# Introduction to Machine Learning

# Time For Some Cool Demonstrations!

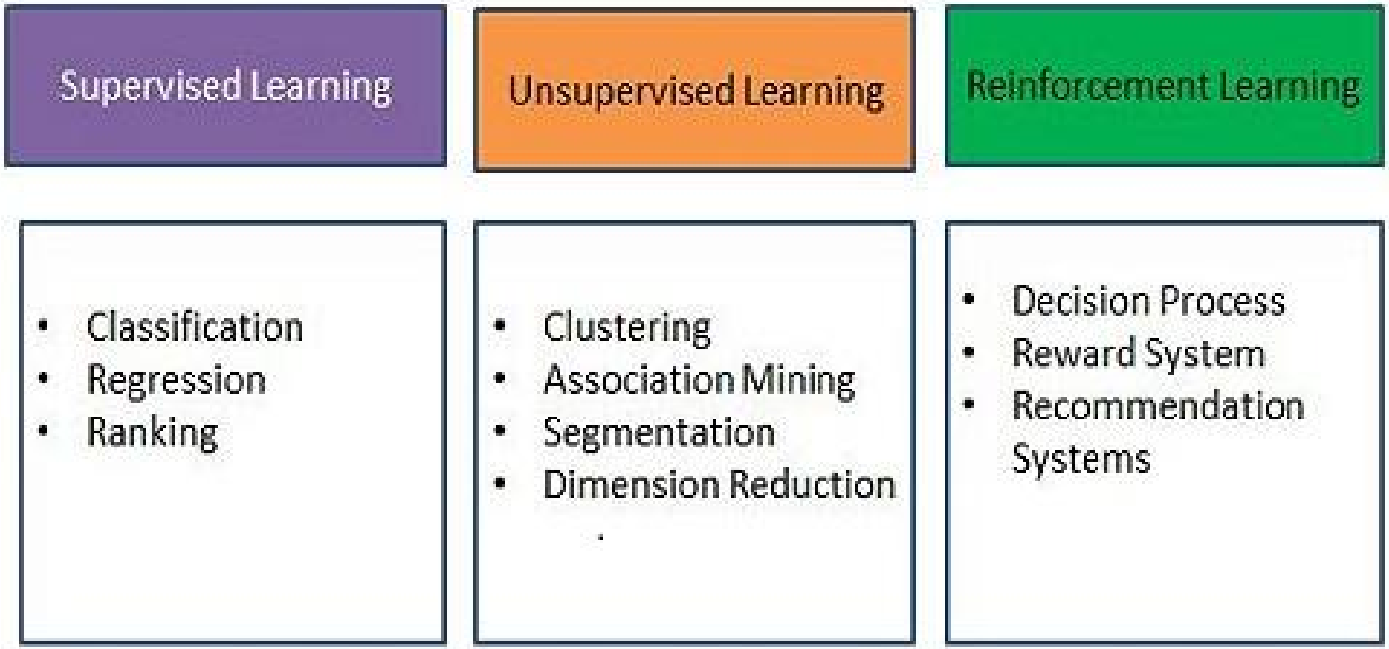
# Contents

# What is Machine Learning

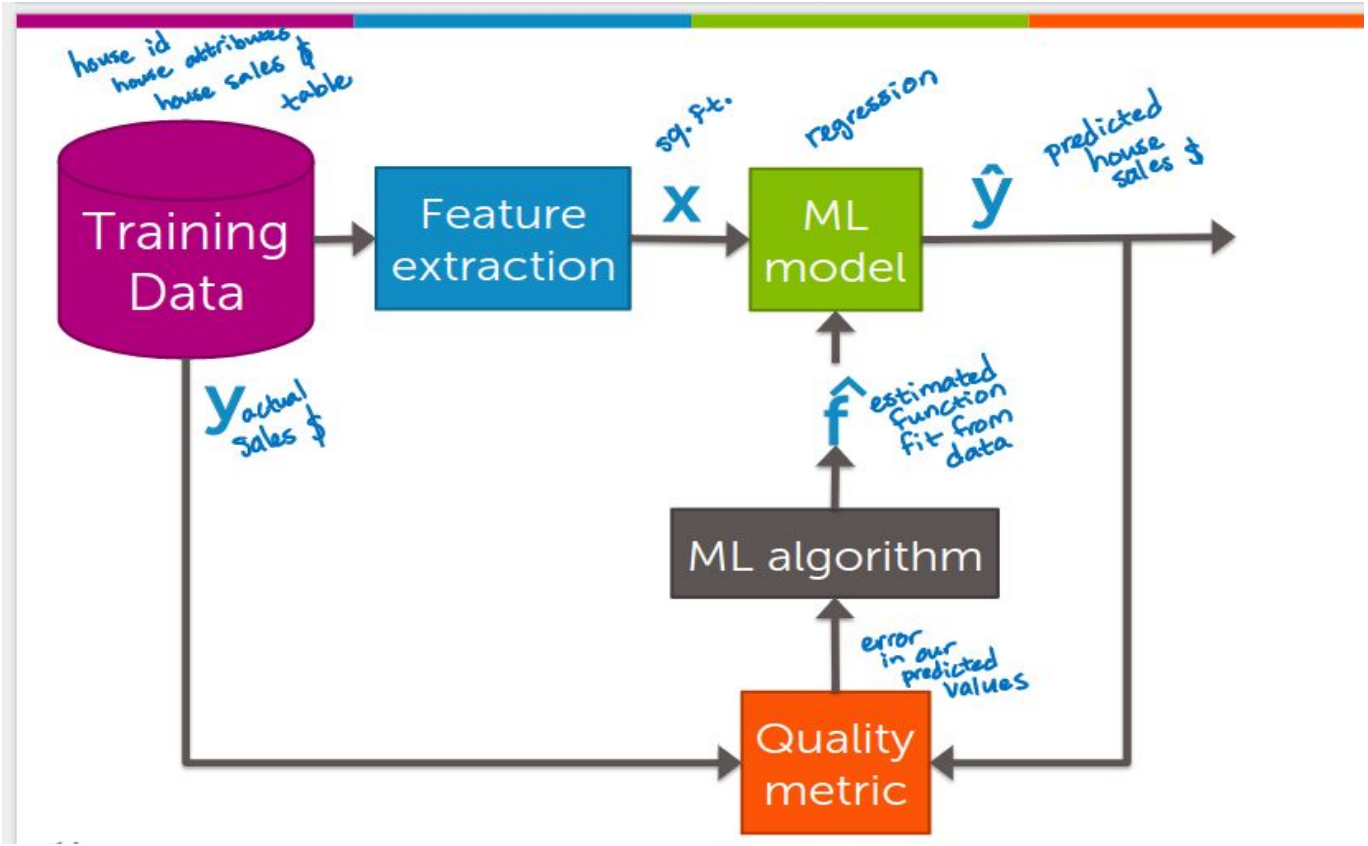
"the field of study that gives computers the ability to learn without being explicitly programmed."

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

# Types of Machine Learning



# How does supervised learning work ?



# Regression Model

# How much is my house worth?

# Look at recent sales in my neighborhood

- How much did they sell for?

