Regression For Machine learning

Source: University of Washington

Linear regression with one input

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 = sq.ft., y_1 = \$)

(x_2 = sq.ft., y_2 = \$)

(x_3 = sq.ft., y_3 = \$)

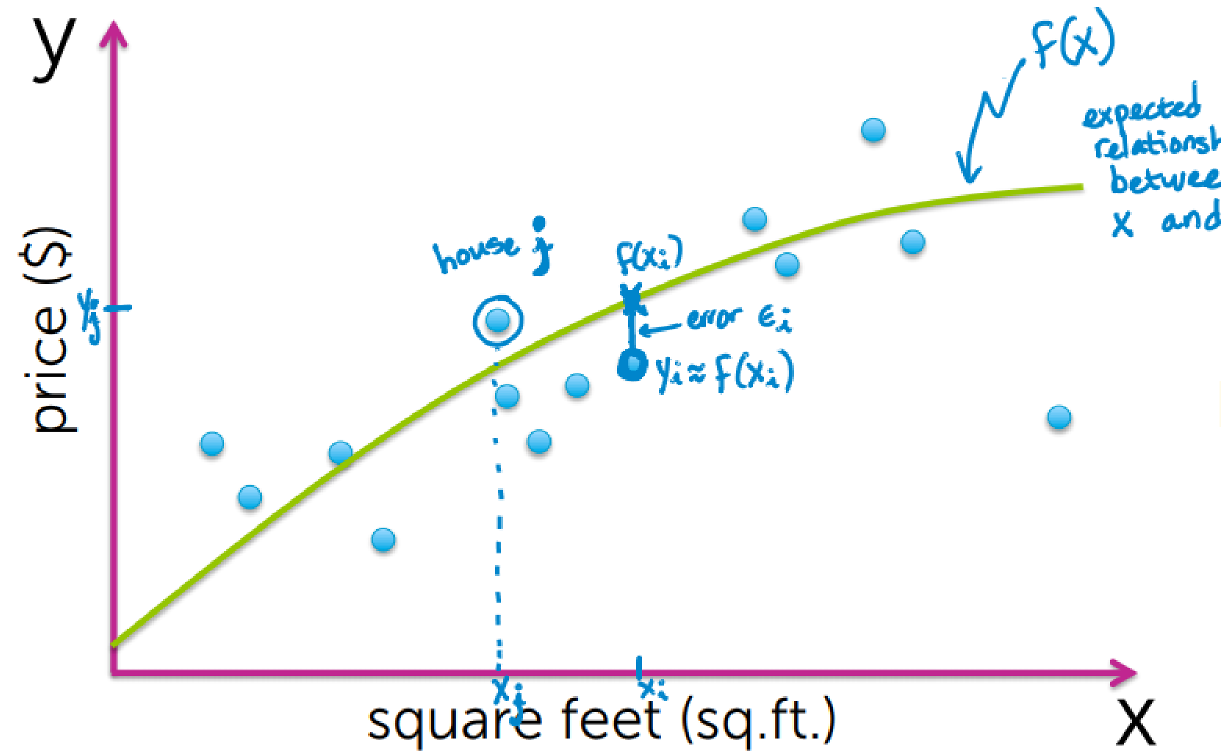
(x_4 = sq.ft., y_4 = \$)

(x_5 = sq.ft., y_5 = \$)

Input vs. Output:

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

Model



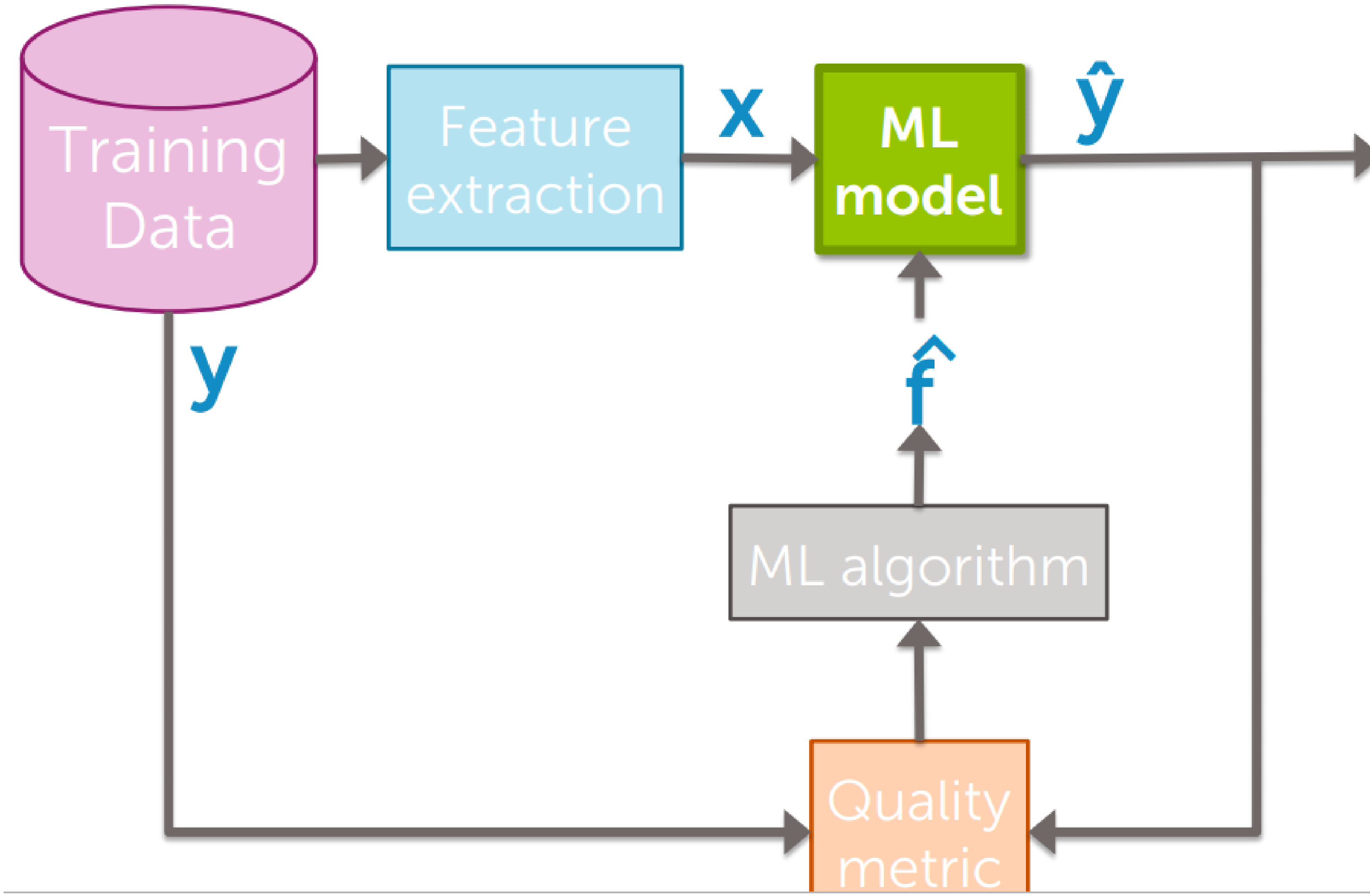
Regression model:

$$y_i = f(x_i) + \epsilon_i$$

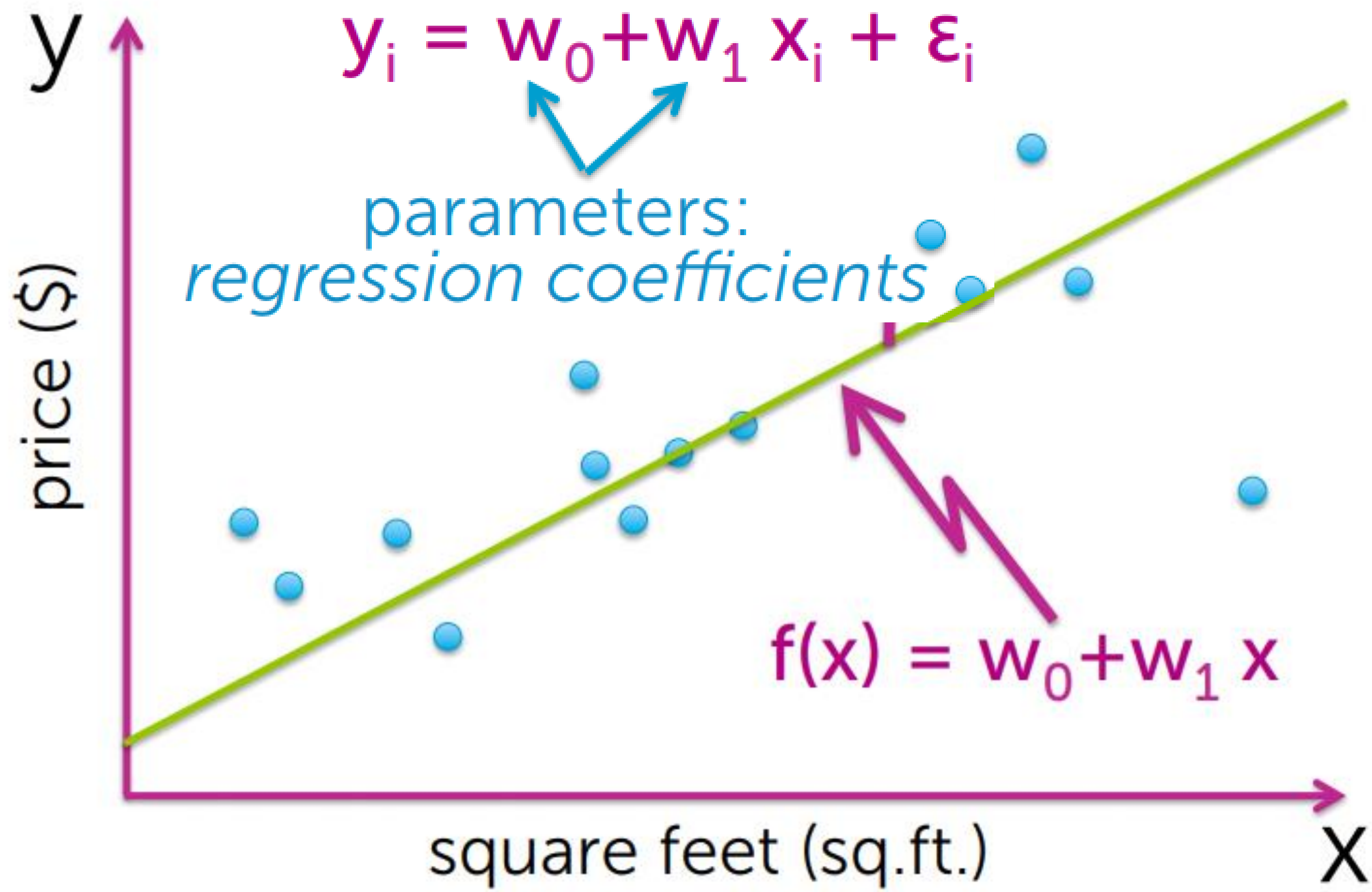
$E[\epsilon_i] = 0$ ← equally likely that error is + or −

