

INTRODUCTION TO DATA SCIENCE

This lecture is
based on course by E. Fox and C. Guestrin, Univ of Washington

15/10, 22/10,
29/10 2019

WFAiS UJ, Informatyka Stosowana
I stopień studiów

Regression for predictions

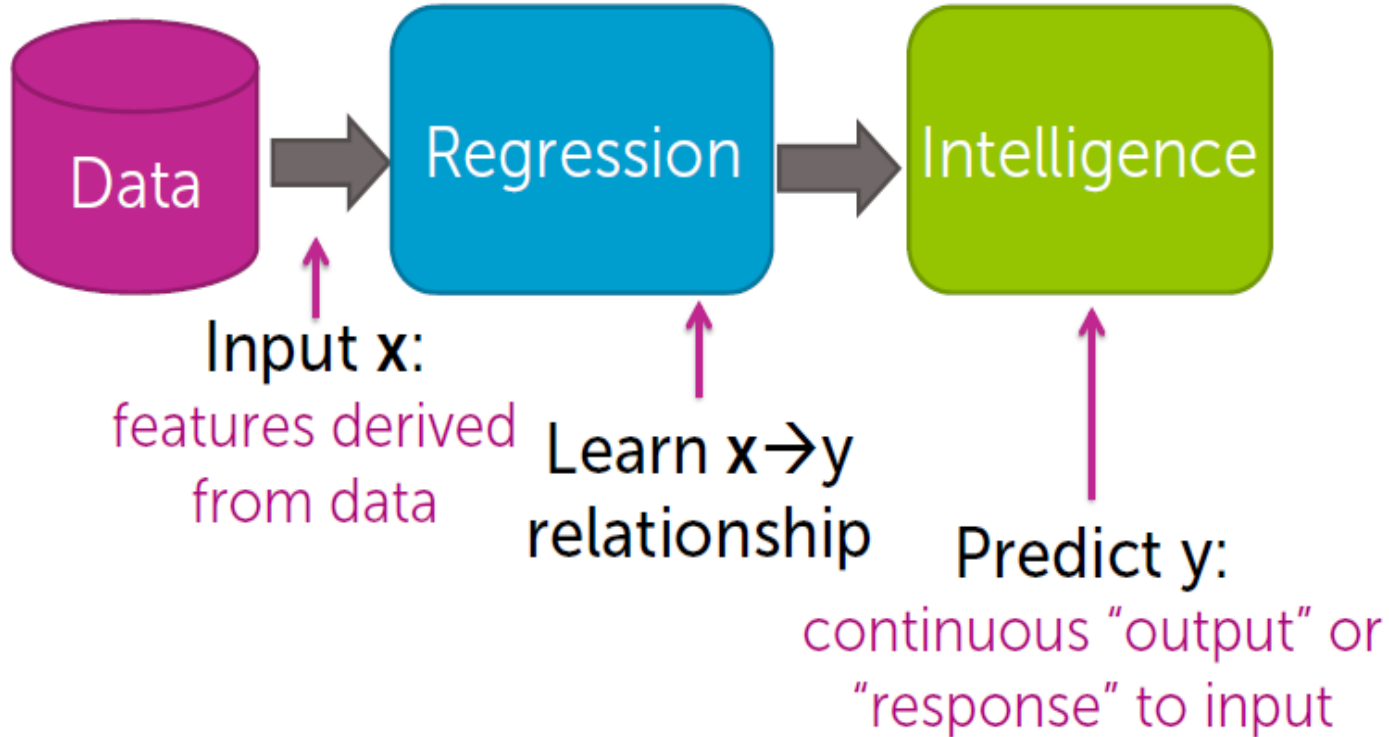
2

- ❑ **Simple regression**
- ❑ **Multiple regression**
- ❑ **Assesing performance**
- ❑ **Ridge regression**
- ❑ **Feature selection and lasso regression**
- ❑ **Nearest neighbor and kernel regression**

What is regression?