# Data

*input* *output*

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

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

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

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

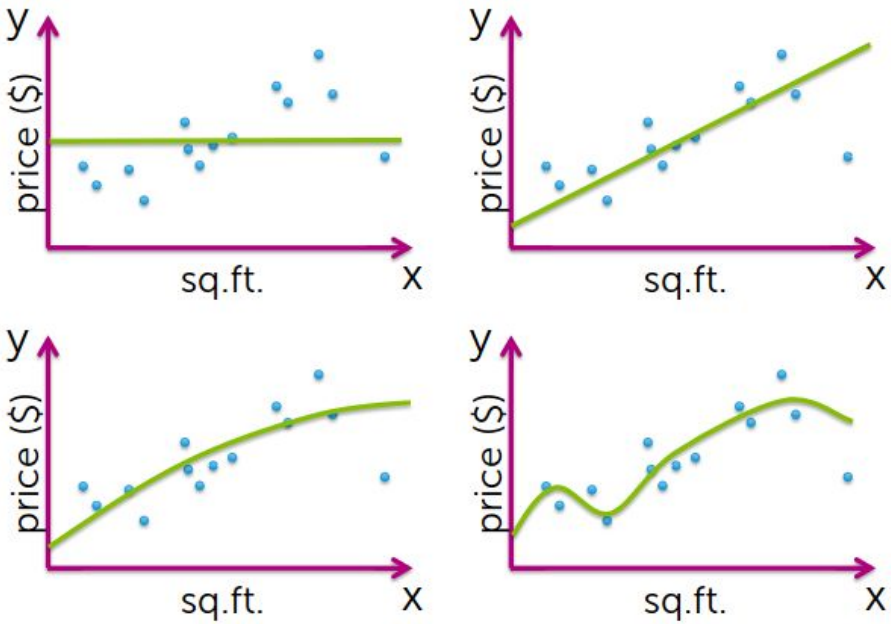
$(x_5 = \text{sq.ft.}, y_5 = \$)$

⋮

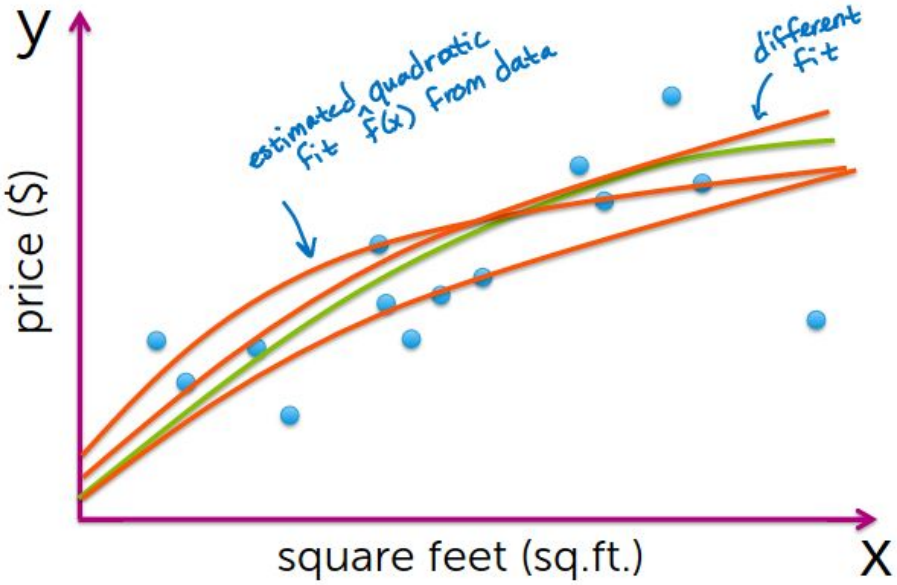
**Input vs. Output:**

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

# Task 1–
# Which model f(x)?

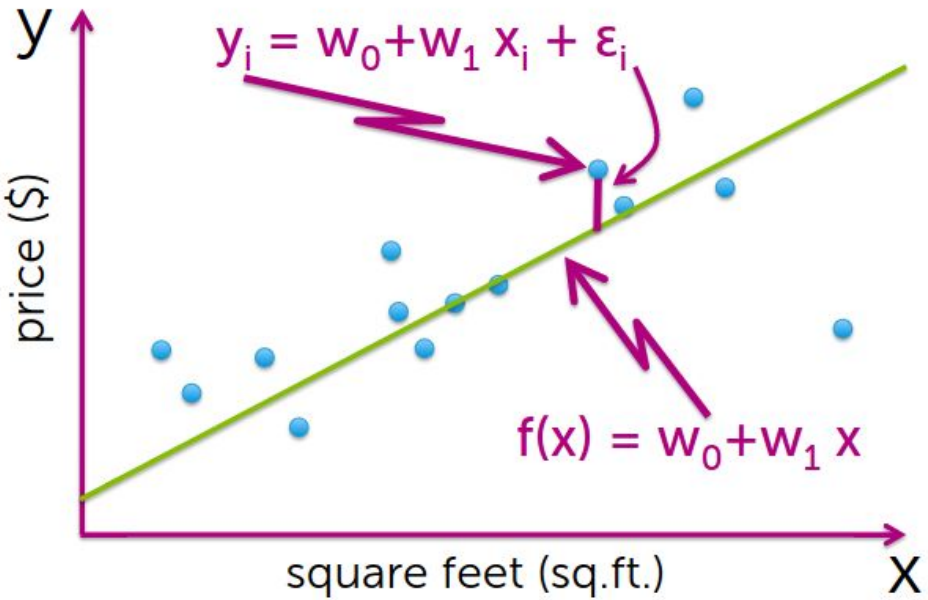


# Task 2 – For a given model f(x), estimate function $\hat{f}(x)$ from data

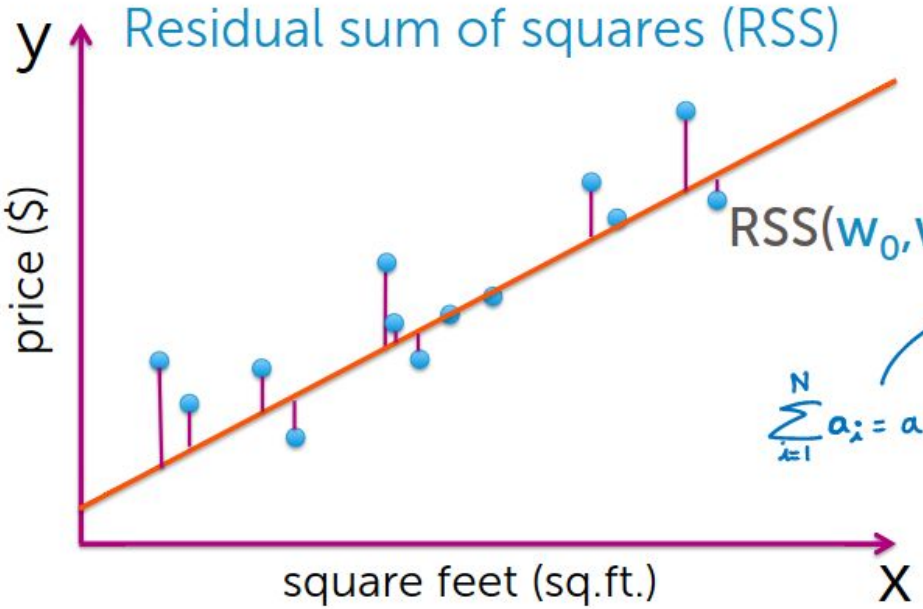



Assume model f(x) is a quadratic function

# Simple linear regression model



# "Cost" of using a given line

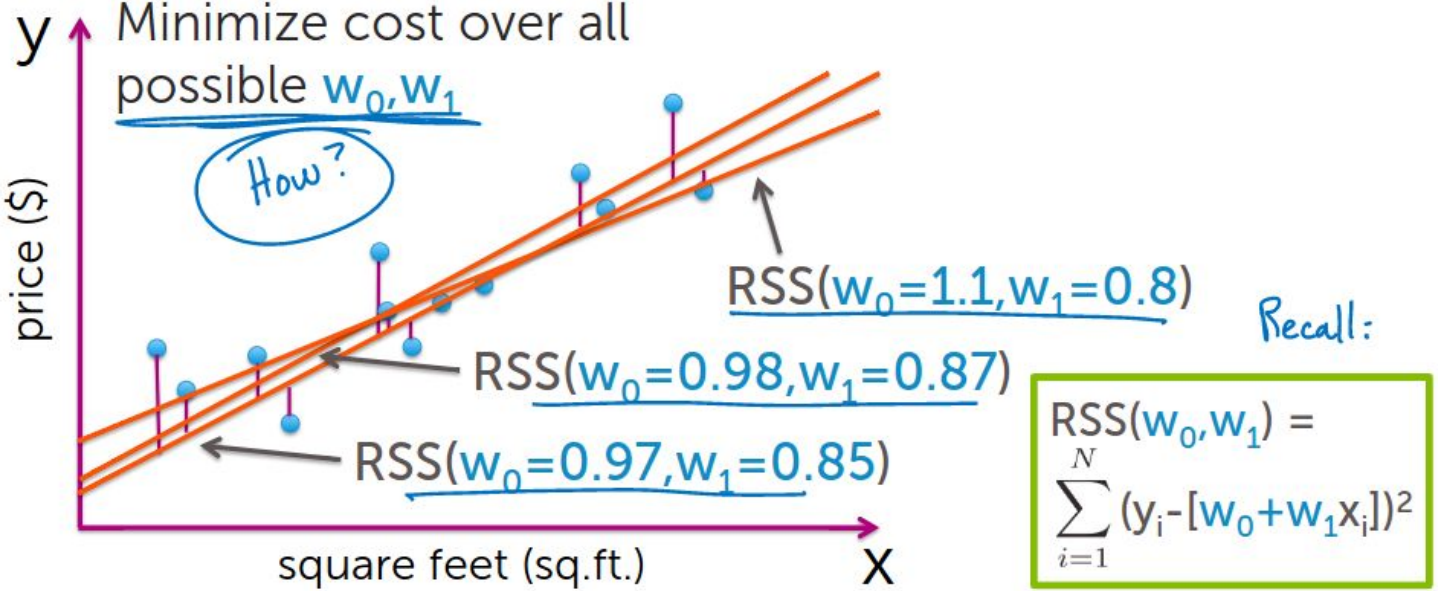