↑ expected value

Simple linear regression

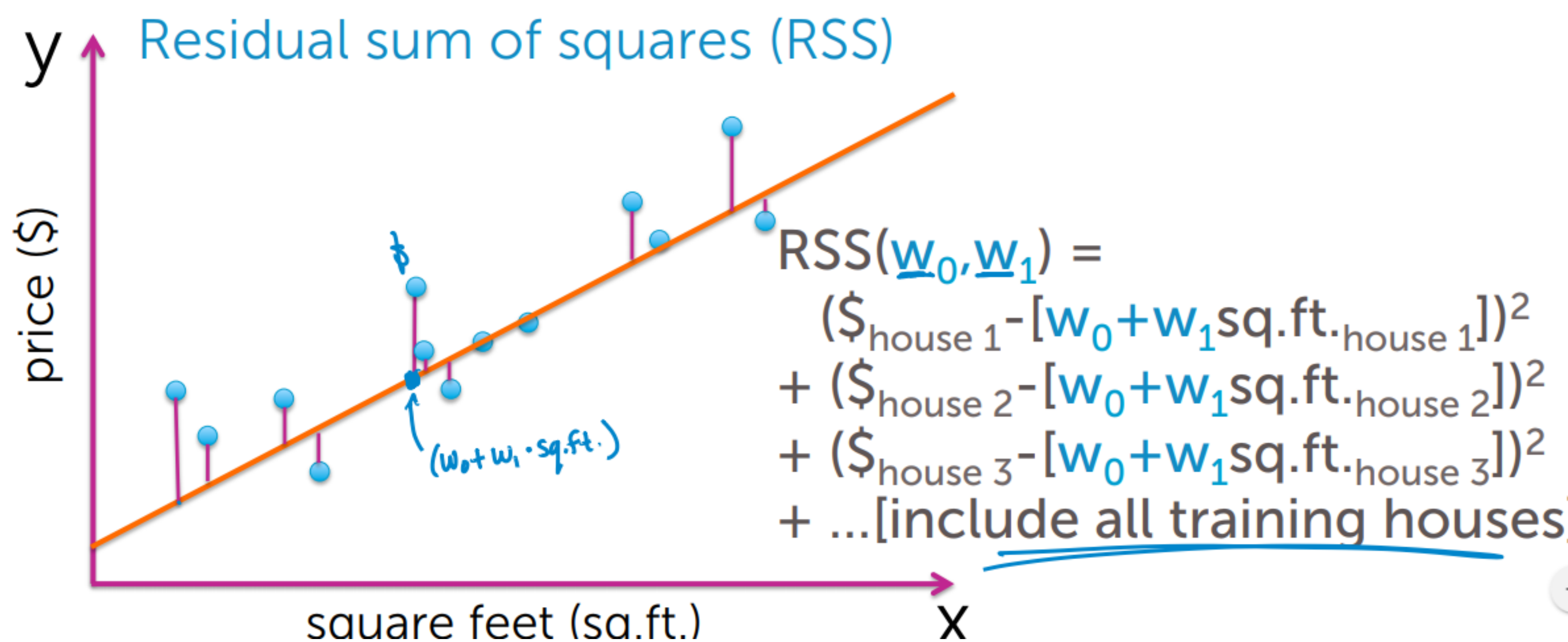


Simple linear Regression

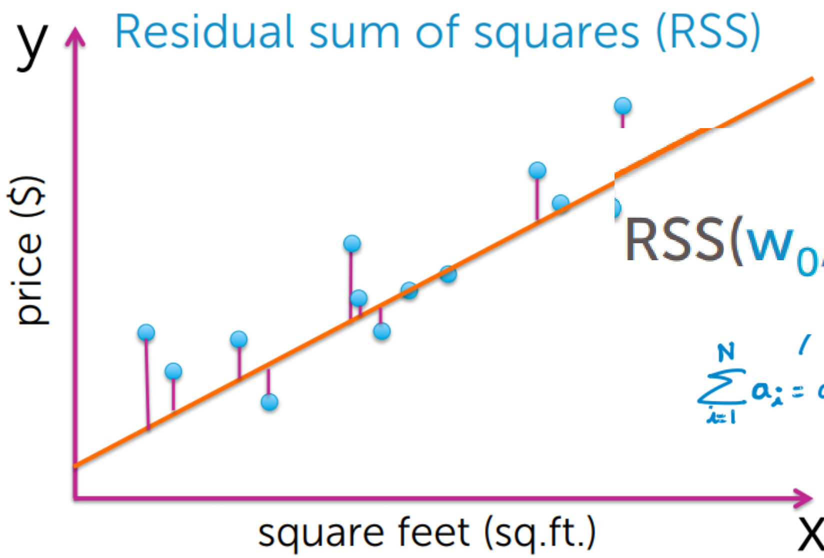


Fitting a line to data

- "Cost" of using a given line.