3

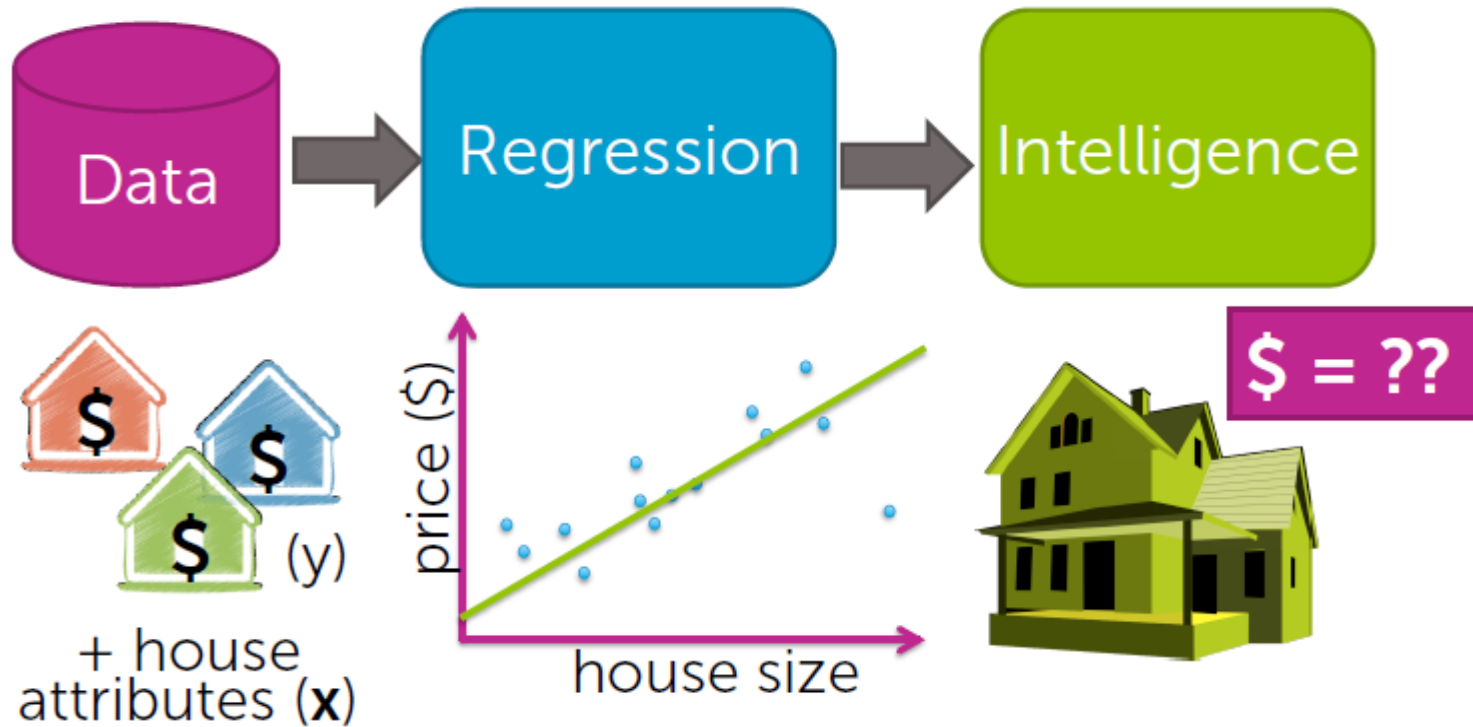
From features to predictions



Case study

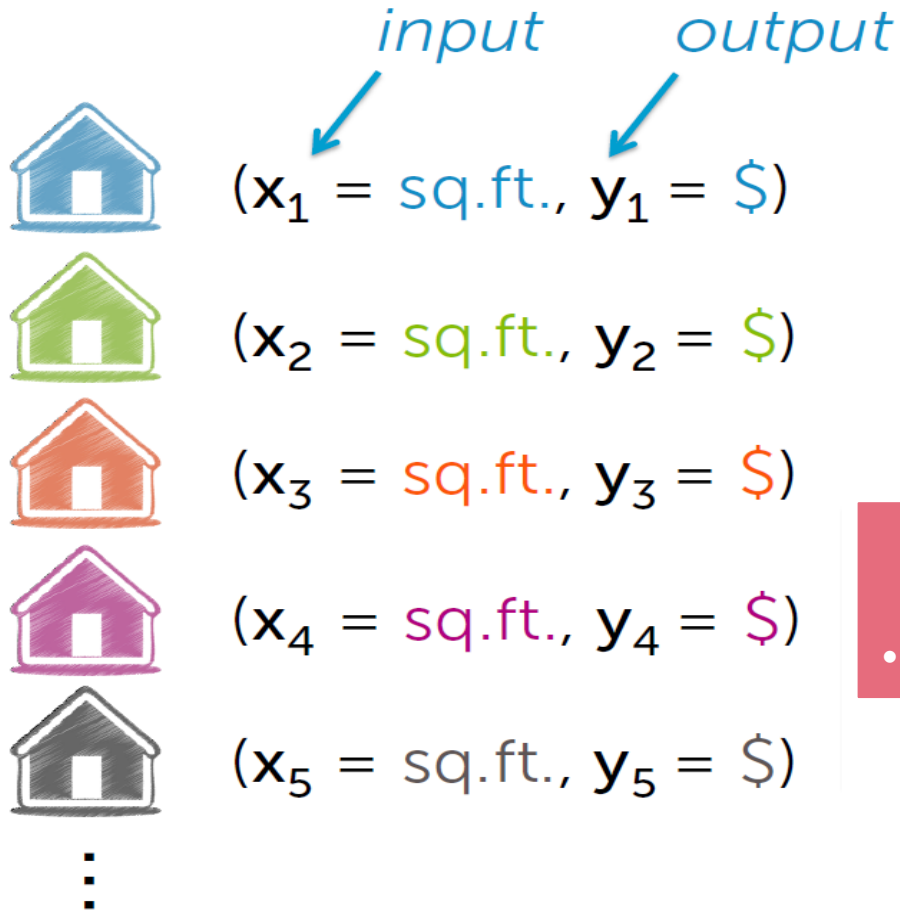
4

Predicting house prices



Data

5

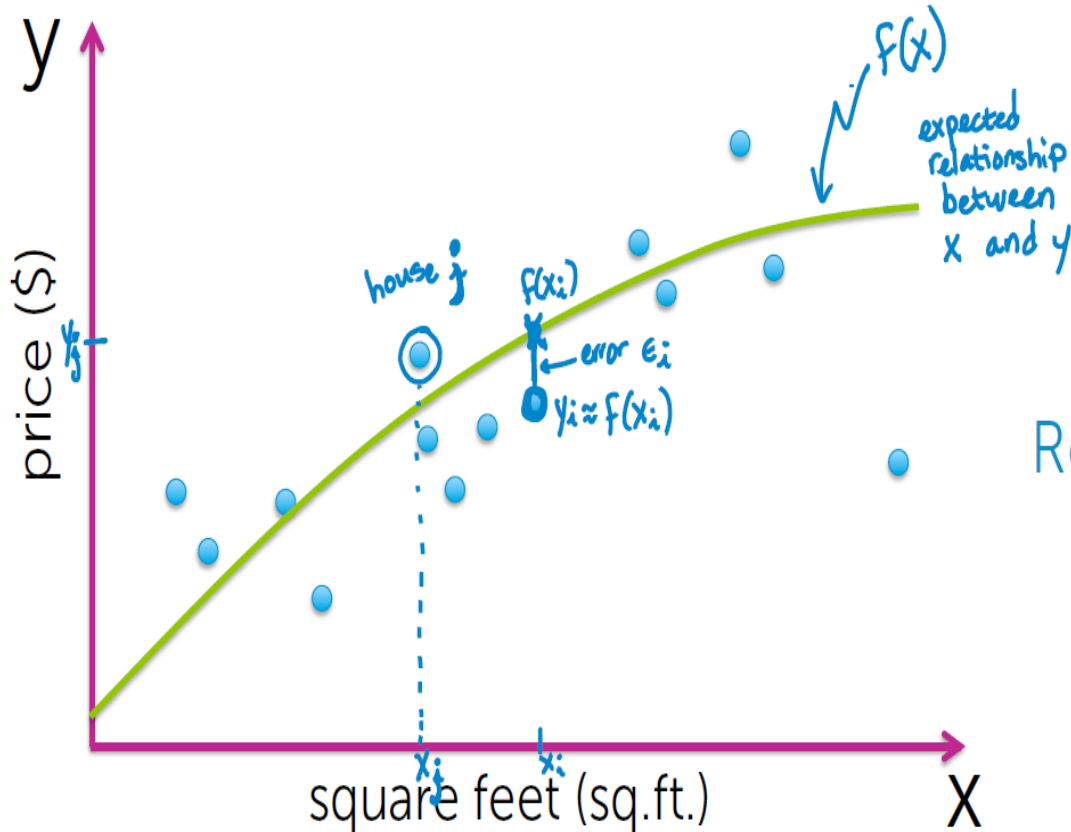


Input vs output

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

Model: assume functional relationship

6



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

Regression model:

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

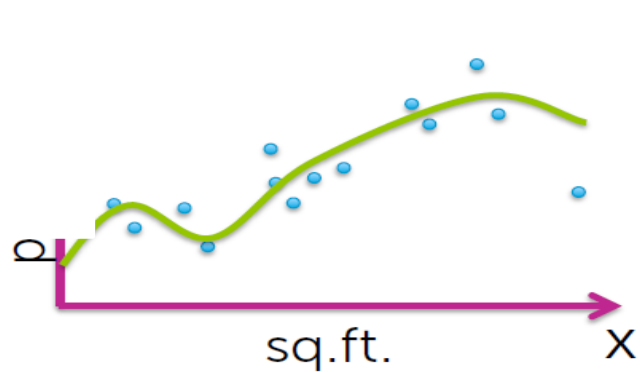
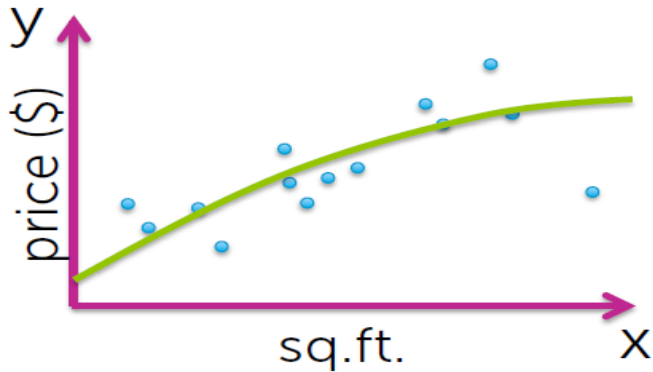
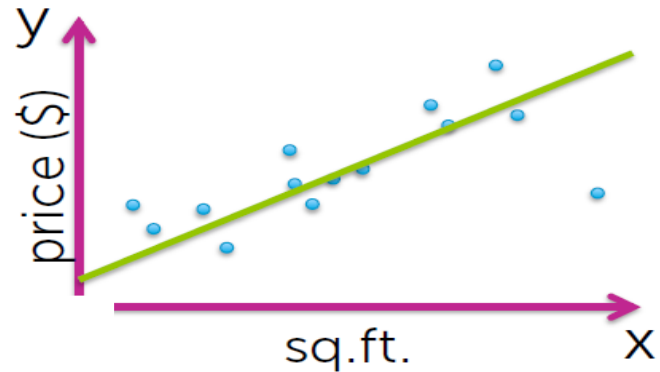
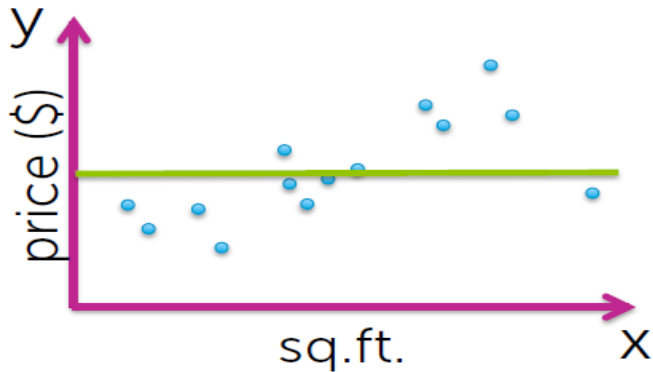
$E[\epsilon_i] = 0$ ← equally likely that error is + or -
↑ expected value

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

Task 1:

7

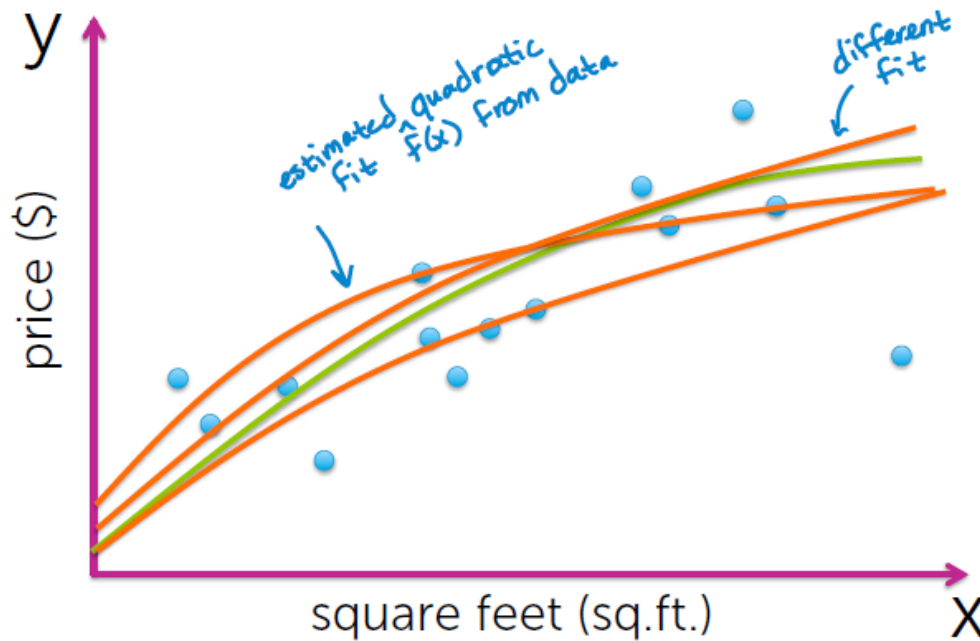
Which model to fit?



Task 2:

8

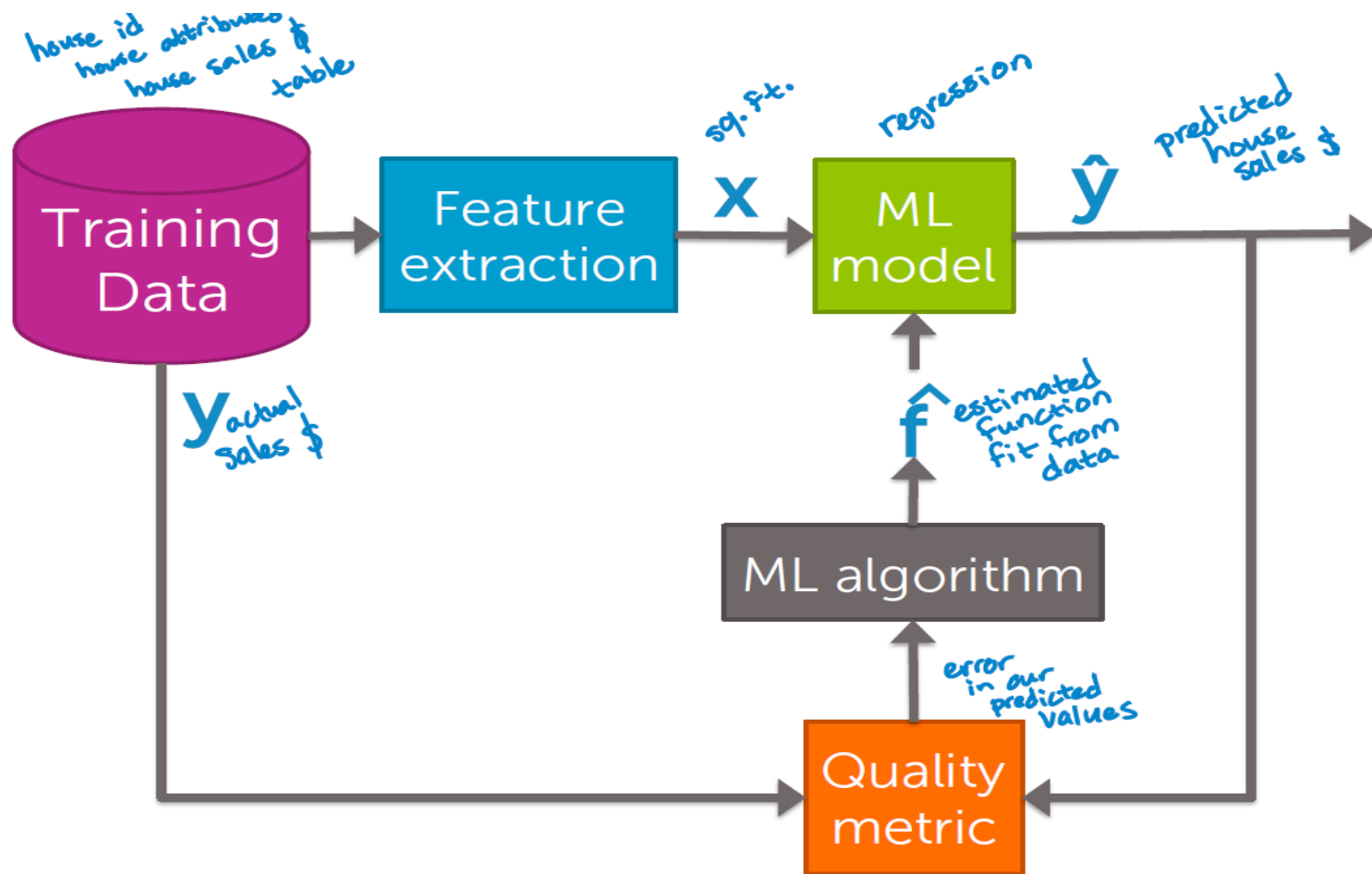
For a given model $f(x)$ estimate function $\hat{f}(x)$ from data