$$RSS(w_0, w_1) = \sum_{i=1}^{N} (y_i - [w_0 + w_1 x_i])^2$$

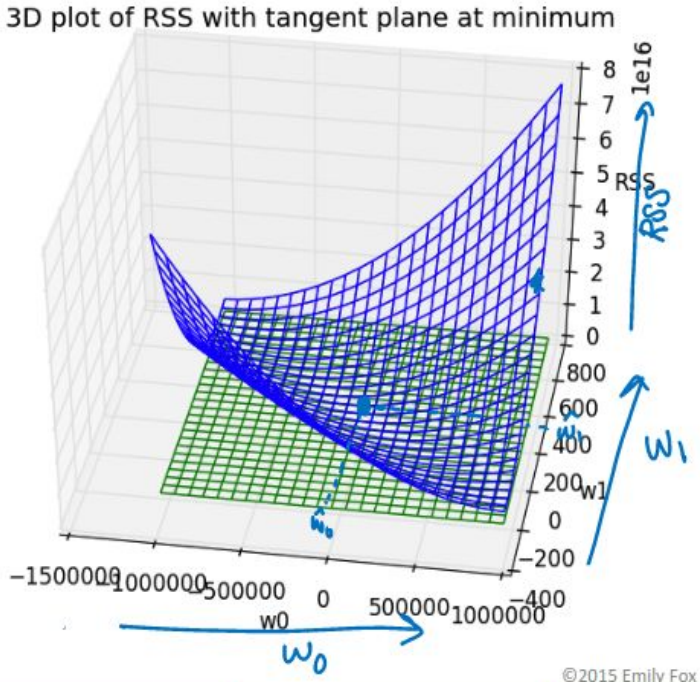
$$\sum_{i=1}^{N} a_i = a_1 + a_2 + \cdots + a_N$$

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

# Find "best" line



# Minimizing the cost

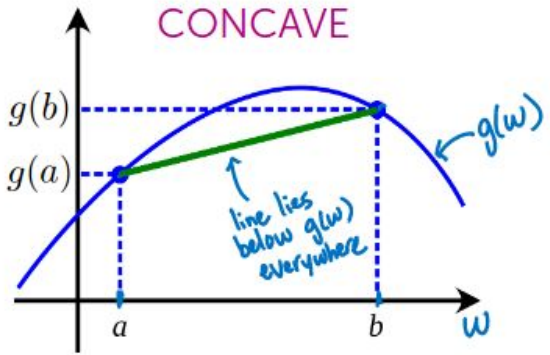


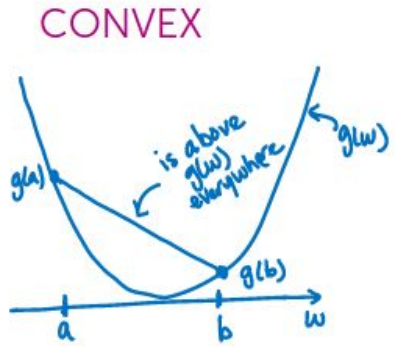

Minimize function over all possible $w_0, w_1$

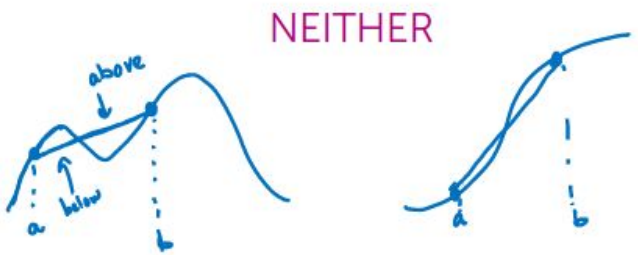
$$\min_{w_0, w_1} \sum_{i=1}^{N} (y_i - [w_0 + w_1 x_i])^2$$

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

# Convex/concave functions

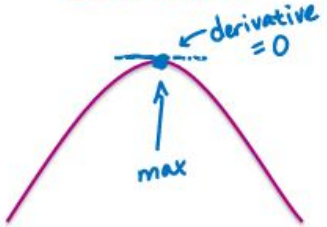


# Finding the max or min analytically

CONCAVE

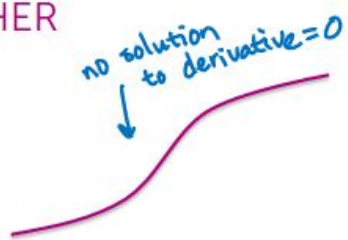



$\max_w g(w)$

CONVEX


$\min_w g(w)$

NEITHER


Example:

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

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

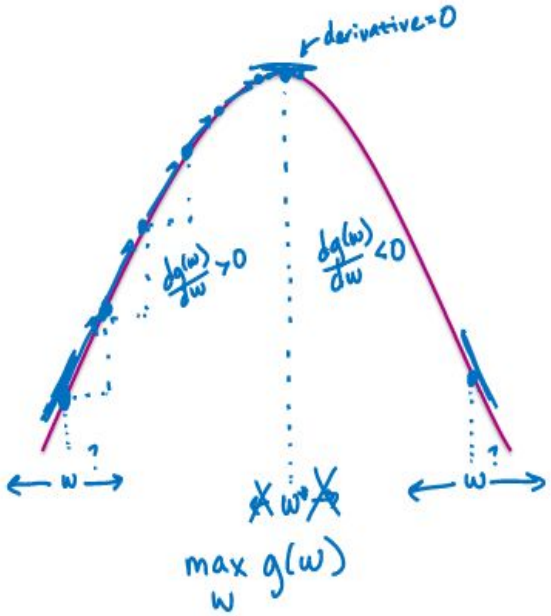
$= -2w + 20$

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?)

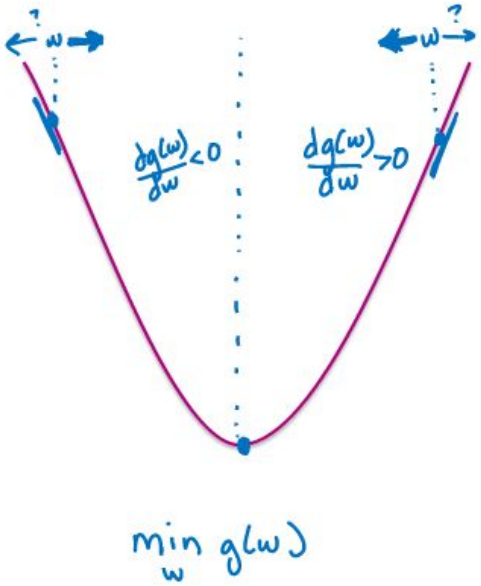
while not converged

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

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:

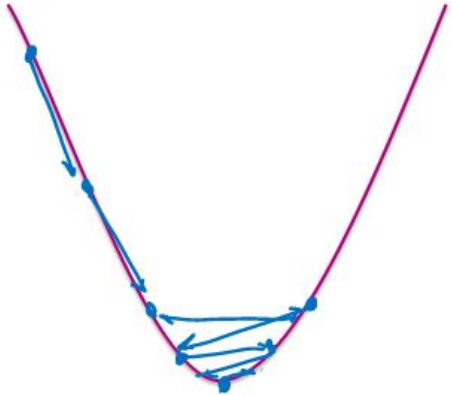
**while** not converged

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

# Choosing the stepsize—

## Fixed stepsize

$\eta$



$\eta = 0.1$

# Convergence criteria

For convex functions, optimum occurs when

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

In practice, stop when

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

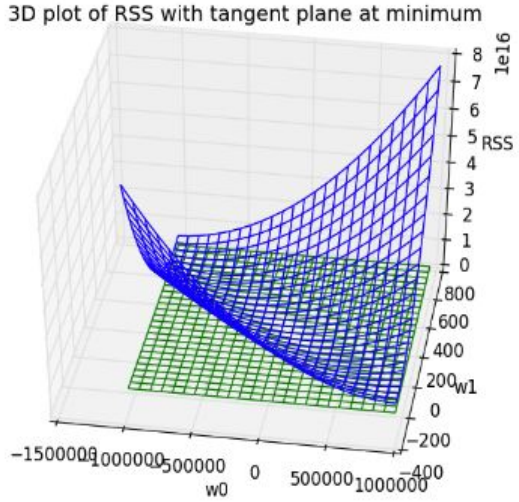
↑ threshold to be set

Algorithm:

**while** not converged

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

# Moving to multiple dimensions: Gradients




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

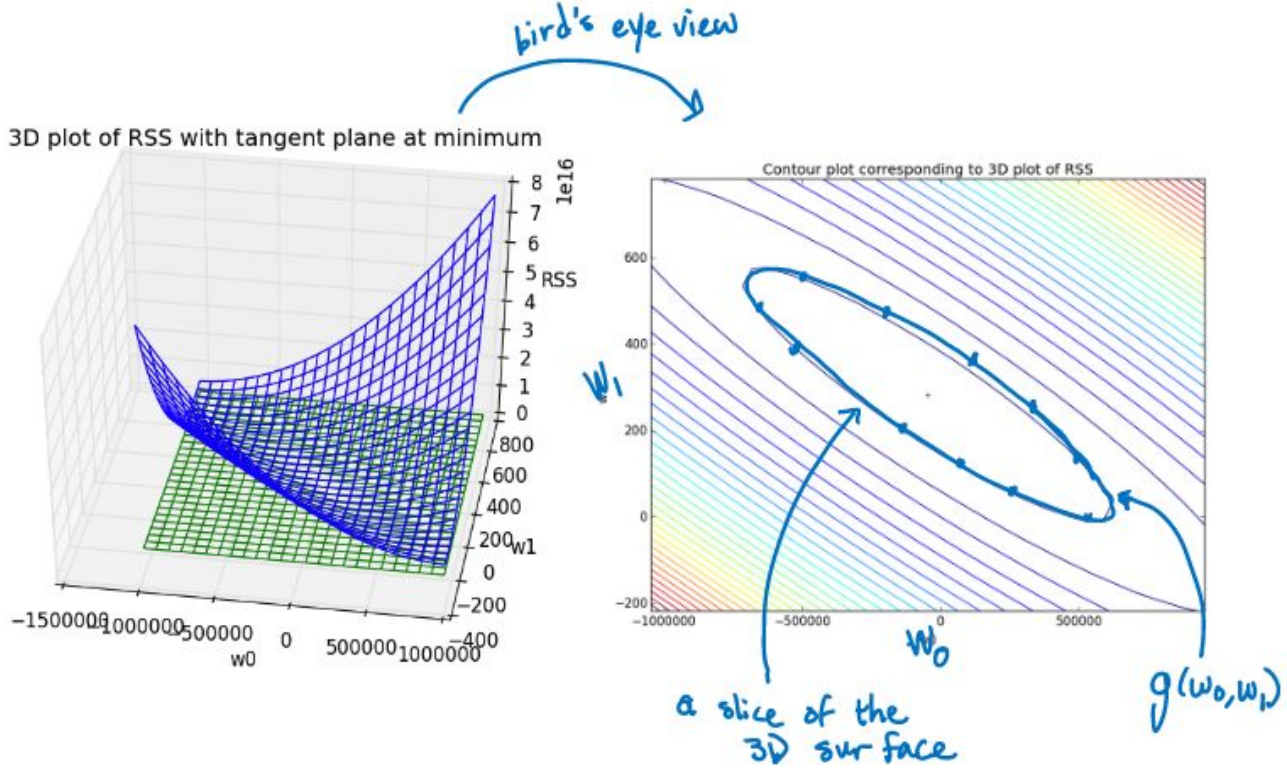
gradient

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

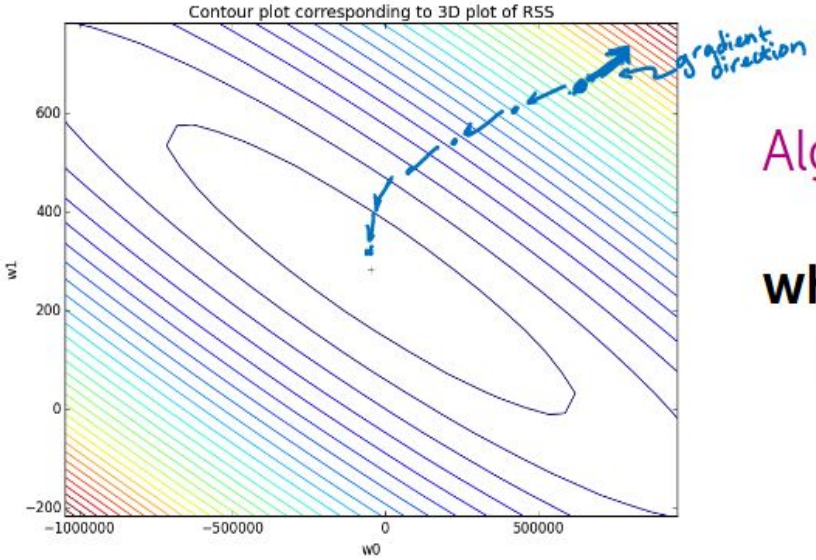
$(p+1)$-dimensional vector

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

# Contour plots



# Gradient descent




Algorithm:

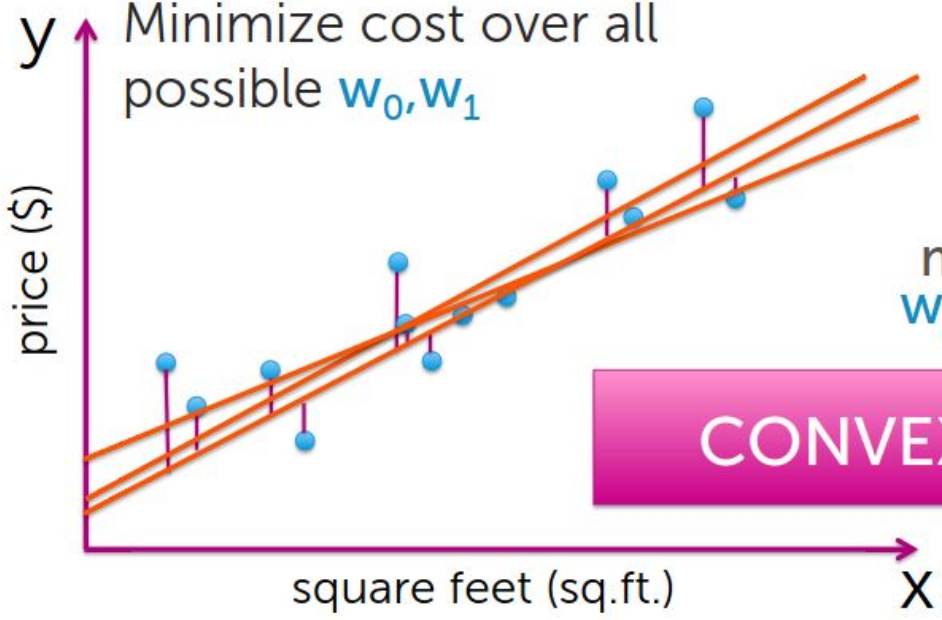
**while** not converged

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

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

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

# 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

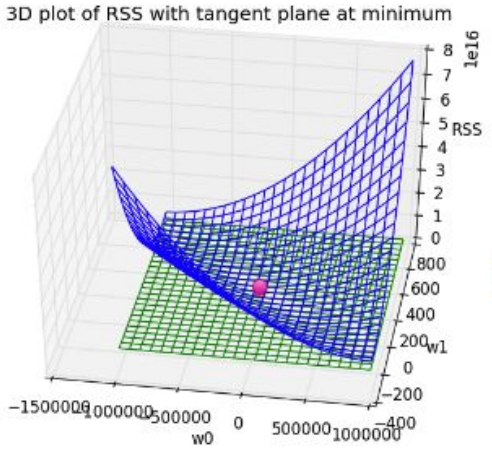
$$RSS(w_0,w_1) = \sum_{i=1}^{N} (y_i-[w_0+w_1x_i])^2$$

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:

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

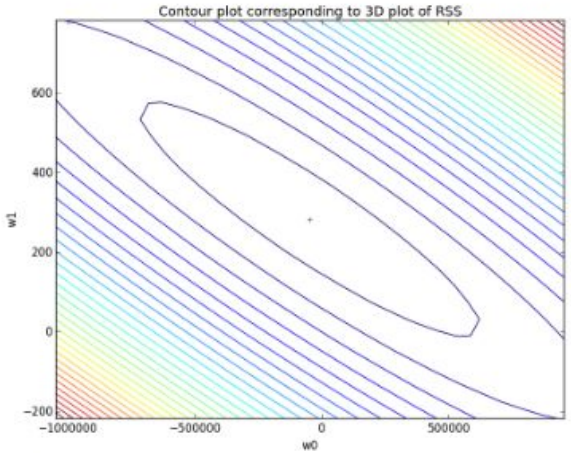
$$\sum_{i=1}^{N} x_i$$

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

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

# Approach 2: Gradient descent

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




while not converged

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

$(-2)\cdot(-\eta)$

If overall, underpredicting $\hat{y}_i$, then $\sum[y_i - \hat{y}_i]$ is positive

→ $w_0$ is going to increase

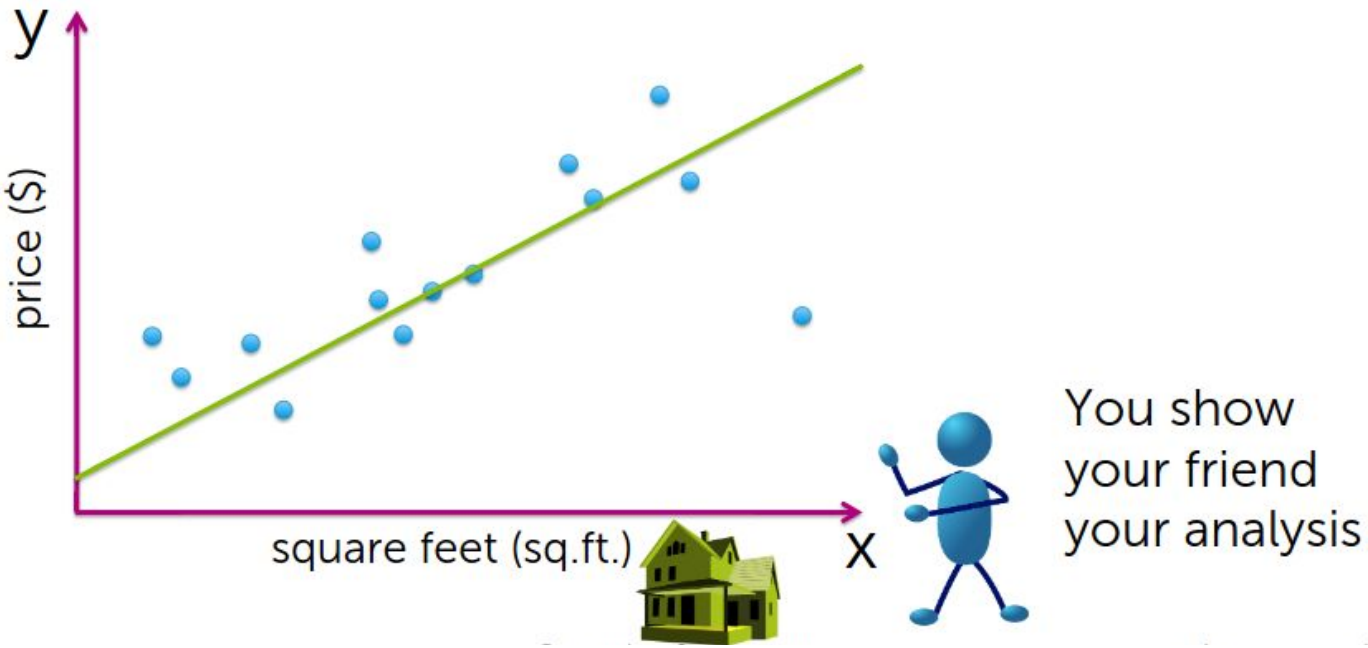
similar intuition for $w_1$, but multiply by $x_i$

