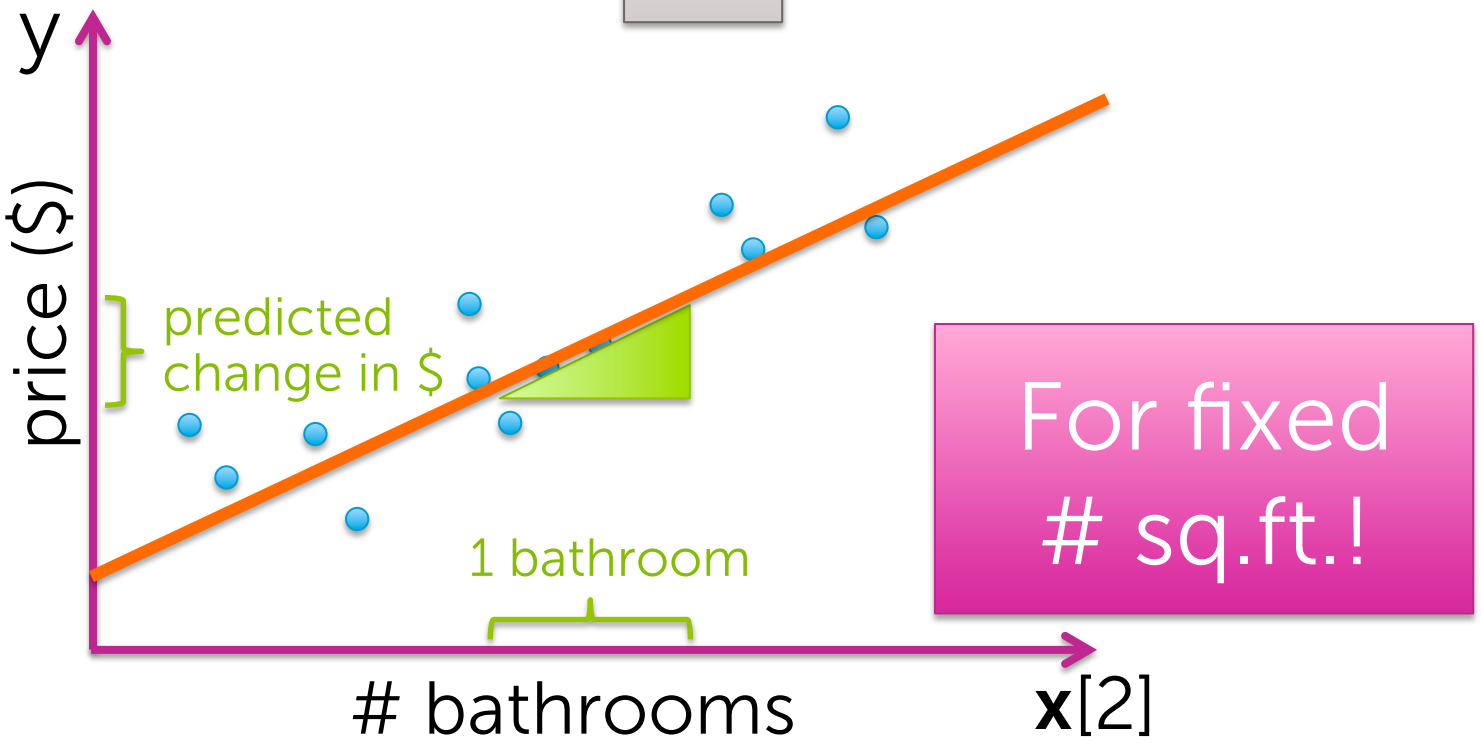
fix



Interpreting the coefficients – Two linear features

$$\hat{y} = \hat{w}_0 + \hat{w}_1 \mathbf{x}[1] + \hat{w}_2 \mathbf{x}[2]$$

fix

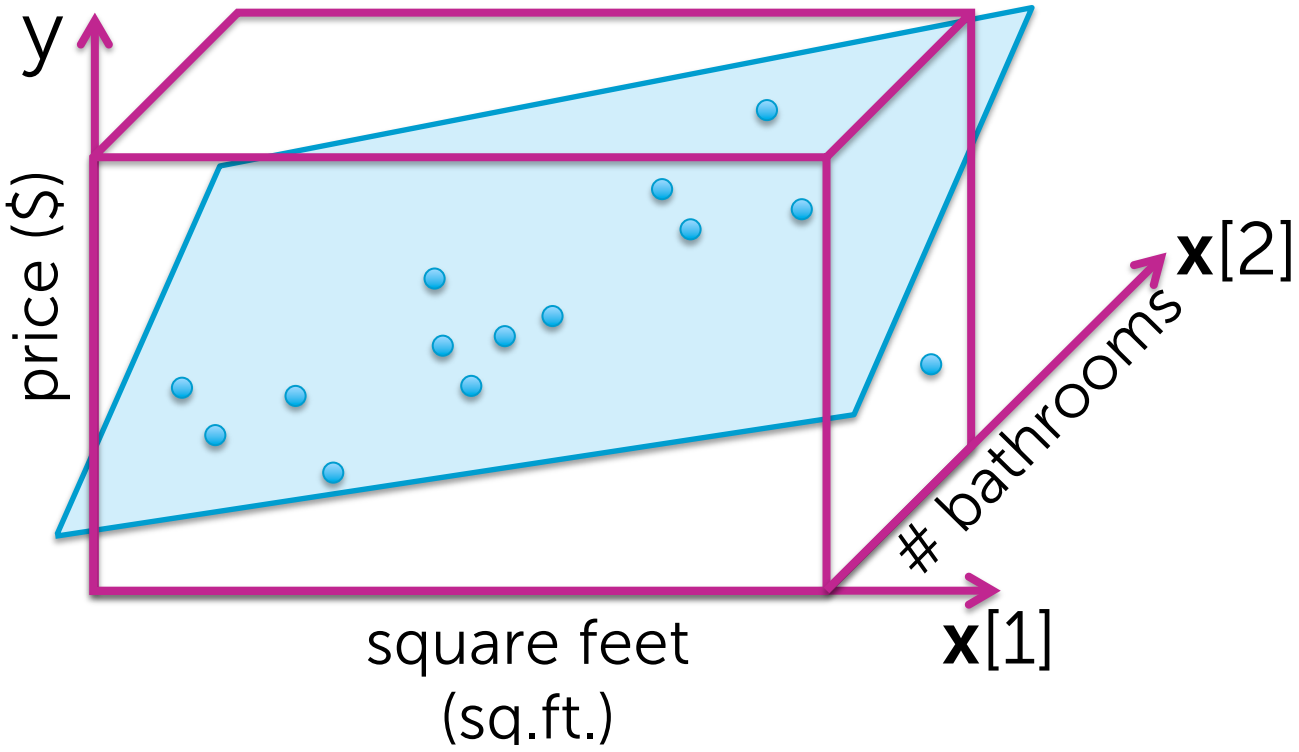


For fixed # sq.ft.!