“Cost” of using a given line



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

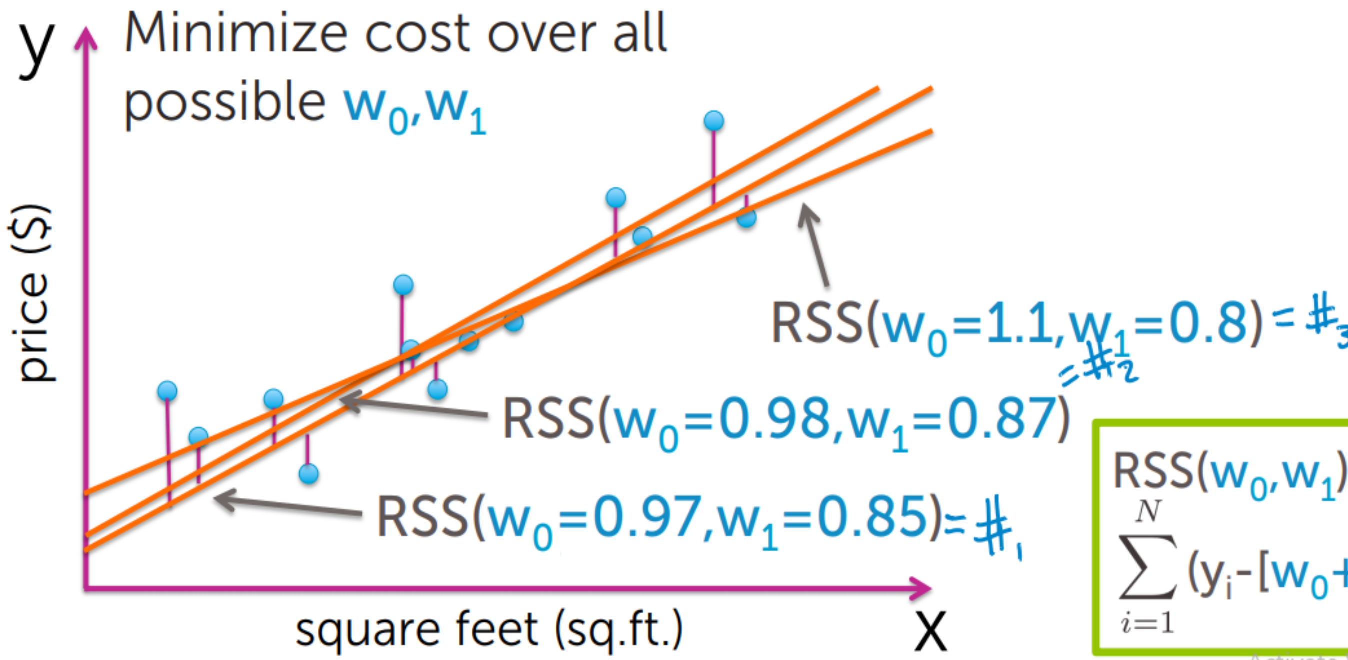
$\sum_{i=1}^{N} a_i = a_1 + a_2$

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

Here,

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

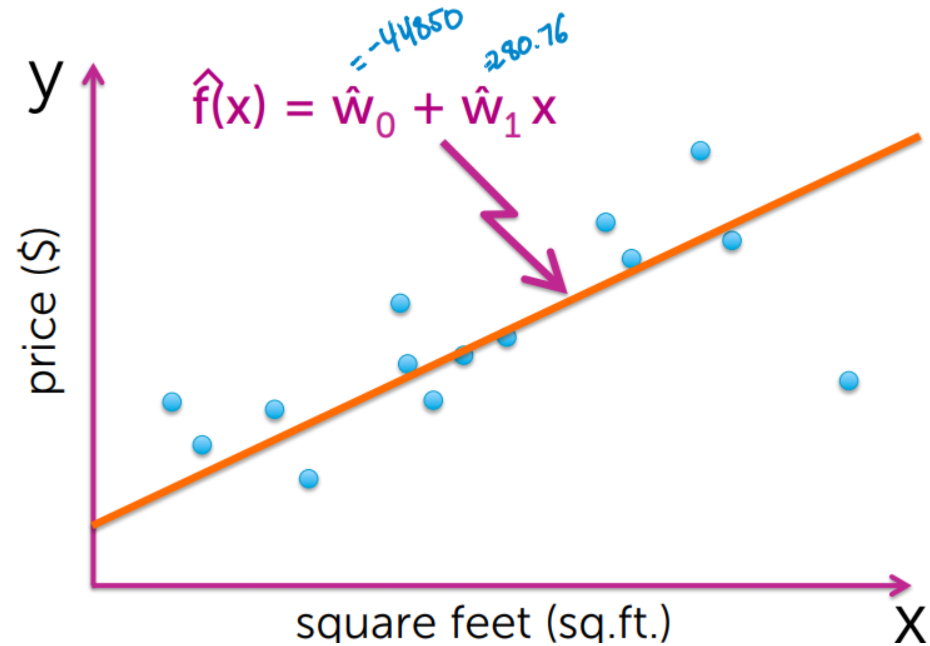
Find “best” line



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

The fitted line: use + interpretation

Model vs. fitted line



Regression model:

$y_i = w_0 + w_1 x_i + \varepsilon_i$

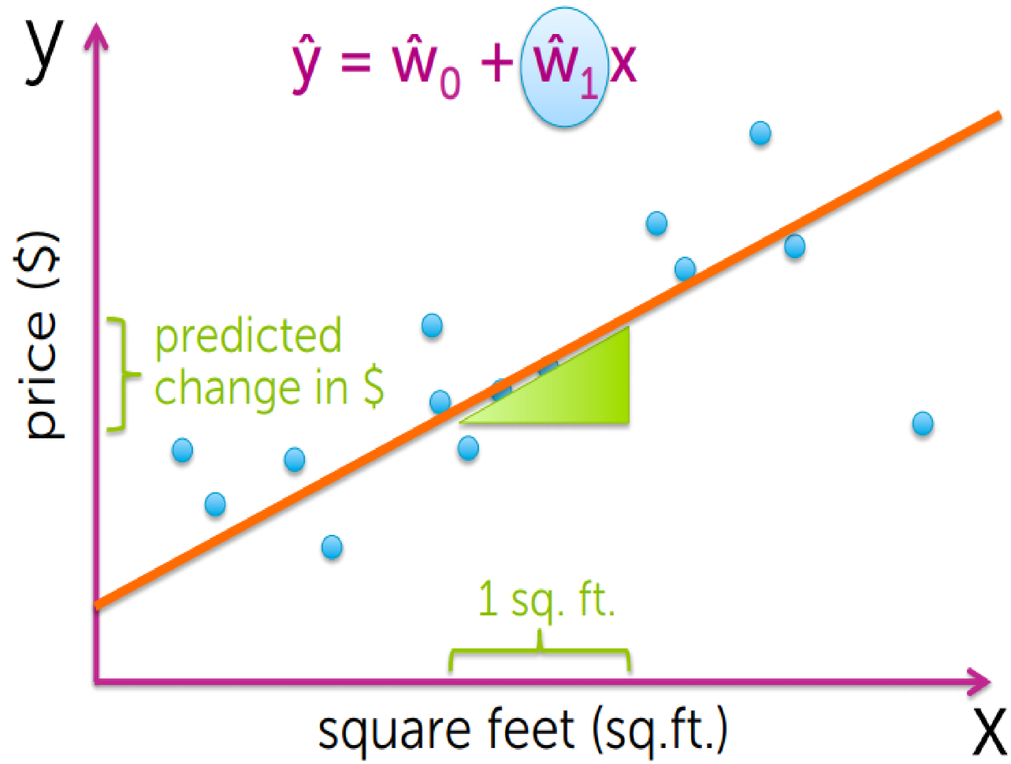
parameters (unknown variables)

Estimated parameters:

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

take actual values

Interpreting the coefficients



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

$= \cancel{\hat{w}_0} + \hat{w}_1 \cdot 1001$ sq.ft.