# Comparing the approaches

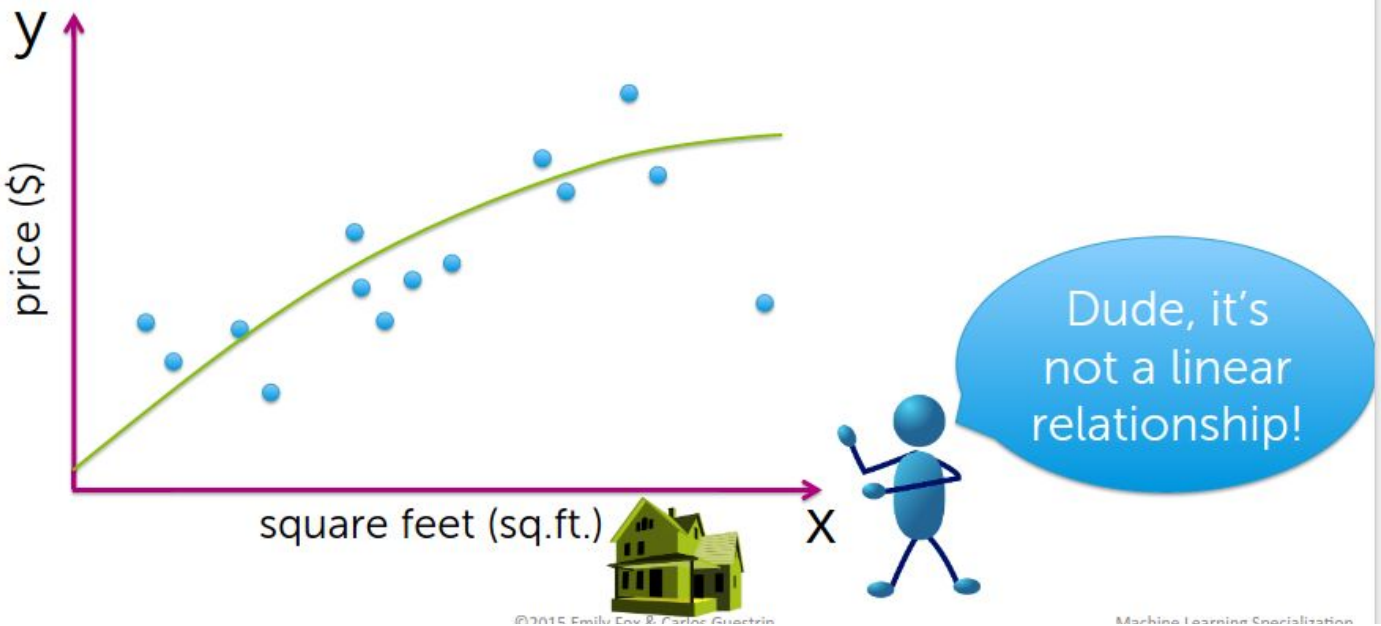
- For most ML problems, cannot solve gradient = 0
- Even if solving gradient = 0 is feasible, gradient descent can be more efficient
- Gradient descent relies on choosing stepsize and convergence criteria

# Influence of high leverage points

# Fit data with a line or … ?



# What about a quadratic function?



# 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 = $w_p$ |

# Generic basis expansion

Model:

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

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

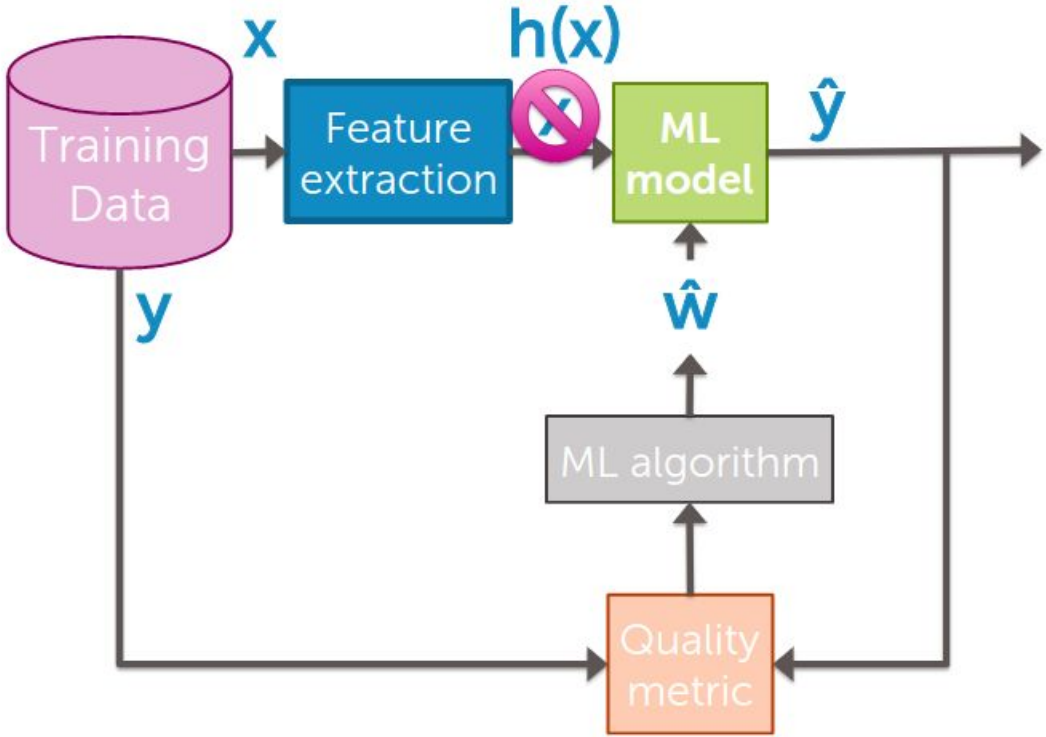
*feature 1* = $h_0(x)$…often 1 (constant)