Assume model $f(x)$ is a quadratic function

How it works: baseline flow chart

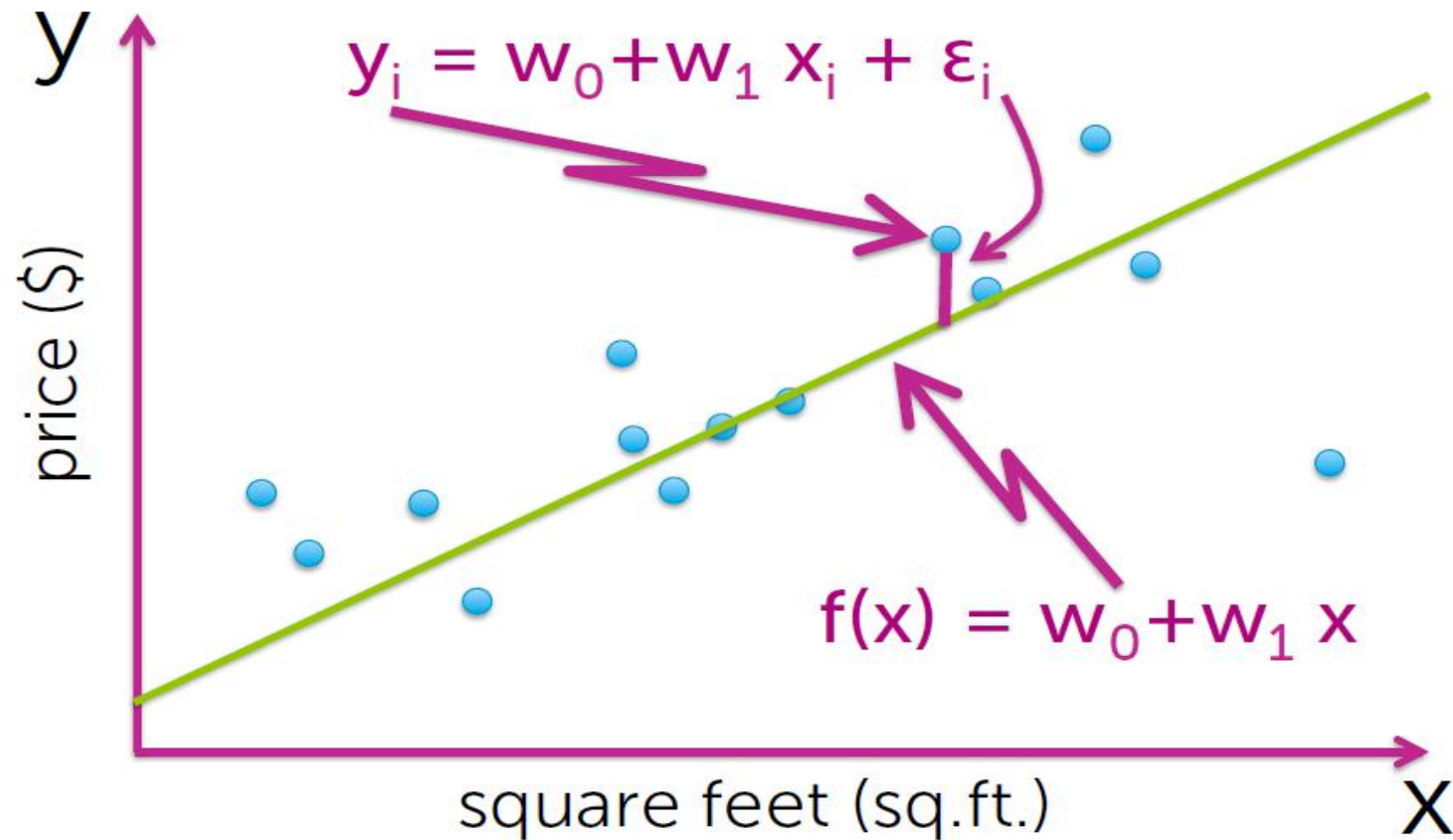
9



SIMPLE LINEAR REGRESSION

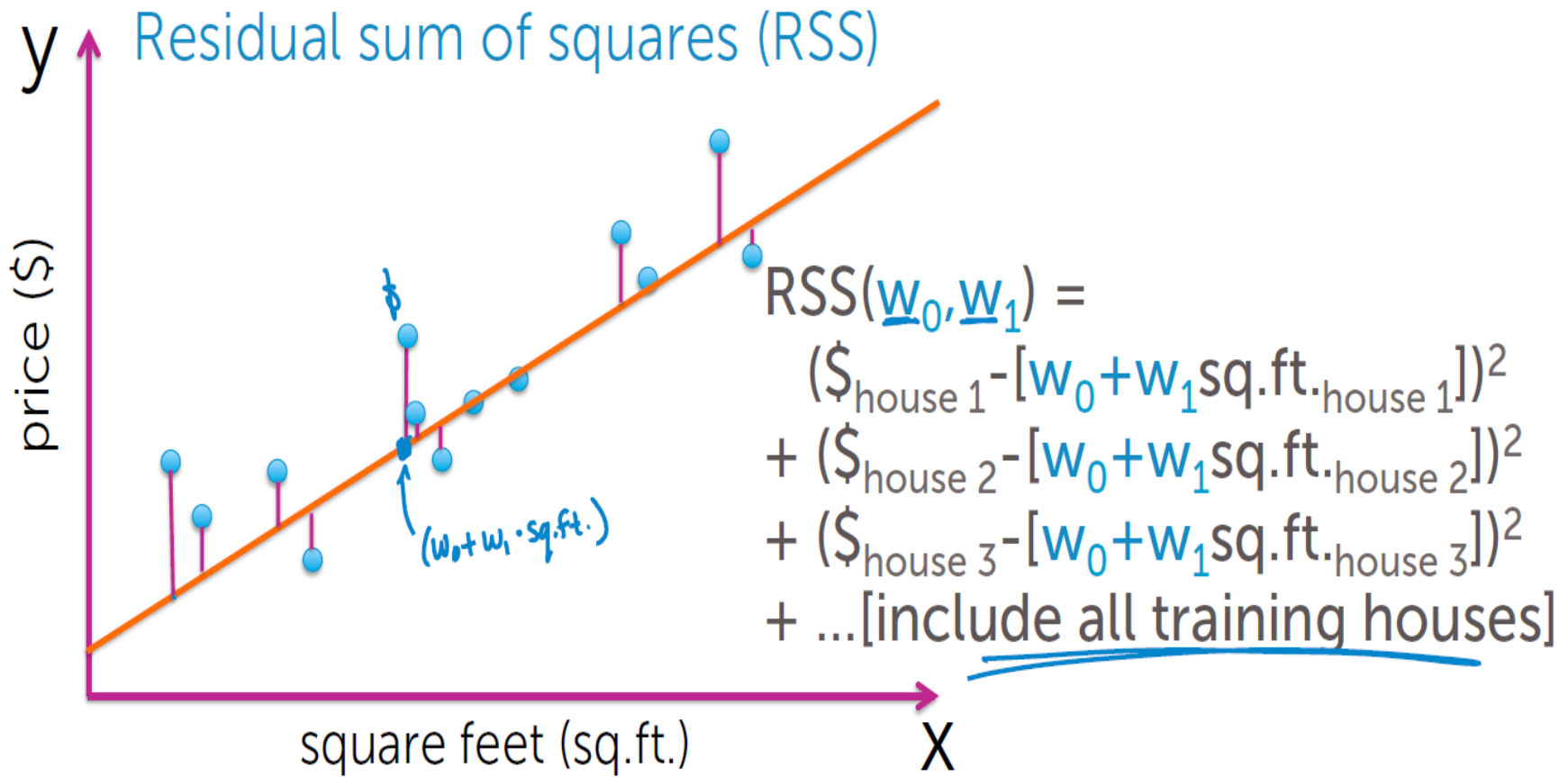
Simple linear regression model

11



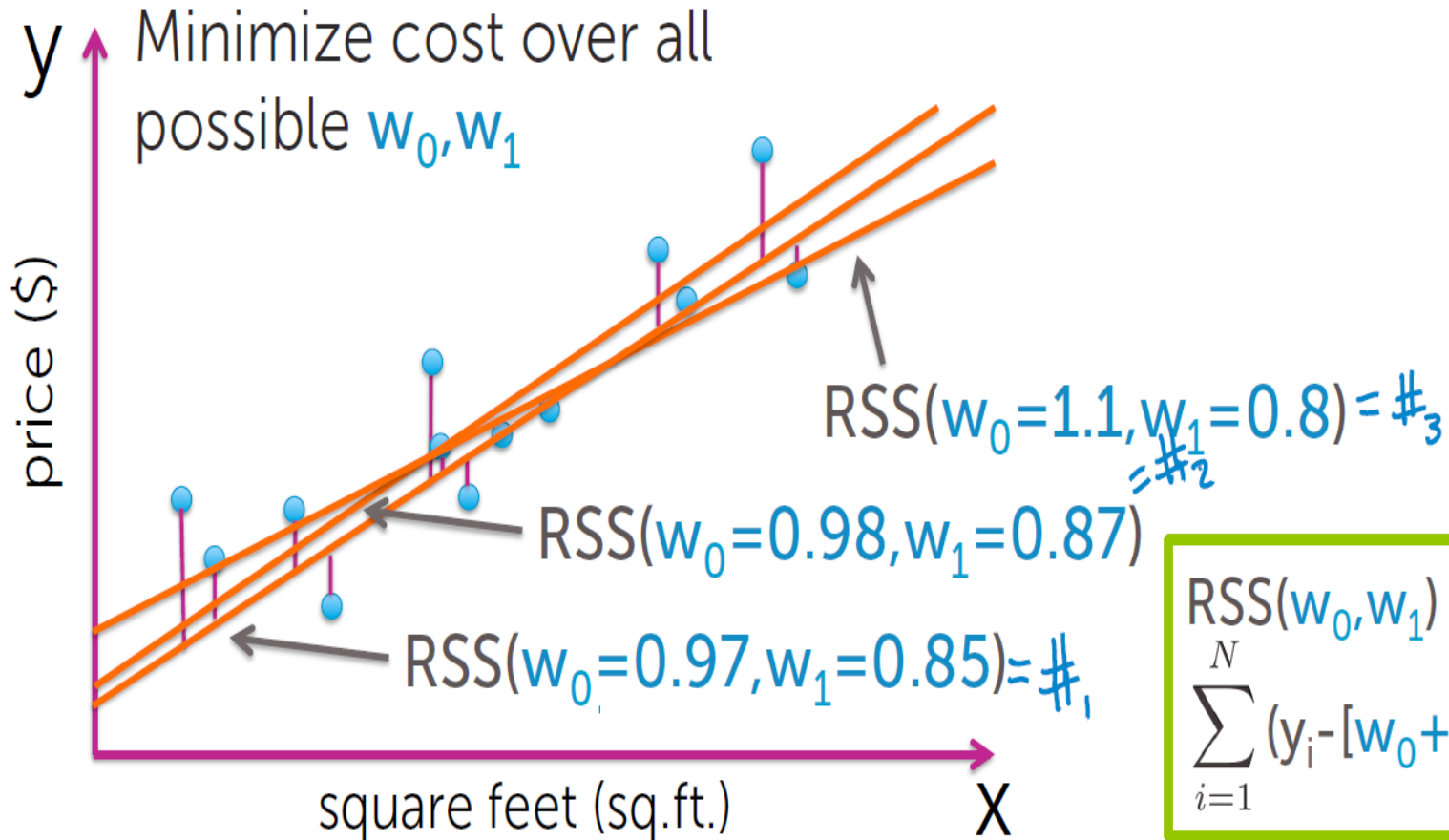
The cost of using a given line

12



Find „best” line

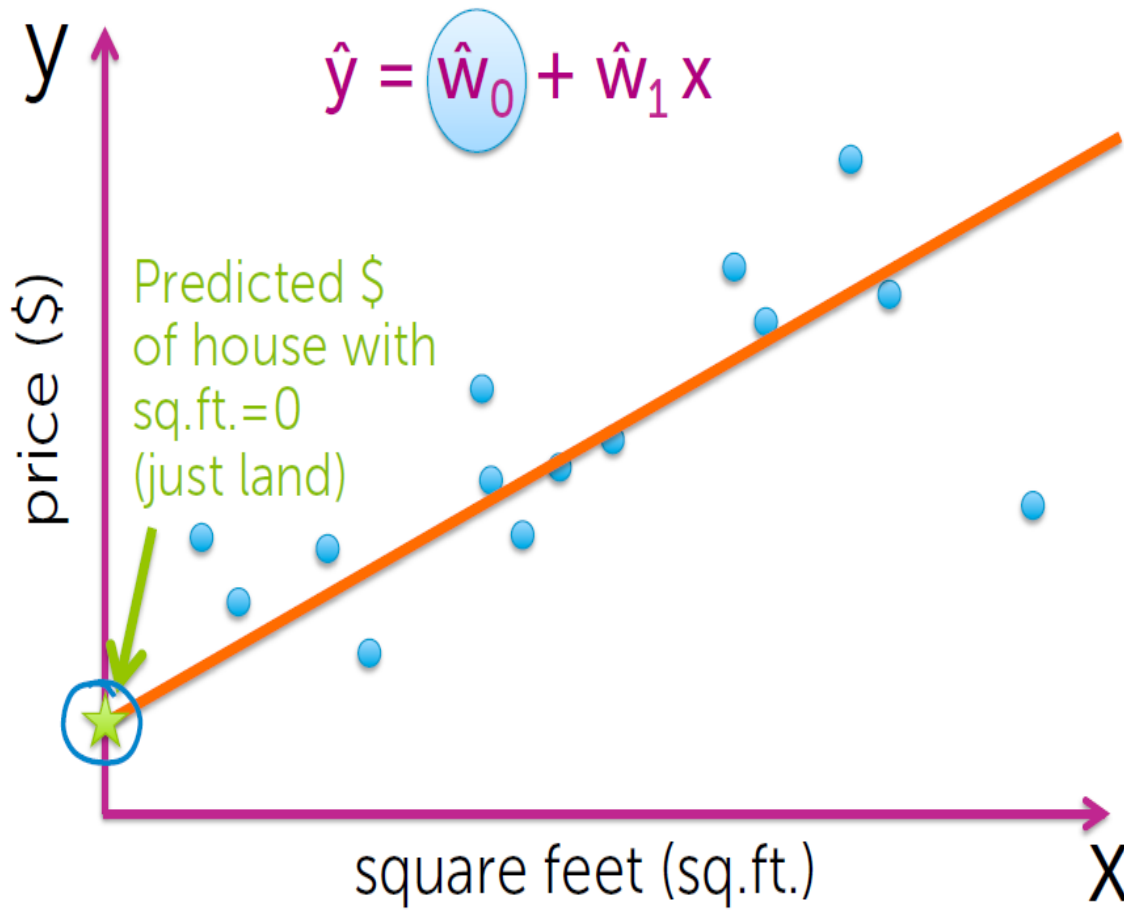
13



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

Interpreting the coefficients

15

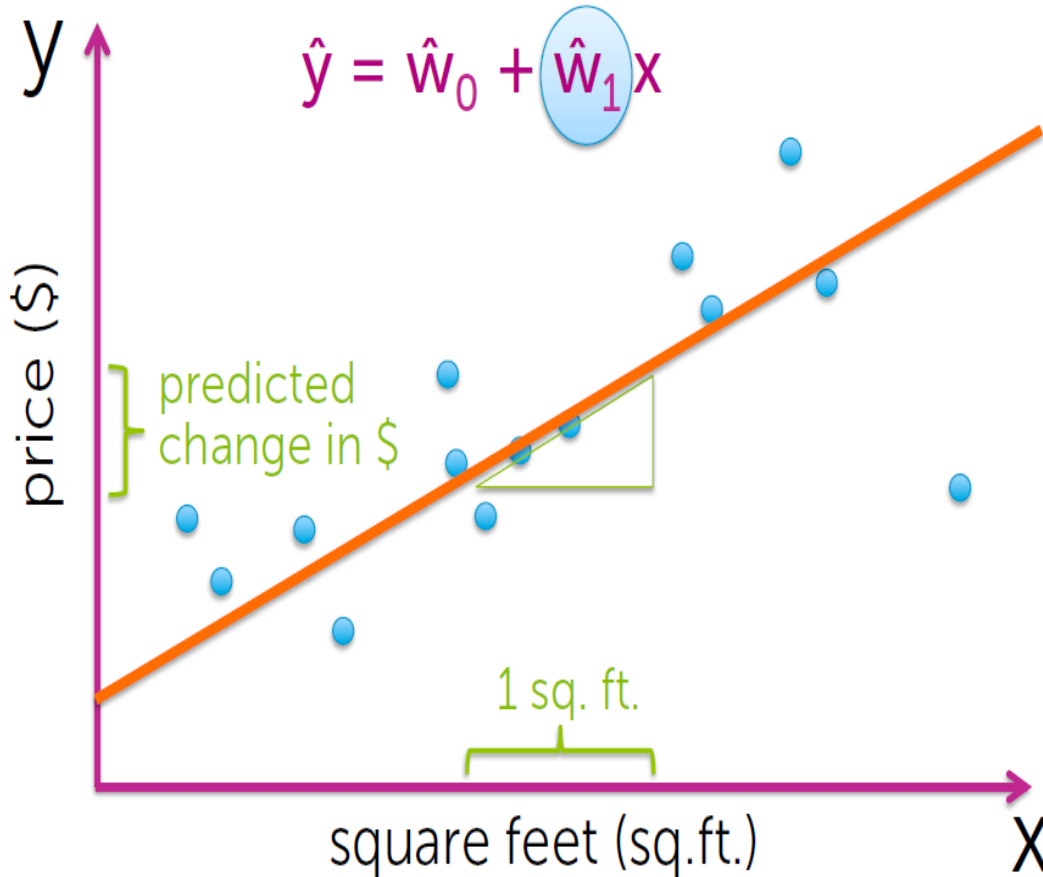


$$\hat{y} = \hat{w}_0 \text{ when } X=0$$

not very meaningful

Interpreting the coefficients