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

$= \hat{w}_1$

predicted change in the output per unit change in input

Predicted Change in the OUTPUT per unit change in INPUT

Case1: Compute the gradient

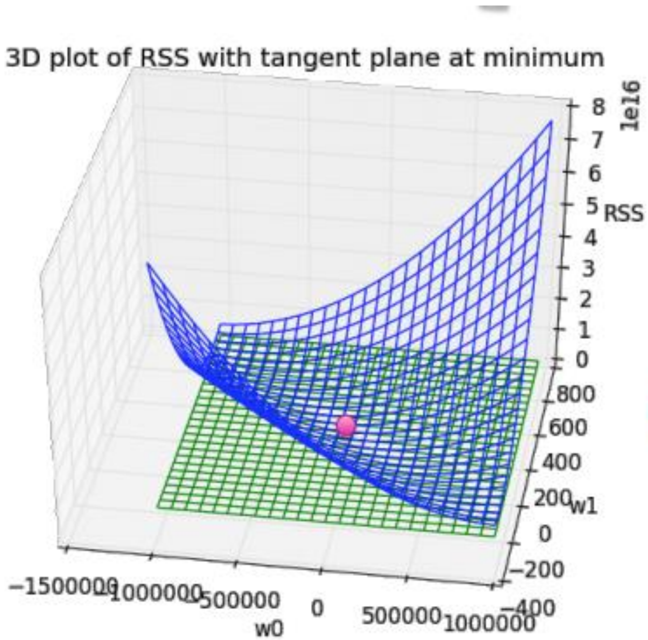
$$\boxed{RSS(w_0,w_1) = \sum_{i=1}^{N}(y_i-[w_0+w_1x_i])^2}$$

Taking the derivative w.r.t. $\underline{w_0}$

Putting it together:

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

Mathematics



top term: average house sales price, estimate the slope, average sq.ft.

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

bottom term:

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

plug in

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

W1- Covariance of (x,y)/var(x)

W0- Mean(y)-W1*Mean(x) this is was we seen Python class

Case 2: Gradient descent

Interpreting the gradient:

actual house sales observation

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

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

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, under predicting $\hat{y}_i$, then $\sum[y_i - \hat{y}_i]$ is positive

$\longrightarrow$ w_0 is going to increase

similar intuition for w_1, but multiply by x_i

Gradient method

- Computing regression parameters (gradient descent example)

The data

Consider the following 5 point synthetic data set:

```
1  X    Y
2  0    1
3  1    3
4  2    7
5  3    13
6  4    21
```

We will need a starting value for the slope and intercept, a step_size and a tolerance

initial_intercept = 0

initial_slope = 0

step_size = 0.05

tolerance = 0.01

In each step of the gradient descent we will do the following:

1. Compute the predicted values given the current slope and intercept
2. Compute the prediction errors (prediction – Y)
3. Update the intercept:

compute the derivative: sum(errors)

compute the adjustment as step_size times the derivative

decrease the intercept by the adjustment

4. Update the slope:

compute the derivative: sum(errors*input)

compute the adjustment as step_size times the derivative

decrease the slope by the adjustment

5. Compute the magnitude of the gradient
6. Check for convergence

The algorithm in action

First step:

Intercept = 0

Slope = 0

1. predictions = [0, 0, 0, 0, 0]
2. errors = [−1, −3, −7, −13, −21]
3. update Intercept

sum([−1, −3, −7, −13, −21]) = −45

adjustment = 0.05 * 45 = −2.25

new_intercept = 0 − −2.25 = 2.25

4. update Slope

sum([0, 1, 2, 3, 4] * [−1, −3, −7, −13, −21]) = −140

adjustment = 0.05 * −140 = −7

new_slope = 0 − −7 = 7

5. magnitude = sqrt((−45)^2 + (−140)^2) = 147.05
6. magnitude > tolerance: not converged

Second step:

Intercept = 2.25

Slope = 7

1. predictions = [2.25, 9.25, 16.25, 23.25, 30.25]
2. errors = [1.25, 6.35, 9.25, 10.25, 9.25]
3. update Intercept

sum([1.25, 6.35, 9.25, 10.25, 9.25]) = 36.25

adjustment = 0.05 * 36.25 = 1.8125

new_intercept = 2.25−1.8125 = 0.4375

4. update Slope

sum([0, 1, 2, 3, 4] * [1.25, 6.35, 9.25, 10.25, 9.25]) = 92.5

adjustment = 0.05 * 92.5 = 4.625

new_slope = 7 − 4.625 = 2.375

5. magnitude = sqrt((36.25)^2 + (92.5)^2) = 99.35
6. magnitude > tolerance: not converged

Let's skip forward a few steps… after the 77th step we have gradient magnitude 0.0107.

78th Step:

Intercept = −0.9937

Slope = 4.9978

1. predictions = [−0.99374, 4.00406, 9.00187, 13.99967, 18.99748]
2. errors = [−1.99374, 1.00406, 2.00187, 0.99967, −2.00252]
3. update Intercept

sum([−1.99374, 1.00406, 2.00187, 0.99967, −2.00252]) = 0.009341224

adjustment = 0.05 * 0.009341224 = 0.0004670612

new_intercept = −0.9937 − 0.0004670612 = −0.994207

4. update Slope

sum([0, 1, 2, 3, 4] * [−1.99374, 1.00406, 2.00187, 0.99967, −2.00252]) = −0.0032767

adjustment = 0.05 *−0.0032767 = −0.00016383

new_slope = 4.9978 −−0.00016383 = 4.9979

5. magnitude = sqrt[()^2 + ()^2] = 0.0098992
6. magnitude < tolerance: converged!

Final slope: −0.994

Final Intercept: 4.998

If you continue you will get to (−1, 5) but at this point the change in RSS (our cost) is negligible.