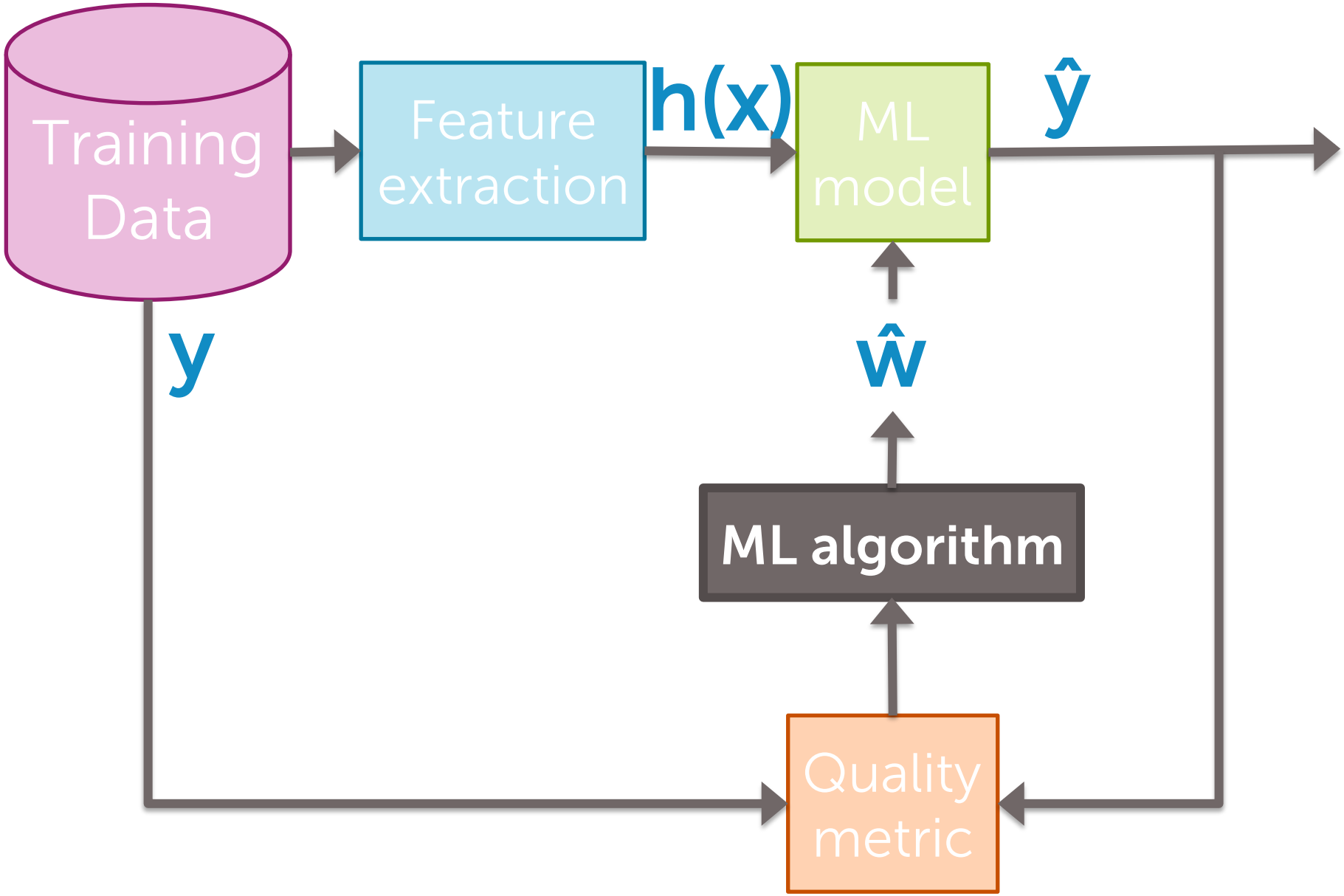
Interpreting the coefficients – Multiple linear features

$$\hat{y} = \hat{w}_0 + \hat{w}_1 \mathbf{x}[1] + \ldots + \hat{w}_j \mathbf{x}[j] + \ldots + \hat{w}_d \mathbf{x}[d]$$