*feature 2* = $h_1(x)$… e.g., $x$

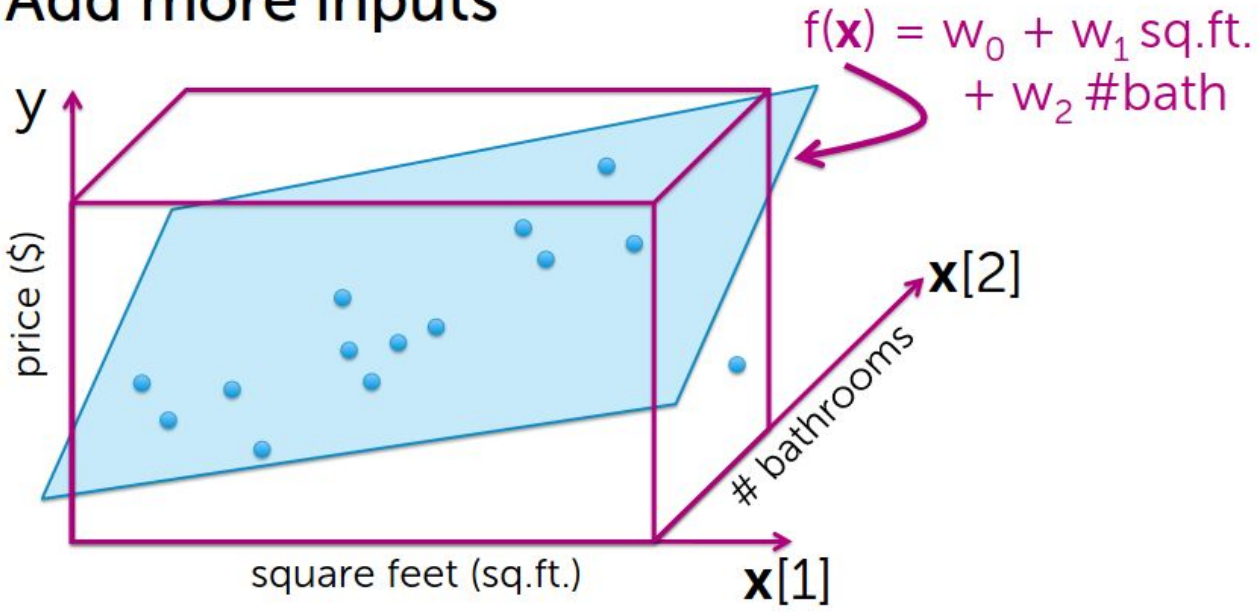
*feature 3* = $h_2(x)$… e.g., $x^2$ or $\sin(2\pi x/12)$

…

*feature D+1* = $h_D(x)$… e.g., $x^p$


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

# Add more inputs



# 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*)

# 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

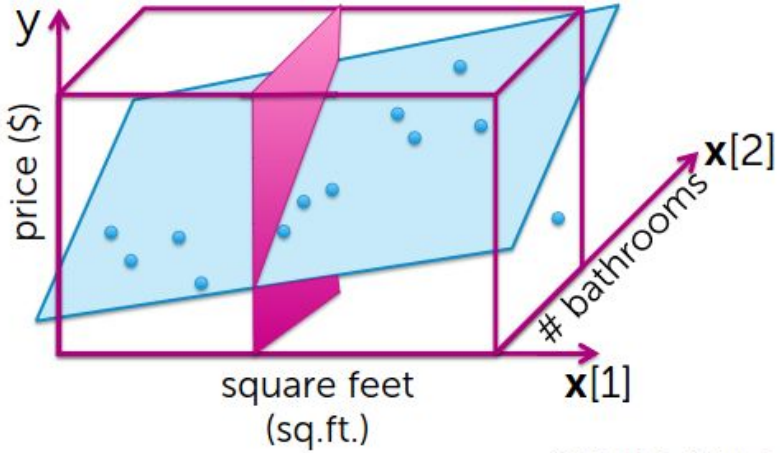
# inputs $\mathbf{x}[j]$ : d

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

# Interpreting the coefficients – Two linear features

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

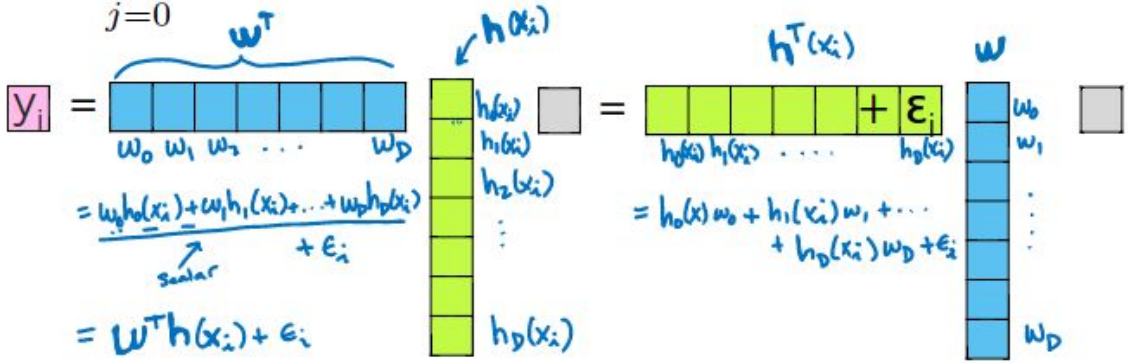
fix



# Rewrite in matrix notation

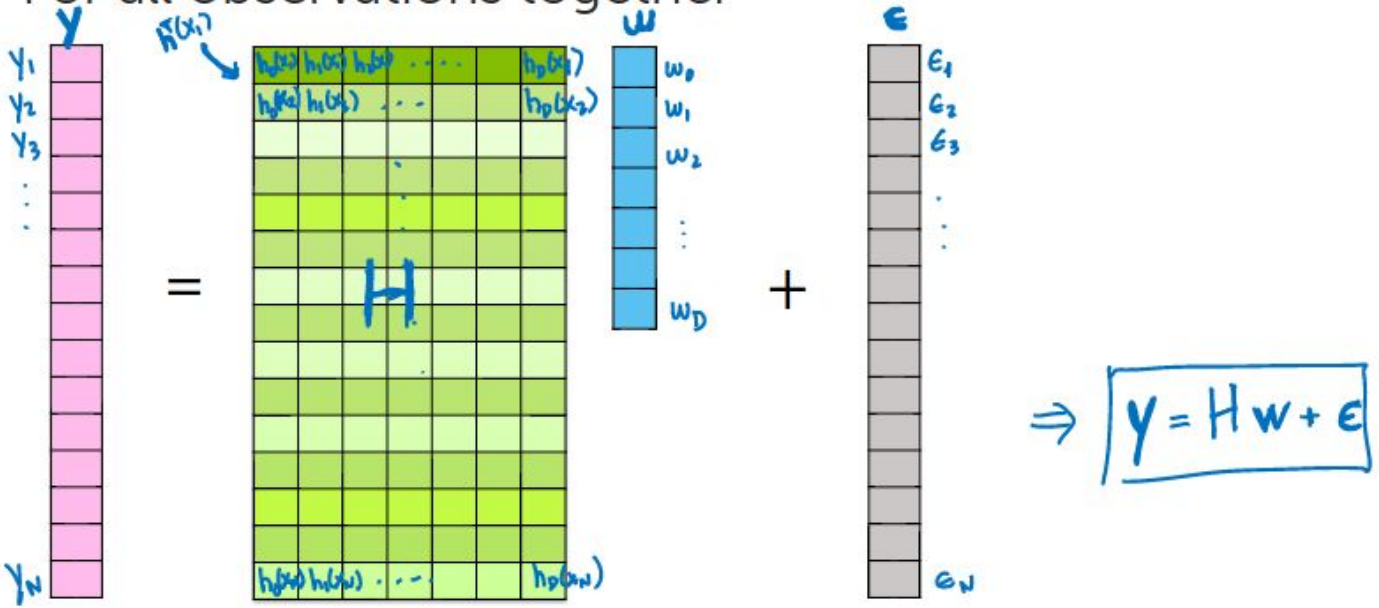
For observation i

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



# Rewrite in matrix notation

For all observations together



# 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)

| | | | | |
|---|---|---|---|---|
| $\text{residual}_1$ | $\text{residual}_2$ | $\text{residual}_3$ | ... | $\text{residual}_N$ |

$$\begin{bmatrix} \text{residual}_1 \\ \text{residual}_2 \\ \text{residual}_3 \\ \dots \\ \text{residual}_N \end{bmatrix} = (\text{residual}_1^2 + \text{residual}_2^2 + \dots + \text{residual}_N^2) = \sum_{i=1}^{N} \text{residual}_i^2 \triangleq \text{RSS}(\mathbf{w})$$