Multiple Linear regression

Generic basis expansion

Model:

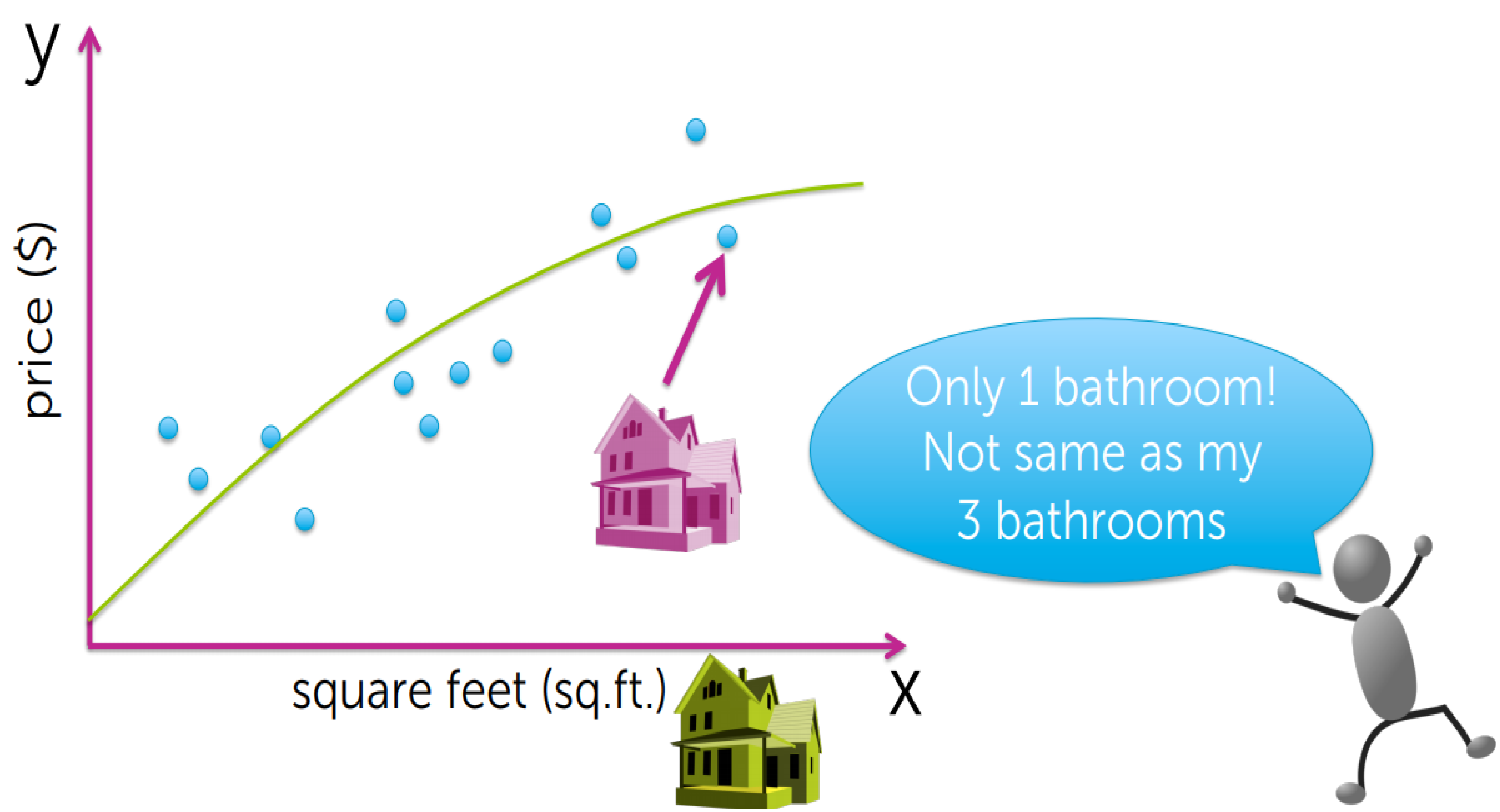
$$y_i = w_0 h_0(x_i) + w_1 h_1(x_i) + \ldots + w_D h_D(x_i) + \varepsilon_i$$

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

j^{th} *feature*

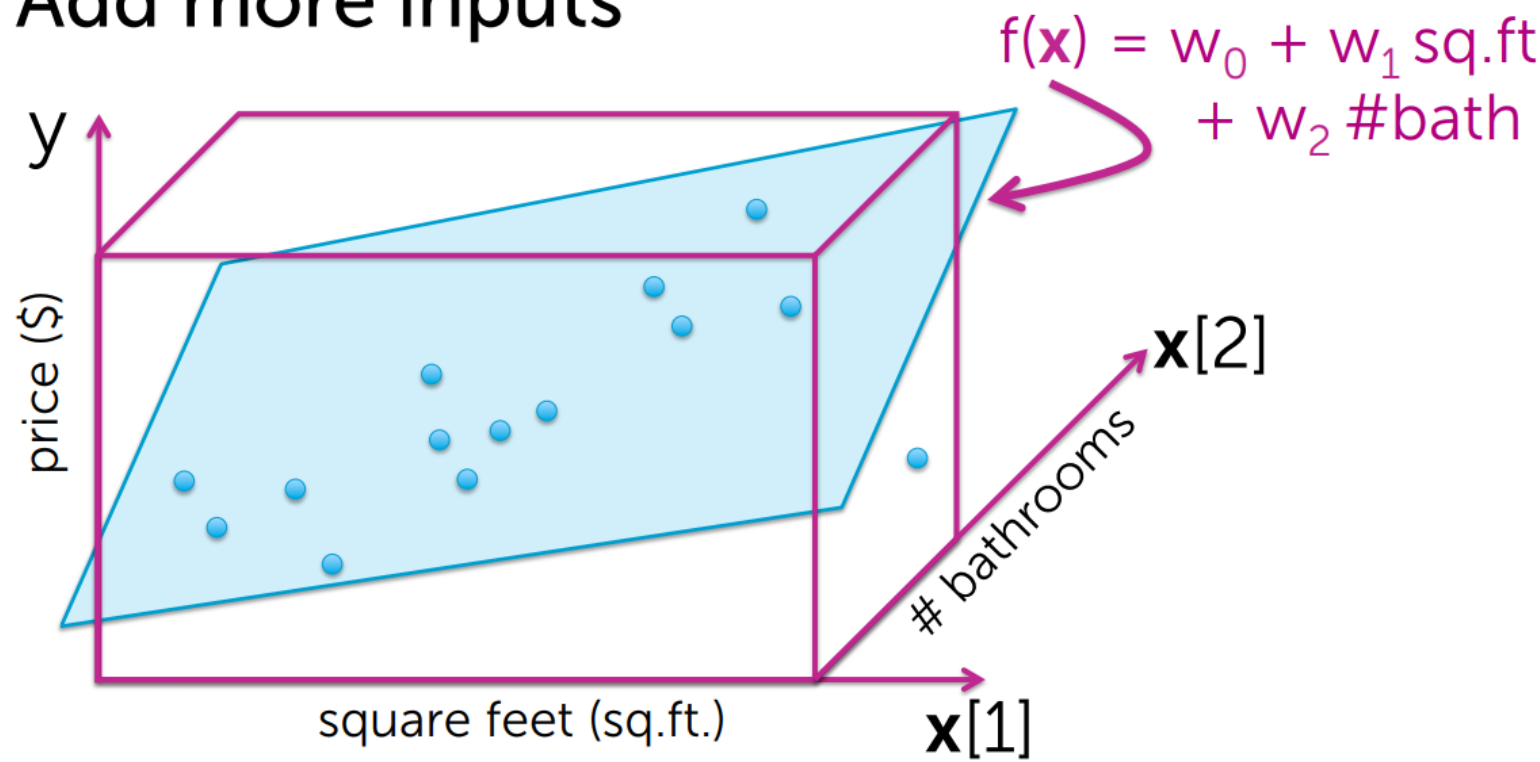
j^{th} *regression coefficient*
or *weight*