fix fix fix fix



Fitting D-dimensional curves

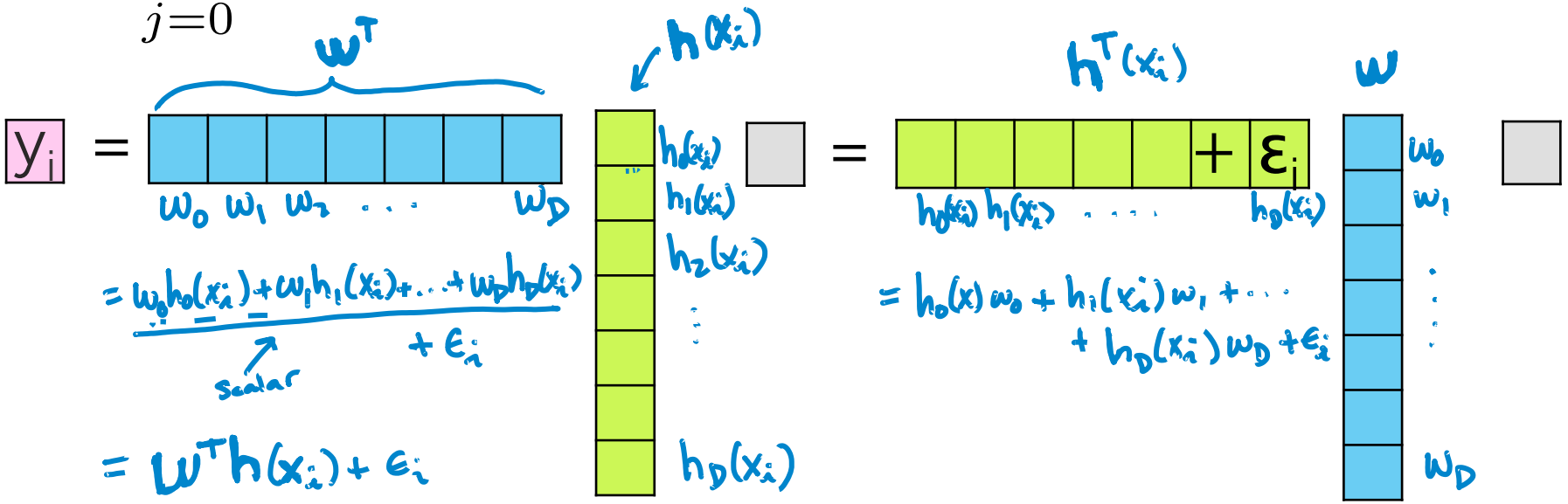

Training Data
Feature extraction
h(x)
ML model
ŷ
y
ŵ
ML algorithm
Quality metric

Step 1:
Rewrite the regression model

Rewrite in matrix notation

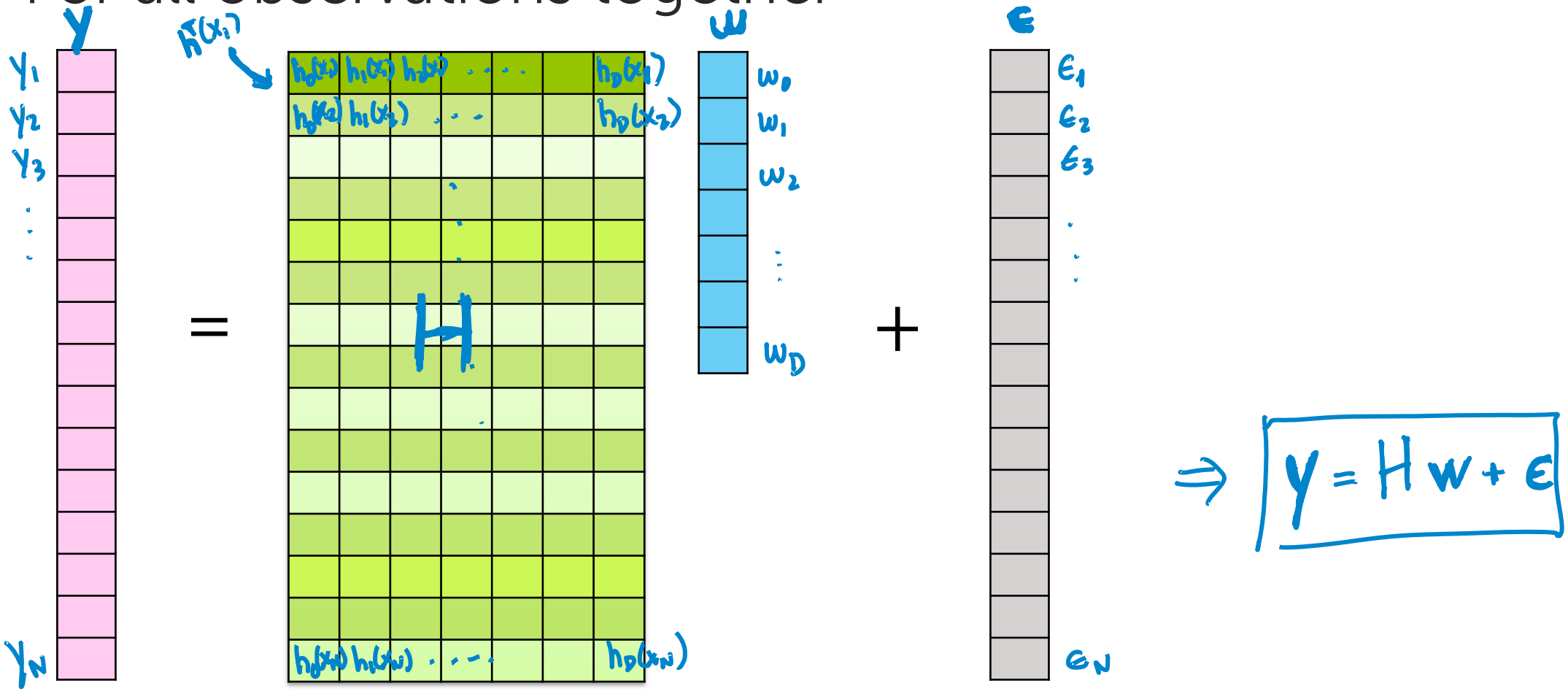
For observation i

$$y_i = \sum_{j=0}^{D} w_j \, h_j(\mathbf{x}_i) + \varepsilon_i$$



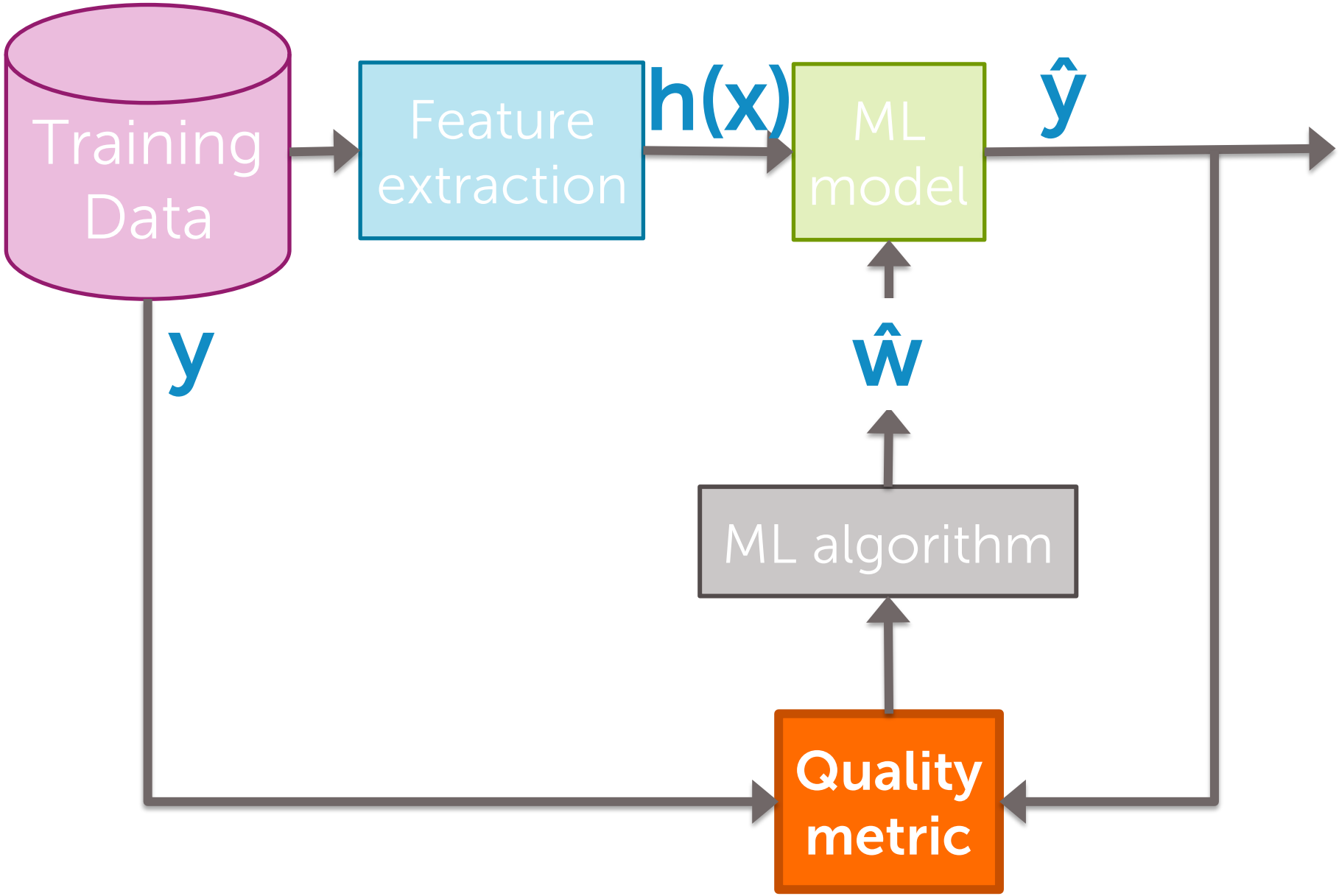
Rewrite in matrix notation

For all observations together

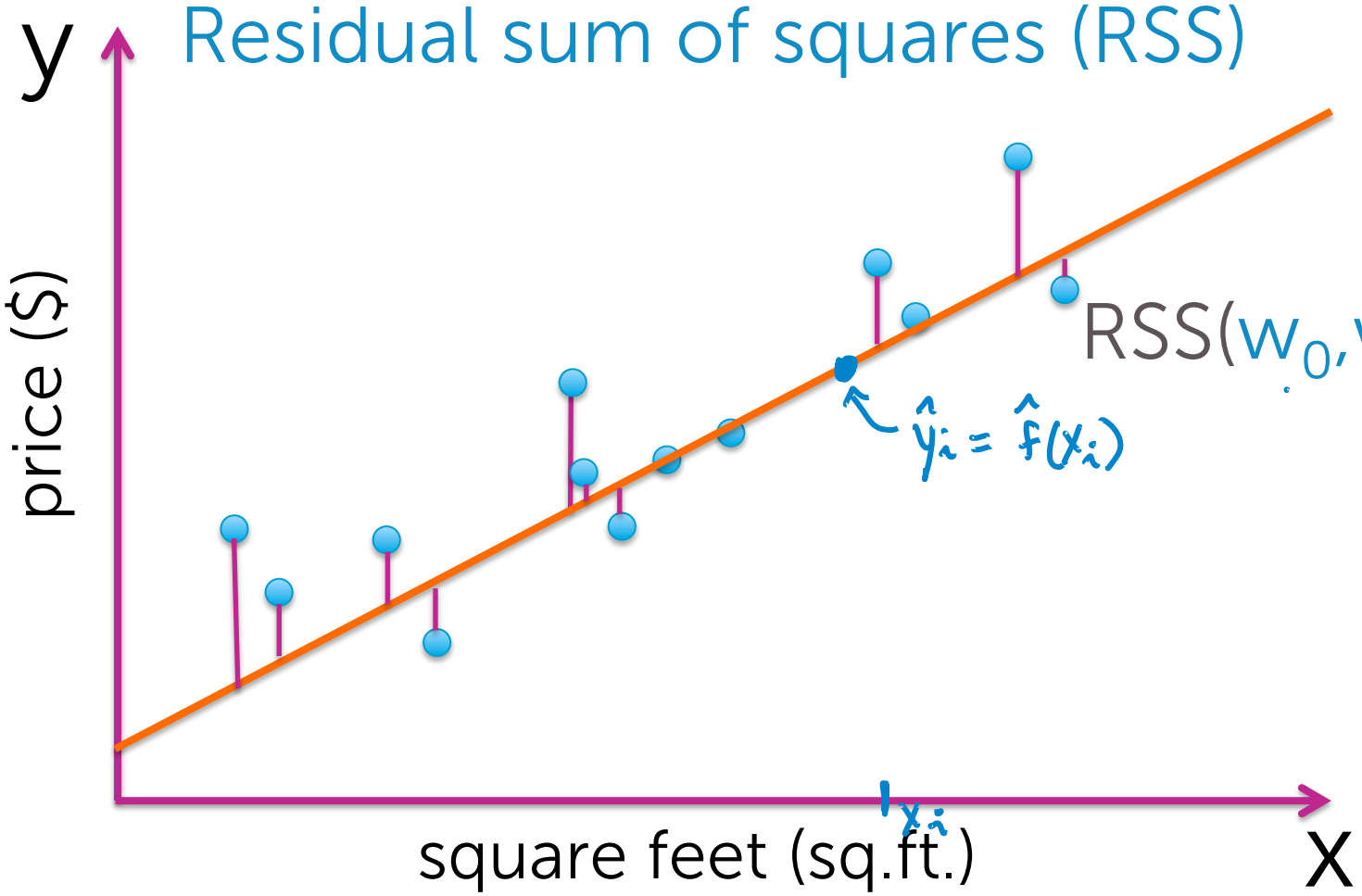


$$\Rightarrow \boxed{y = Hw + \epsilon}$$

Step 2:
Compute the cost

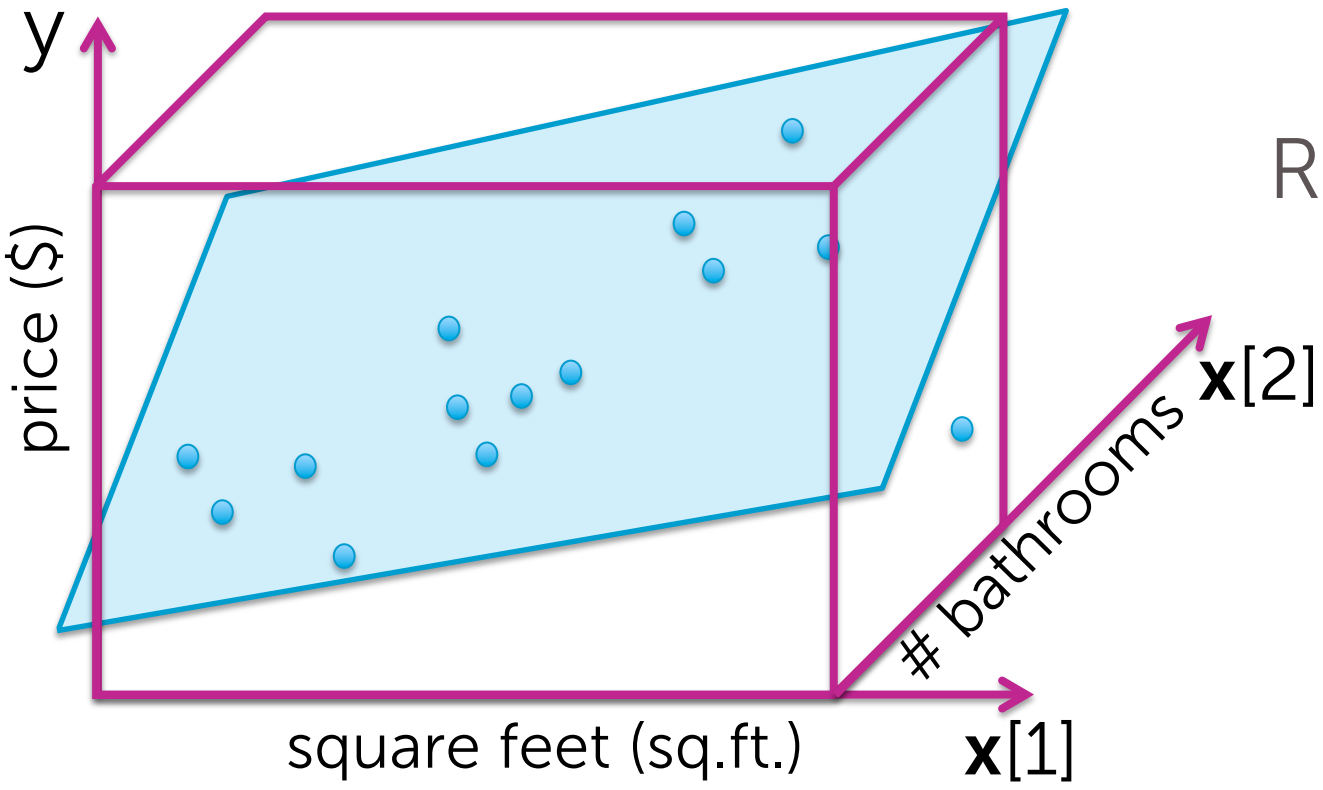

Training Data
Feature extraction
h(x)
ML model
ŷ
y
ŵ
ML algorithm
Quality metric

"Cost" of using a given line