16



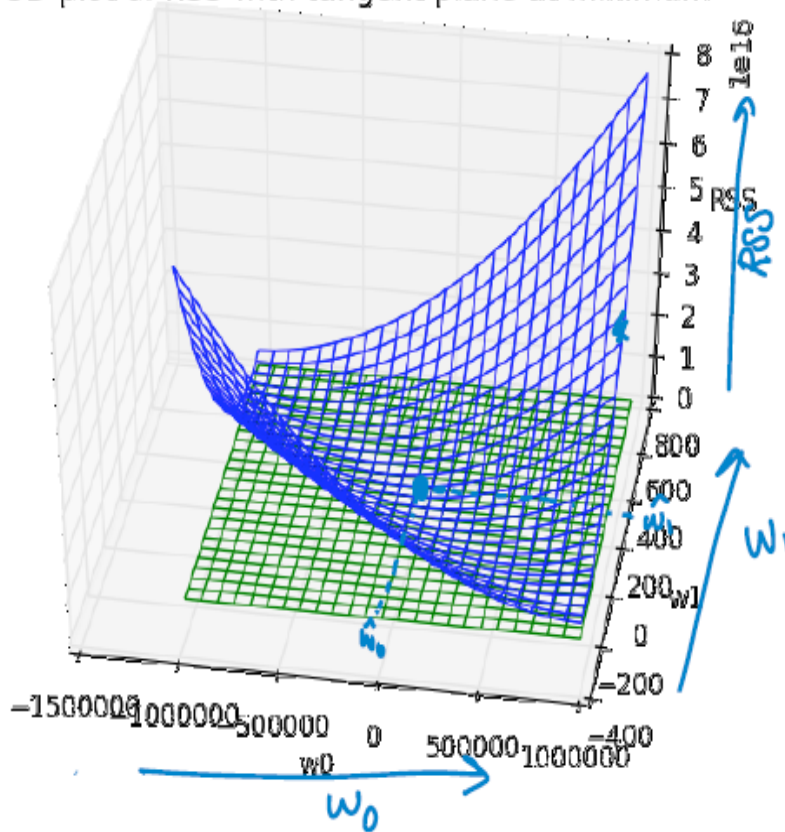
Magnitude of fit parameters depend on the units of both features and observations

$$\begin{aligned} & \hat{\$}_{1001 \text{ sq.ft.}} - \hat{\$}_{1000 \text{ sq.ft.}} \\ &= \hat{w}_0 + \hat{w}_1 \cdot 1001 \text{ sq.ft.} \\ & \quad - (\hat{w}_0 + \hat{w}_1 \cdot 1000 \text{ sq.ft.}) \\ &= \hat{w}_1 \\ & \text{predicted change in the output} \\ & \quad \text{per unit change in input} \end{aligned}$$

ML algorithm: minimising the cost

17

3D plot of RSS with tangent plane at minimum



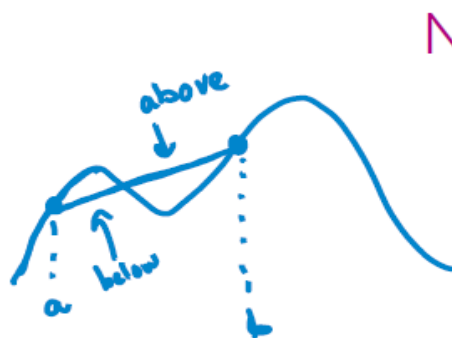
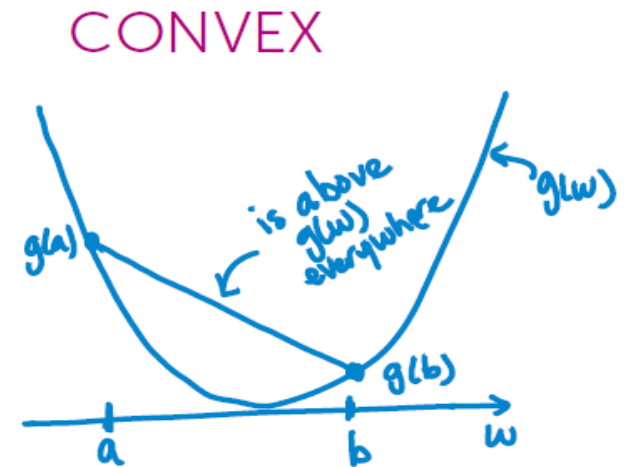