Predictions just based on house size



Dimensionality changes addition of Features

Add more inputs



More generically… D-dimensional curve

Model:

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

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

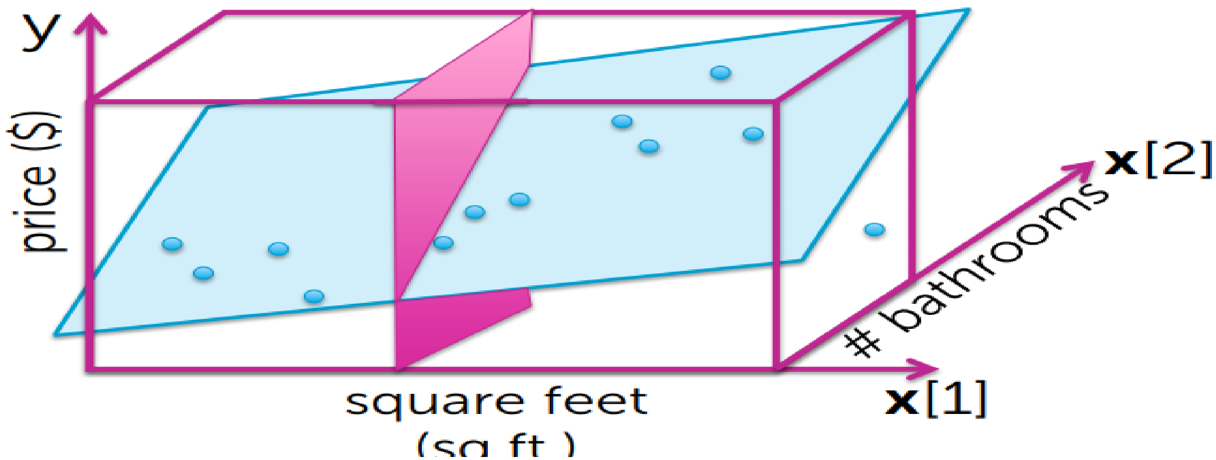
feature 1 = $h_0(\mathbf{x})$ … e.g., 1
feature 2 = $h_1(\mathbf{x})$ … e.g., $\mathbf{x}[1]$ = sq. ft.
feature 3 = $h_2(\mathbf{x})$ … e.g., $\mathbf{x}[2]$ = #bath
or, $\log(\mathbf{x}[7])\, \mathbf{x}[2]$ = log(#bed) x #bath
…
feature D+1 = $h_D(\mathbf{x})$ … some other function of $\mathbf{x}[1],\ldots, \mathbf{x}[d]$

Interpreting the fitted function

Two linear features

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

fix



When we fix one feature and rest one unit increase what will be the changes in Y

Fitting D–dimensional curves

- Step 1: Rewrite the regression model

For observation i

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

$$y_i = \overbrace{\begin{bmatrix} w_0 & w_1 & w_2 & \cdots & w_D \end{bmatrix}}^{w^T} \overbrace{\begin{bmatrix} h_0(x_i) \\ h_1(x_i) \\ h_2(x_i) \\ \vdots \\ h_D(x_i) \end{bmatrix}}^{h(x_i)} \quad = \quad \overbrace{\begin{bmatrix} h_0(x_i) & h_1(x_i) & \cdots & h_D(x_i) \end{bmatrix}}^{h^T(x_i)} + \varepsilon_i \quad \overbrace{\begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_D \end{bmatrix}}^{w}$$

$$= \underline{w_0 h_0(x_i) + w_1 h_1(x_i) + \dots + w_D h_D(x_i)} + \epsilon_i$$