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

RSS for multiple regression



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

$h^T(x_i)\,\mathbf{w} = \hat{y}_i(\mathbf{w})$

$$\hat{y}_i = \begin{bmatrix} h_0(x_i) & h_1(x_i) & \cdots & h_D(x_i) \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_D \end{bmatrix}$$

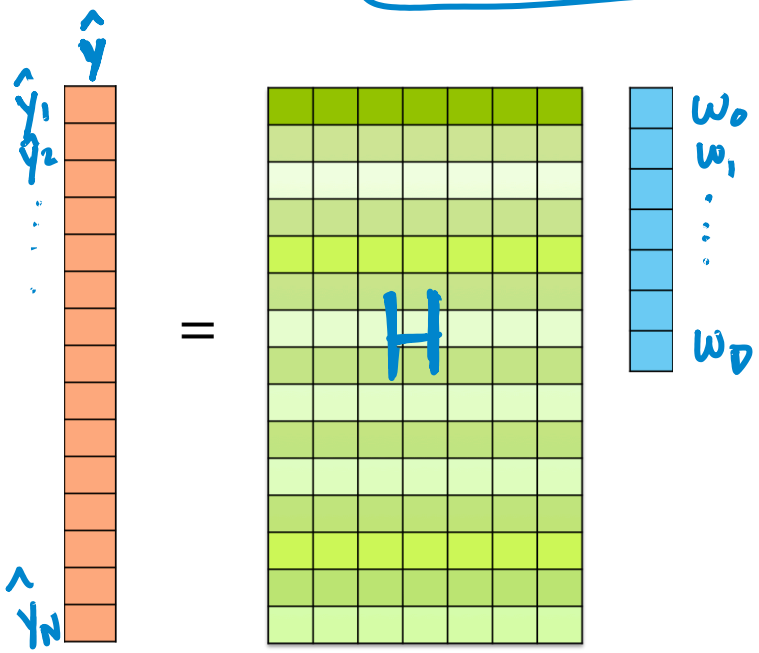
$h^T(x_i)$ $\quad$ $\mathbf{w}$

RSS in matrix notation

$$\text{RSS}(\mathbf{w}) = \sum_{i=1}^{N} (y_i - h(\mathbf{x}_i)^T \mathbf{w})^2$$

$$= (y - Hw)^T (y - Hw)$$

Why? (part 1)



$$\hat{y} = Hw$$

$$(y - Hw) = (y - \hat{y}) = \begin{bmatrix} \text{residual}_1 \\ \text{residual}_2 \\ \vdots \\ \text{residual}_N \end{bmatrix}$$

$\text{residual}_i = y_i - \hat{y}_i$

RSS in matrix notation

$$\text{RSS}(\mathbf{w}) = \sum_{i=1}^{N} (y_i - h(\mathbf{x}_i)^T \mathbf{w})^2$$

$$= (\mathbf{y} - \mathbf{H}\mathbf{w})^T (\mathbf{y} - \mathbf{H}\mathbf{w})$$

Why? (part 2)

residual_1	residual_2	residual_3	...	residual_N

residual_1
residual_2
residual_3
...
residual_N

$=$

$$(\text{residual}_1^2 + \text{residual}_2^2 + \cdots + \text{residual}_N^2)$$

$$= \sum_{i=1}^{N} \text{residual}_i^2$$

$$\triangleq \text{RSS}(\mathbf{w})$$

Step 3:
Take the gradient

Gradient of RSS

$$\nabla \text{RSS}(\mathbf{w}) = \nabla[(\mathbf{y}-\mathbf{H}\mathbf{w})^T(\mathbf{y}-\mathbf{H}\mathbf{w})]$$

$$= -2\mathbf{H}^T(\mathbf{y}-\mathbf{H}\mathbf{w})$$

Why? By analogy to 1D case:

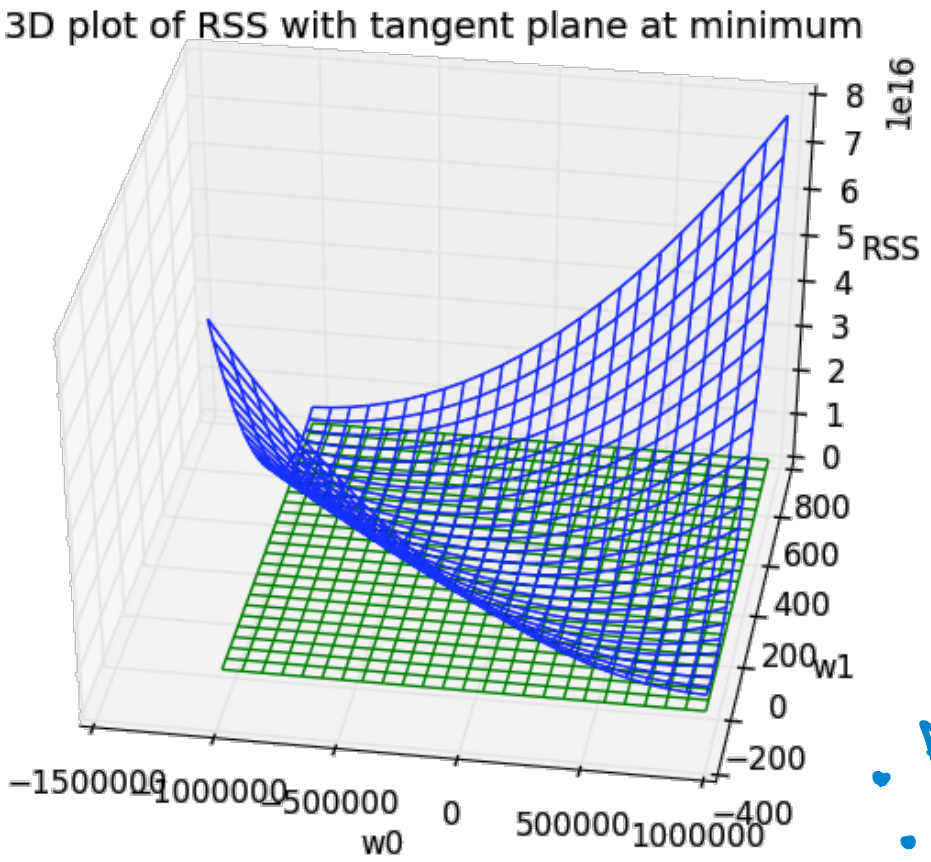
$$\frac{d}{dw}(y-hw)(y-hw) = \frac{d}{dw}(y-hw)^2 = 2\cdot(y-hw)^1(-h)$$

$$= -2h(y-hw)$$

scalars

Step 4, Approach 1:
Set the gradient = 0

Closed-form solution



$$\nabla \mathrm{RSS}(\mathbf{w}) = -2\mathbf{H}^T(\mathbf{y}-\mathbf{H}\mathbf{w}) = 0$$

Solve for $\mathbf{w}$:

$$-2H^Ty + 2H^TH\hat{w} = 0$$

$$H^TH\hat{w} = H^Ty$$

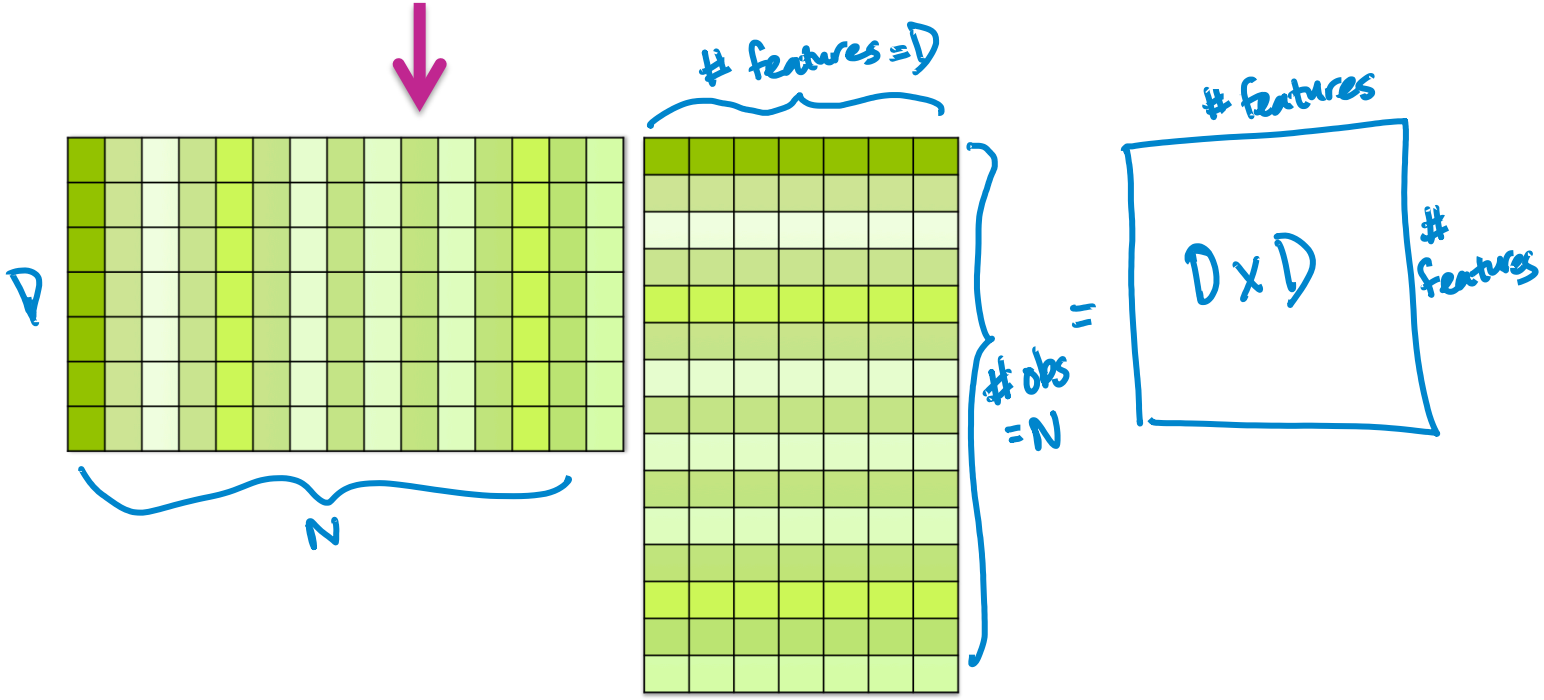
$$(H^TH)^{-1}H^TH\hat{w} = (H^TH)^{-1}H^Ty$$