# Gradient of RSS

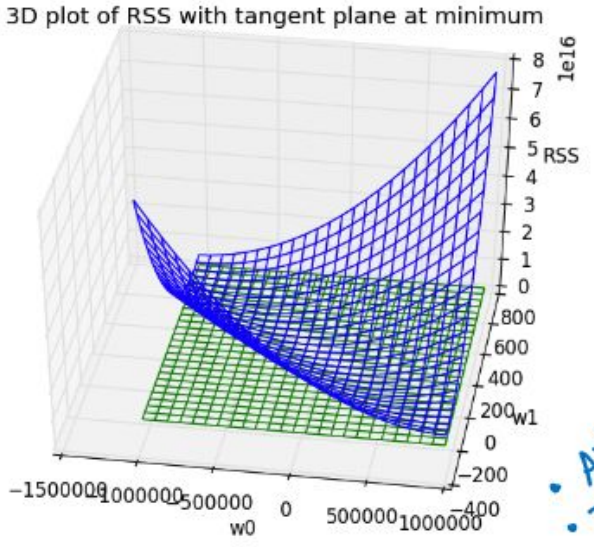
$$\nabla \mathrm{RSS}(\mathbf{w}) = \nabla[(\mathbf{y}-\mathbf{H}\mathbf{w})^T(\mathbf{y}-\mathbf{H}\mathbf{w})]$$
$$= -2\mathbf{H}^T(\mathbf{y}-\mathbf{H}\mathbf{w})$$

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

# Closed-form solution




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

Solve for $\mathbf{w}$:

$$-2H^T y + 2H^T H\hat{w} = 0$$

$$H^T H\hat{w} = H^T y$$

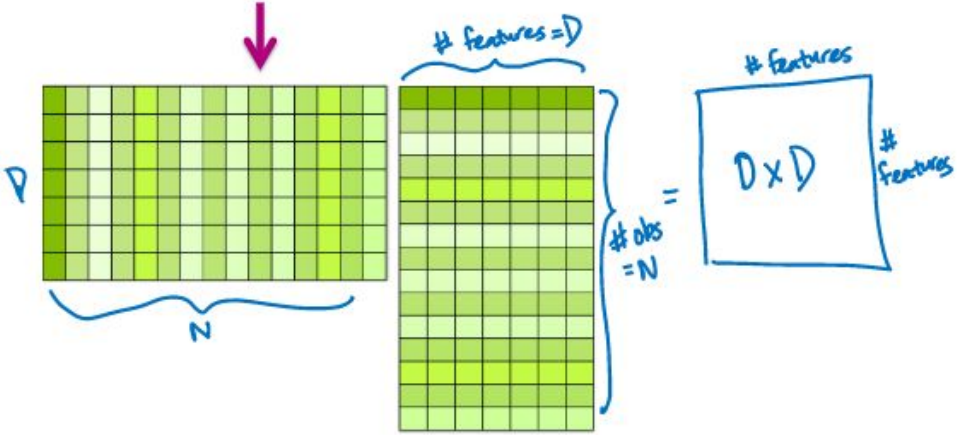
$$\underbrace{(H^T H)^{-1} H^T H}_{I}\hat{w} = (H^T H)^{-1} H^T y$$

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

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

# Closed-form solution

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




really, # of linearly ind. observations

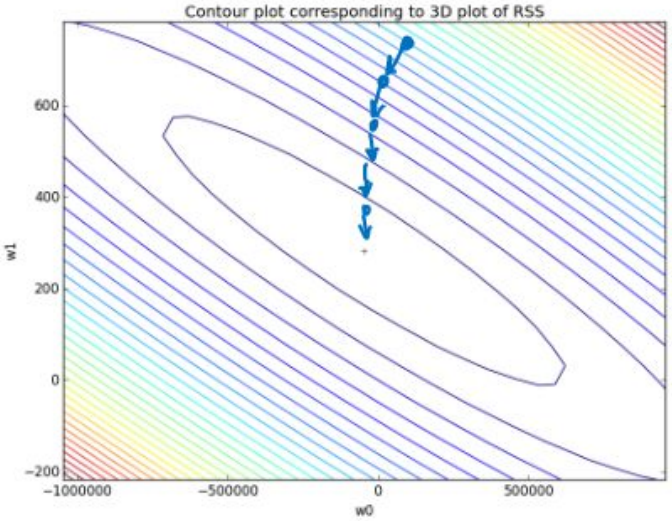
Invertible if: $N > D$

In most cases is

Complexity of inverse:

$$O(D^3)$$

# Gradient descent




**while** not converged

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

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

# Feature-by-feature update

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

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

Partial with respect to $w_j$

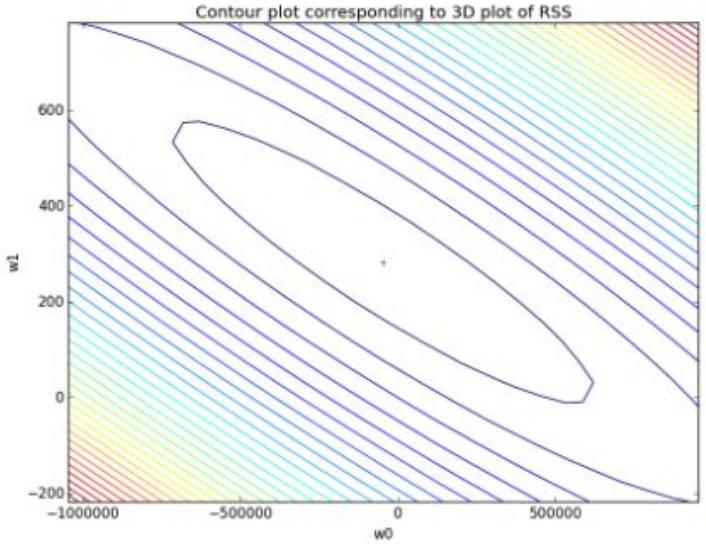
$$\sum_{i=1}^{N} 2(y_i - w_0 h_0(x_i) - w_1 h_1(x_i) \dots - w_D h_D(x_i))^1 \cdot (-h_j(x_i))$$

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

Update to $j^{th}$ feature weight:

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

# 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

$\sqrt{\text{partial}[0]^2 + \ldots + \text{partial}[D]^2}$

  **for** $j=0,\ldots,D$

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

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

  $t \leftarrow t + 1$