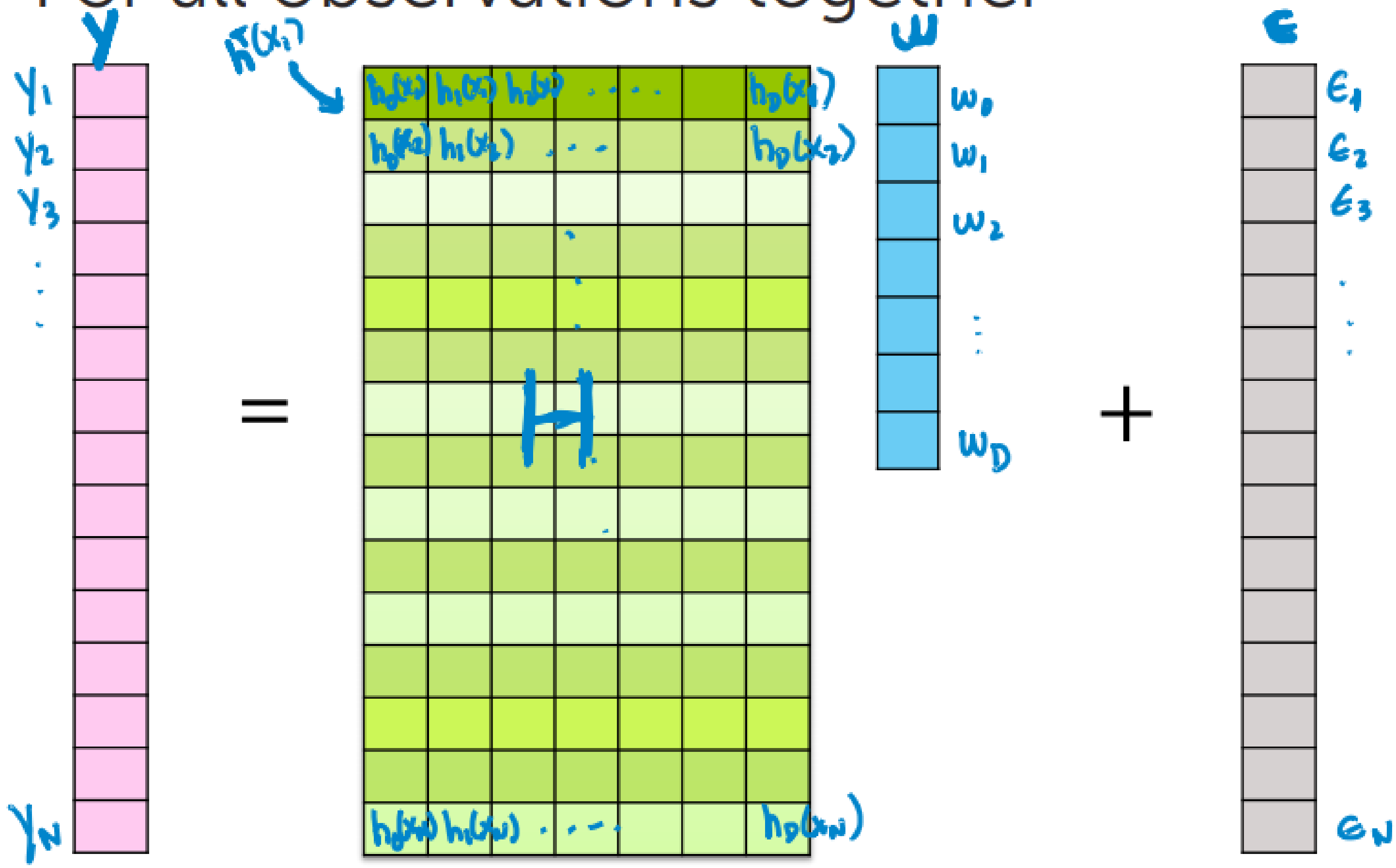
scalar

$$= w^T h(x_i) + \epsilon_i$$

$$= h_0(x)\, w_0 + h_1(x_i)\, w_1 + \dots + h_D(x_i)\, w_D + \epsilon_i$$

Rewrite in matrix notation

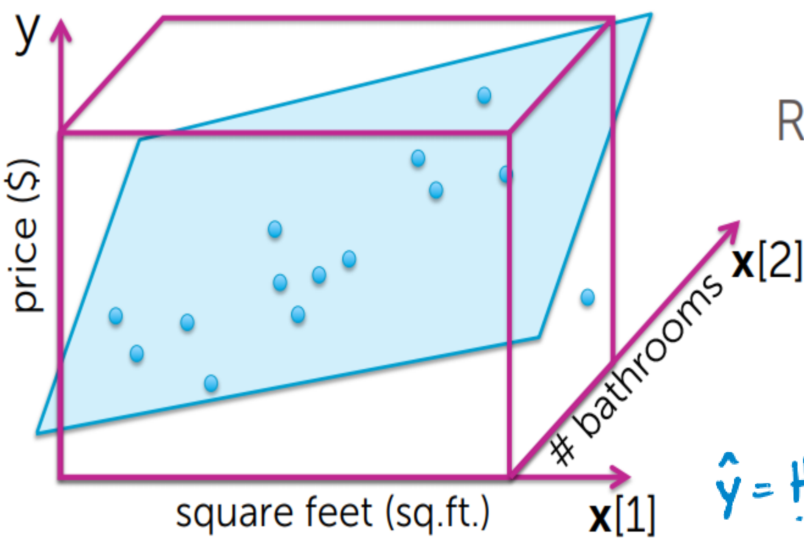
For all observations together



We can write this as Y=HW+epsilon

Step 2: Compute the cost

- RSS for multiple regression



$$\mathrm{RSS}(\mathbf{w}) = \sum_{i=1}^{N} (y_i - \underbrace{h^T(x_i)\, w}_{\hat{y}_i(w)})^2$$

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

$$\hat{y} = Hw$$

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

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

RSS in Matrix Notations

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

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

residual_1	residual_2	residual_3	…	residual_N

residual_1
residual_2
residual_3
…
residual_N

Step 3: Take the gradient

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

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

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

s

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

Solve for **w**:

$$-\cancel{2}H^Ty + \cancel{2}H^THŵ = 0$$

$$H^THŵ = H^Ty$$

$$\underbrace{(H^TH)^{-1}H^TH}_{I}ŵ = (H^TH)^{-1}H^Ty$$

$$ŵ = (H^TH)^{-1}H^Ty$$

$A^{-1}A = I$

- $Iv = v$
  $IV = V$

Step 4,
Approach 2: Gradient descent

- Gradient descent is most powerful because earlier methods are highly computational intensive.

- Thank You