$$\hat{w} = (H^TH)^{-1}H^Ty$$

- $A^{-1}A = I$
- $Iv = v$
- $IV = V$

Closed-form solution

$$\hat{\mathbf{w}} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{y}$$



really, # of linearly ind. observations

Invertible if: $N > D$

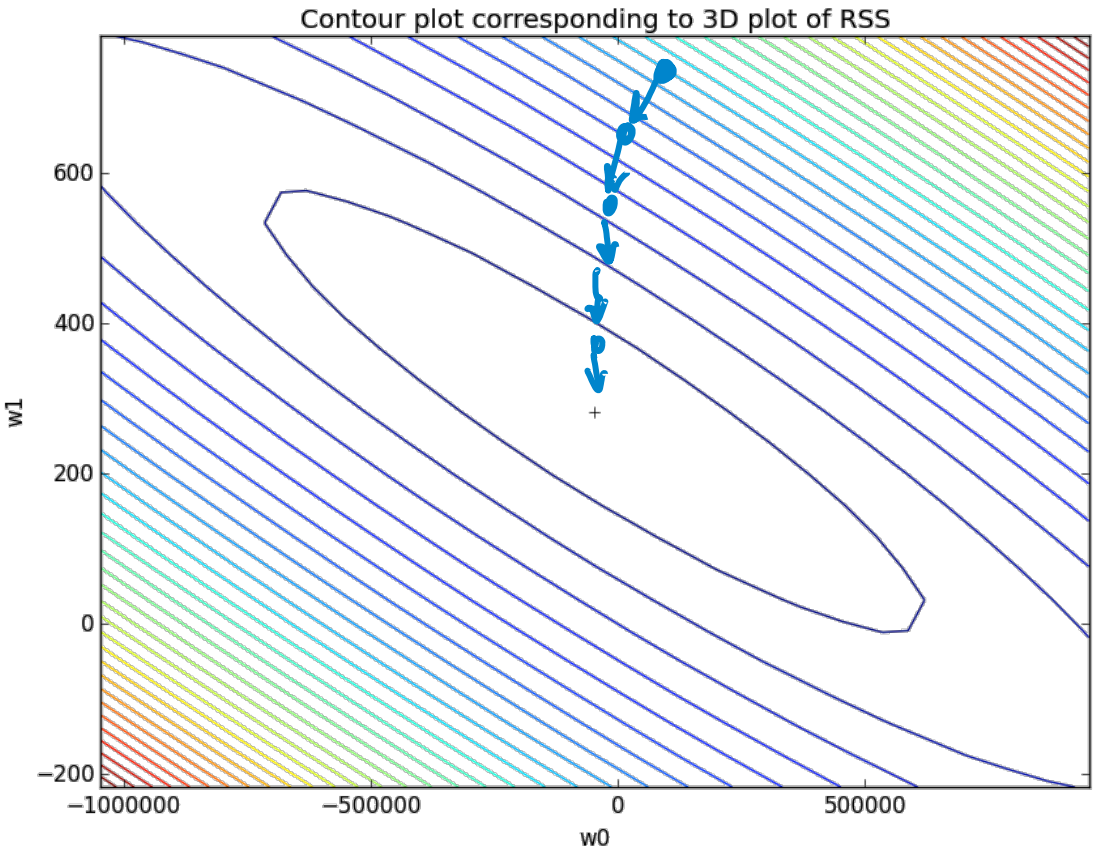
In most cases is

Complexity of inverse:

$$O(D^3)$$

Step 4, Approach 2:
Gradient descent

Gradient descent



while not converged

$$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \eta \underbrace{\nabla \mathrm{RSS}(\mathbf{w}^{(t)})}_{-2\mathbf{H}^T(\mathbf{y}-\mathbf{H}\mathbf{w})}$$

$$\leftarrow w^{(t)} + 2\eta H^T (y - \underbrace{Hw^{(t)}}_{\hat{y}(w^{(t)})})$$

Feature-by-feature update

$$RSS(\mathbf{w}) = \sum_{i=1}^{N} (y_i - h(\mathbf{x}_i)^T\mathbf{w})^2$$

$$= \sum_{i=1}^{N} (y_i - w_0 h_0(x_i) - w_1 h_1(x_i) - \cdots - w_D h_D(x_i))^2$$

Partial with respect to w_j

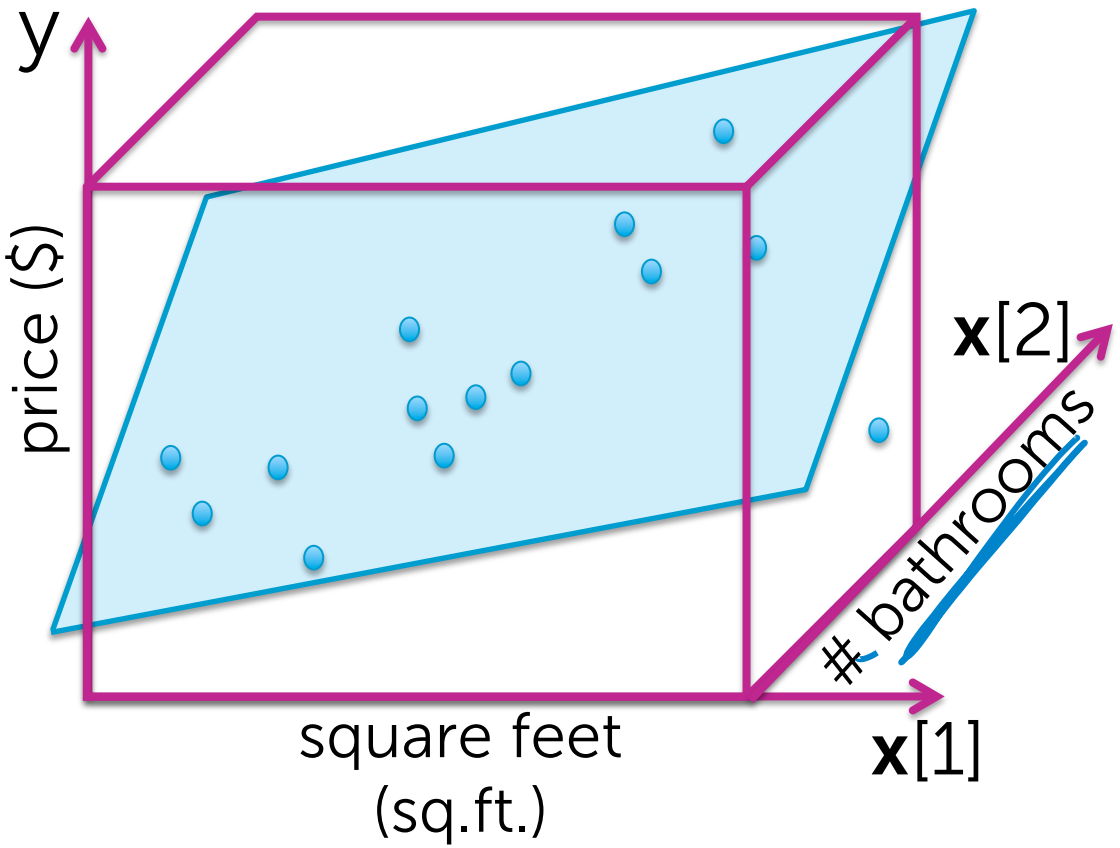
$$\sum_{i=1}^{N} 2(y_i - w_0 h_0(x_i) - w_1 h_1(x_i) \cdots - w_D h_D(x_i))^1 \cdot (-h_j(x_i))$$

$$= -2\sum_{i=1}^{N} h_j(x_i)(y_i - h(x_i)^T w)$$

Update to j^{th} feature weight:

$$w_j^{(t+1)} \leftarrow w_j^{(t)} - \eta\left(-2\sum_{i=1}^{N} h_j(x_i)\underbrace{(y_i - h^T(x_i)w^{(t)})}_{\hat{y}_i(w^{(t)})}\right)$$

Interpreting elementwise



Update to j^{th} feature weight:

$$w_j^{(t+1)} \leftarrow w_j^{(t)} + 2\eta \sum_{i=1}^{N} h_j(\mathbf{x}_i)(y_i - \hat{y}_i(\mathbf{w}^{(t)}))$$

bathrooms

bath for house i

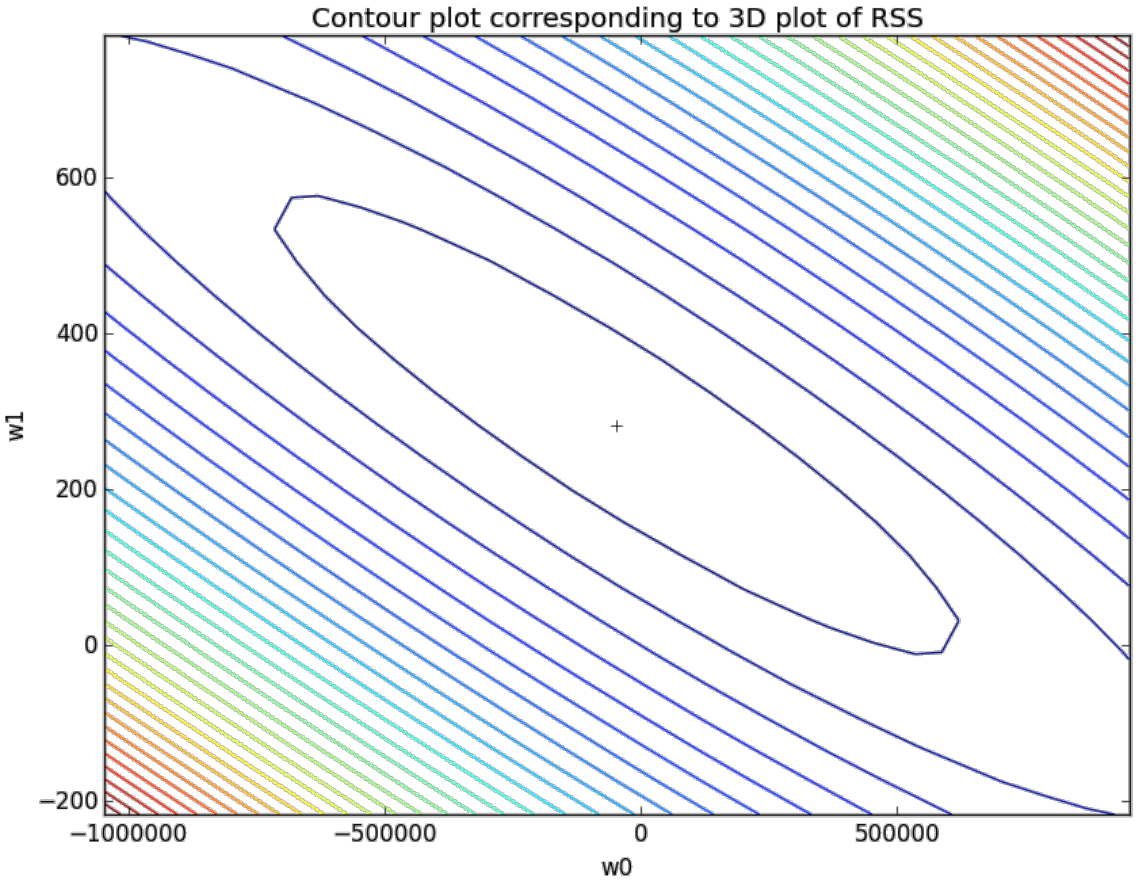
If underestimating impact of # bath ($\hat{w}_j^{(t)}$ is too small)

then $(y_i - \hat{y}_i(w^{(t)}))$ on average

weighted by # bath will be positive

$\Rightarrow w_j^{(t+1)} > w_j^{(t)}$ (increase)

Summary of gradient descent for multiple regression



init $\mathbf{w}^{(1)}=0$ (or randomly, or smartly), $t=1$

while $\|\nabla RSS(\mathbf{w}^{(t)})\| > \varepsilon$ ← tolerance

  for $j=0,\ldots,D$

  $partial[j] = -2\sum_{i=1}^{N} h_j(\mathbf{x}_i)(y_i - \hat{y}_i(\mathbf{w}^{(t)}))$

  $\mathbf{w}_j^{(t+1)} \leftarrow \mathbf{w}_j^{(t)} - \eta\, partial[j]$

  $t \leftarrow t + 1$

An extremely useful algorithm

Summary for multiple linear regression

What you can do now...

- Describe polynomial regression
- Detrend a time series using trend and seasonal components
- Write a regression model using multiple inputs or features thereof
- Cast both polynomial regression and regression with multiple inputs as regression with multiple features
- Calculate a goodness-of-fit metric (e.g., RSS)
- Estimate model parameters of a general multiple regression model to minimize RSS:
  - In closed form
  - Using an iterative gradient descent algorithm
- Interpret the coefficients of a non-featurized multiple regression fit
- Exploit the estimated model to form predictions
- Explain applications of multiple regression beyond house price modeling