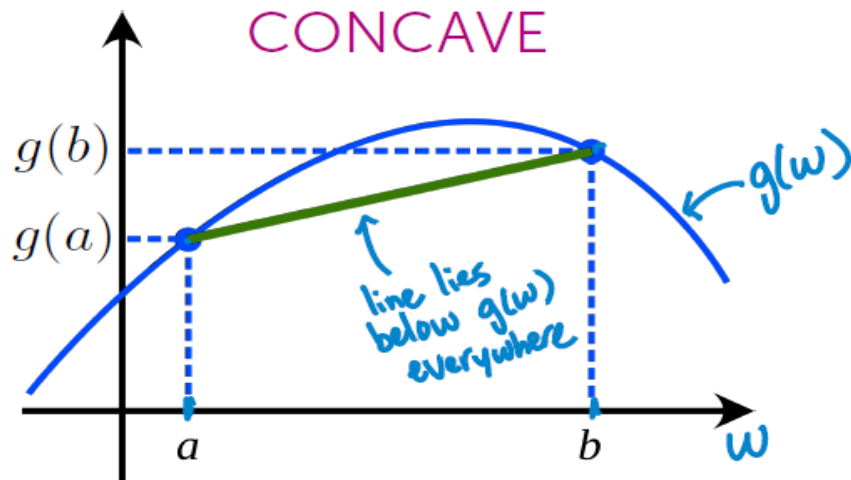
Minimize function
over all possible w_0, w_1

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

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

Convex/concave function

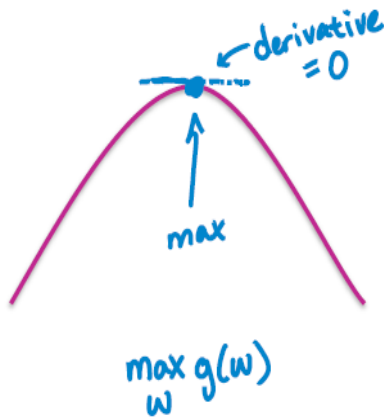
18



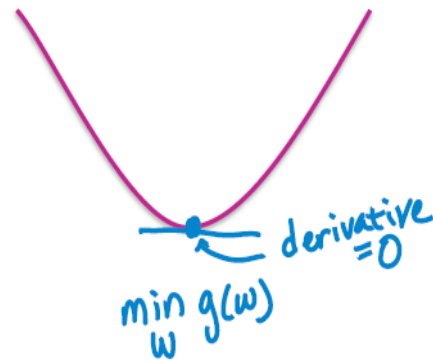
Finding max/min analytically

19

CONCAVE



CONVEX



NEITHER



no solution to derivative = 0



Example:

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

$$\frac{dg(w)}{dw} = 0 - 2(w-10)^1 \cdot 1 = -2w + 20$$

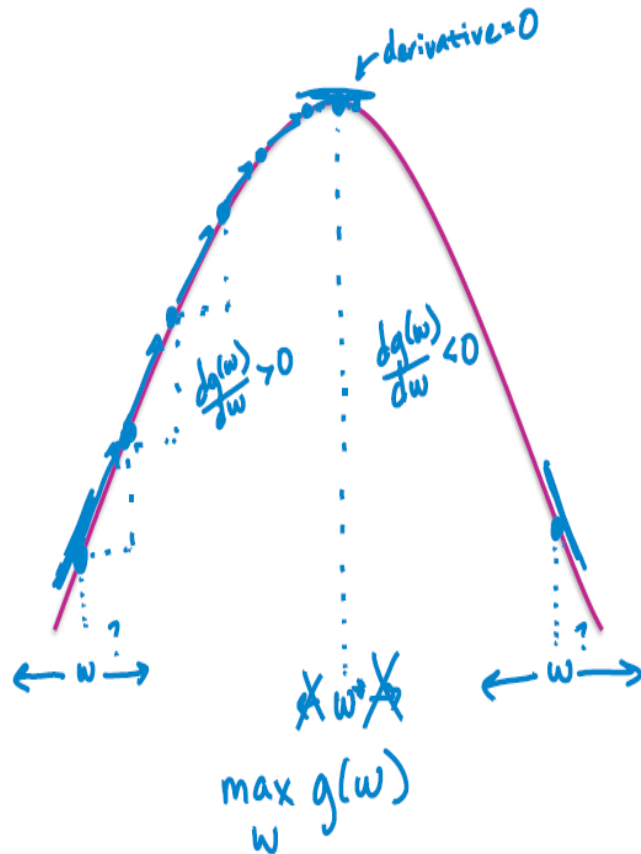
set derivate = 0:

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



Finding the max via hill climbing

20



Sign of the derivative is saying me what I want to do :move left or right or stay where I am

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

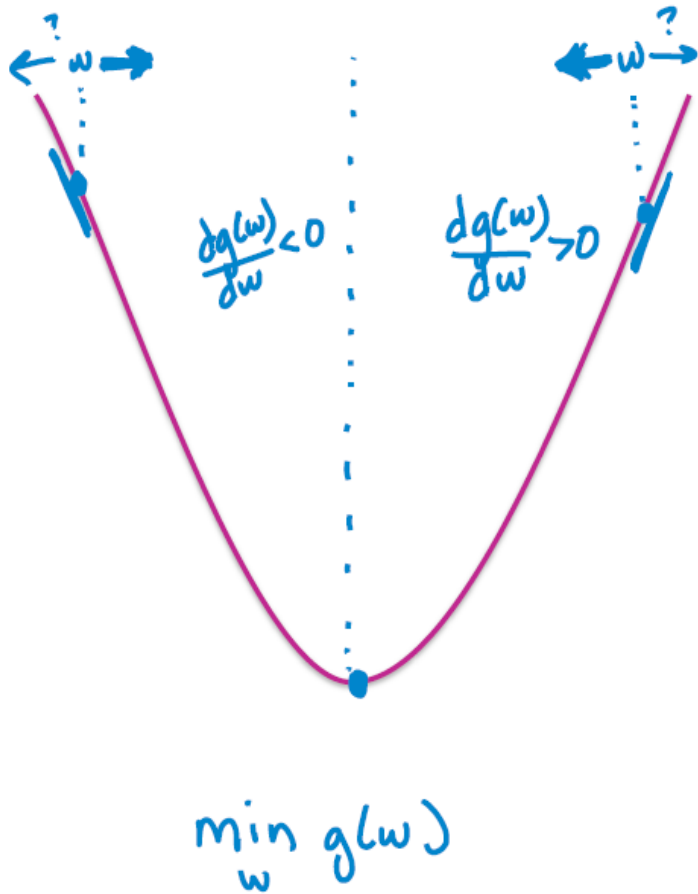
While not converged

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

iteration t stepsize

Finding the min via hill descent

21



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

Algorithm:

while not converged

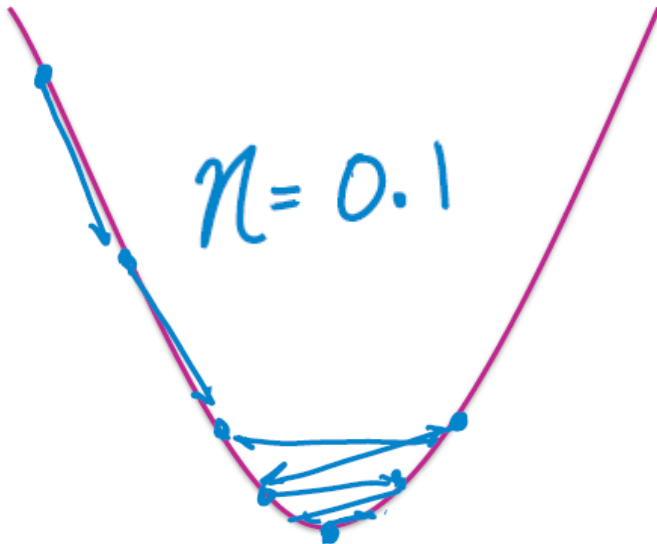
$$w^{(t+1)} \leftarrow w^{(t)} - \eta \left. \frac{dg}{dw} \right|_{w^{(t)}}$$

Choosing the step size (stepsize schedule)

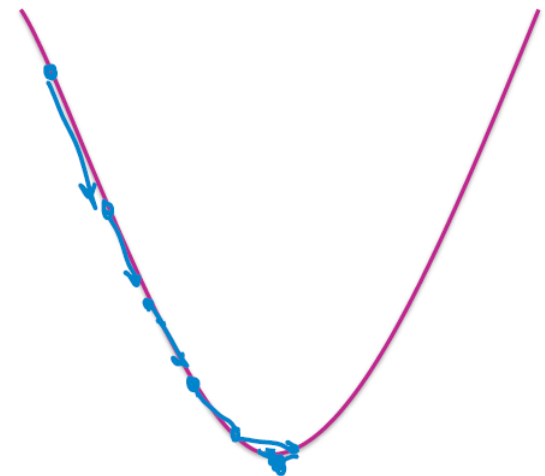
22

Fixed

Works well for strongly convex functions

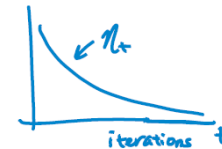


Varying



Common choices:

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



Try not to decrease η too fast

Convergence criteria

23

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

That will be „good enough”
value of ϵ depends on the data we are looking at

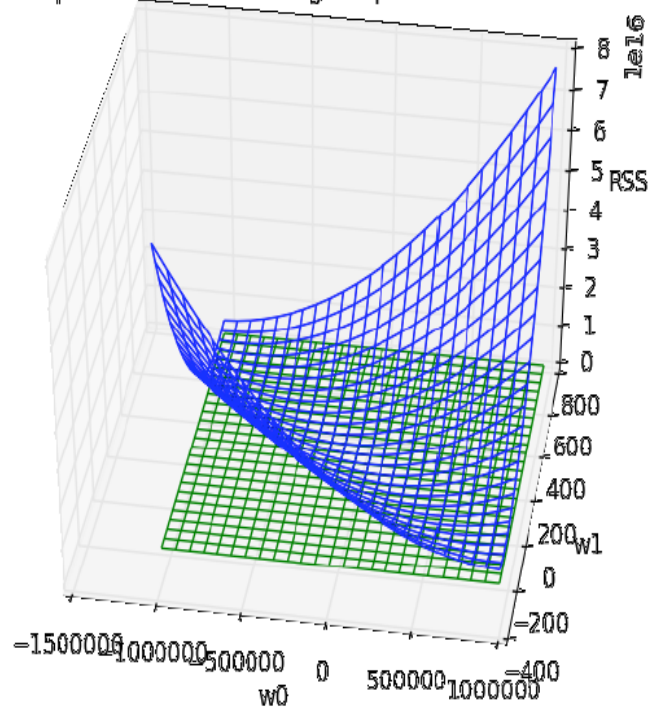
Algorithm:

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

Moving to multiple dimensions

24

3D plot of RSS with tangent plane at minimum



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

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

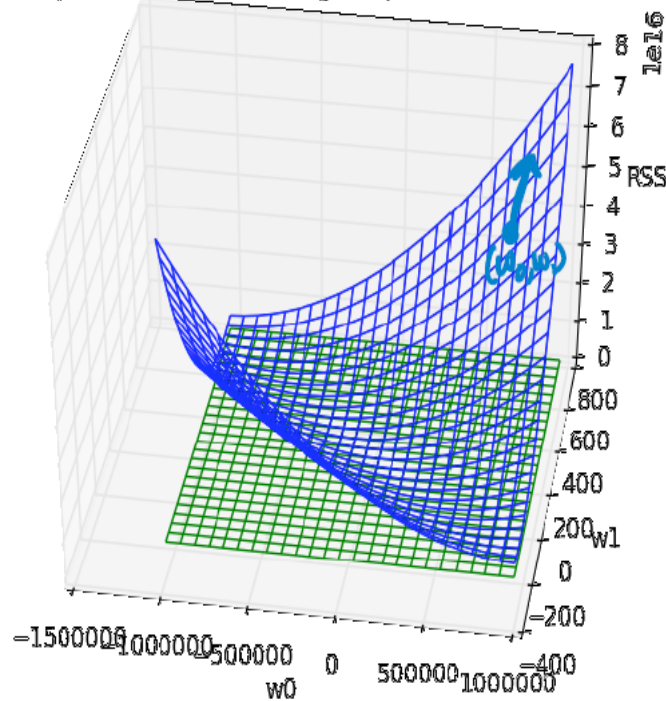
$(p+1)$ -dimensional vector

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

Gradient example

25

3D plot of RSS with tangent plane at minimum



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

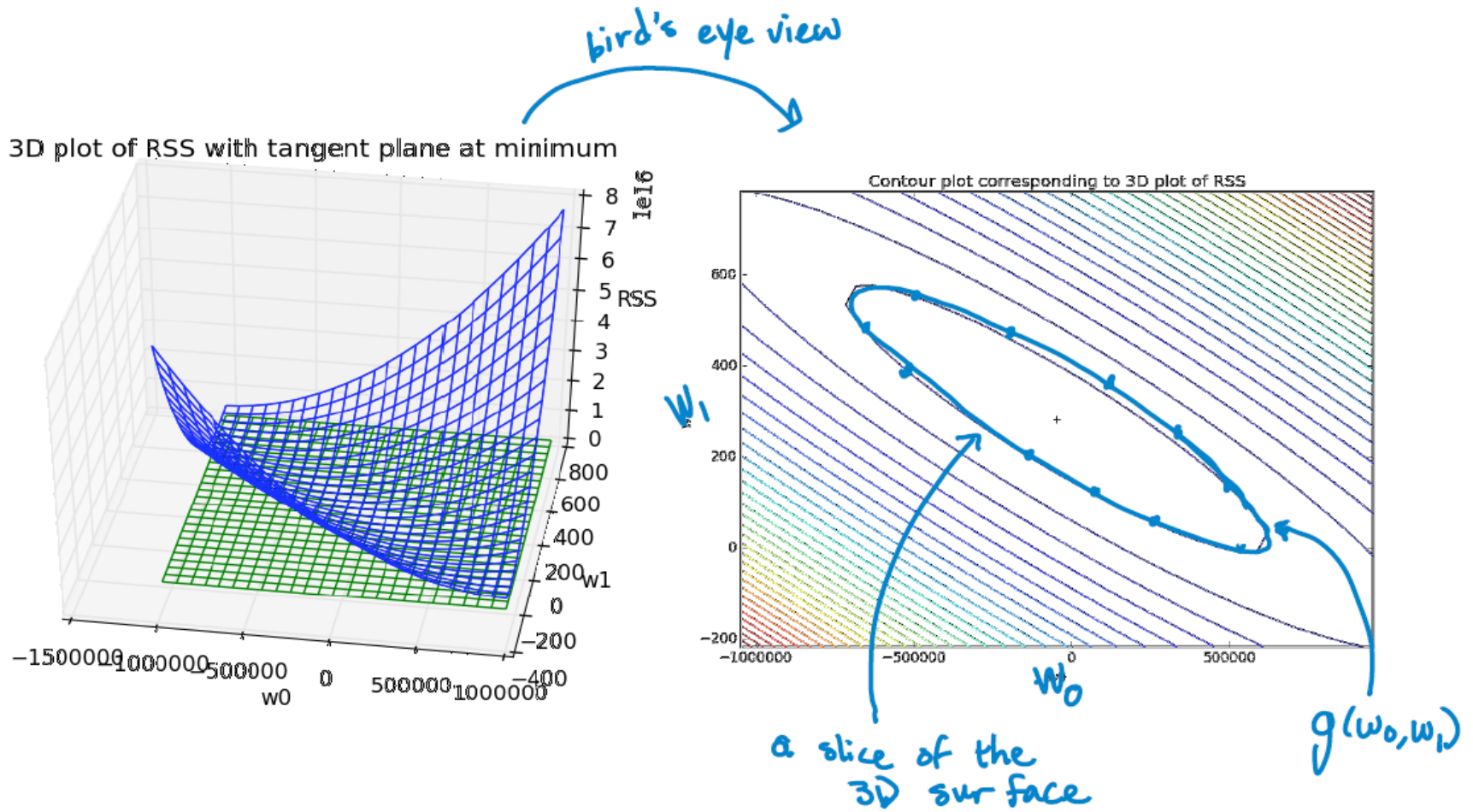
$$\frac{\partial g}{\partial w_0} = 5 + 10w_1$$

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

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

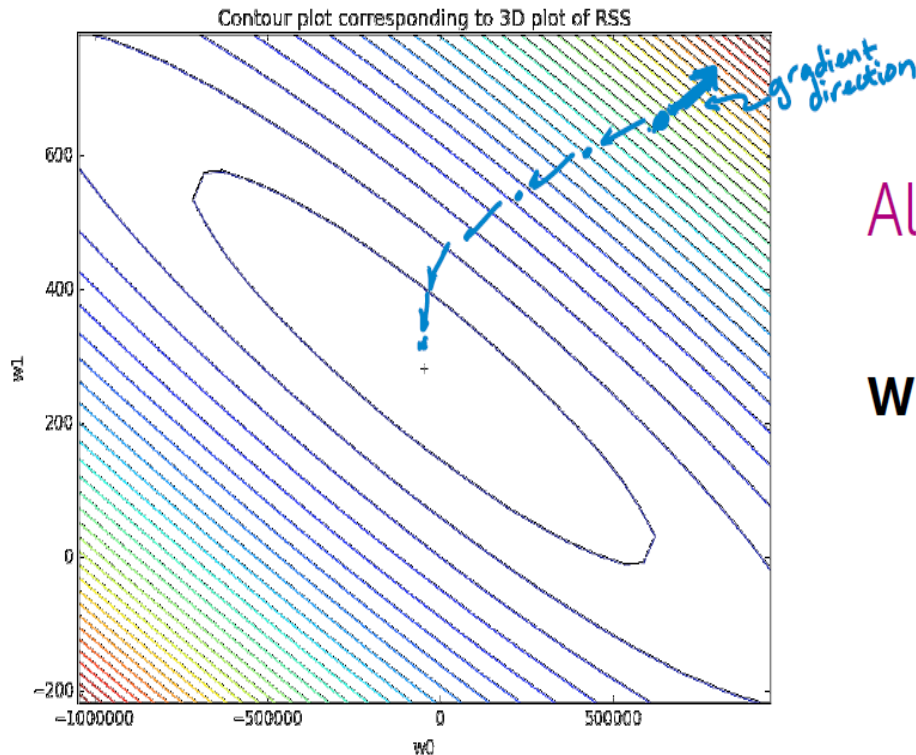
Contour plots

26



Gradient descent

27



Algorithm:

while not converged

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

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

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

Compute the gradient

28

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

Taking the derivative w.r.t. w_0

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

Putting it together:

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

Taking the derivative w.r.t. w_1

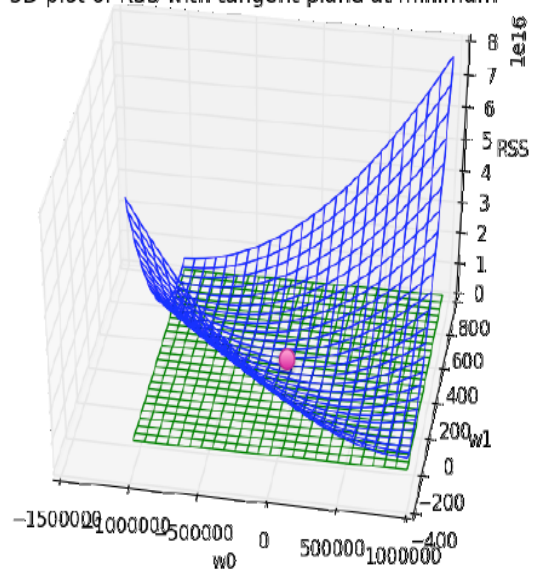
$$\begin{aligned} & \sum_{i=1}^N 2 (y_i - [w_0 + w_1 x_i])' \cdot (-x_i) \\ &= -2 \sum_{i=1}^N (y_i - [w_0 + w_1 x_i]) x_i \end{aligned}$$

Approach 1: set gradient to 0

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

This method is called „Closed form solution“

3D plot of RSS with tangent plane at minimum



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 $\leftarrow \frac{\sum y_i}{N}$
 estimate of the slope $\leftarrow \hat{w}_1$
 average sq-ft. $\leftarrow \frac{\sum x_i}{N}$

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

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

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

30

Interpreting the gradient:

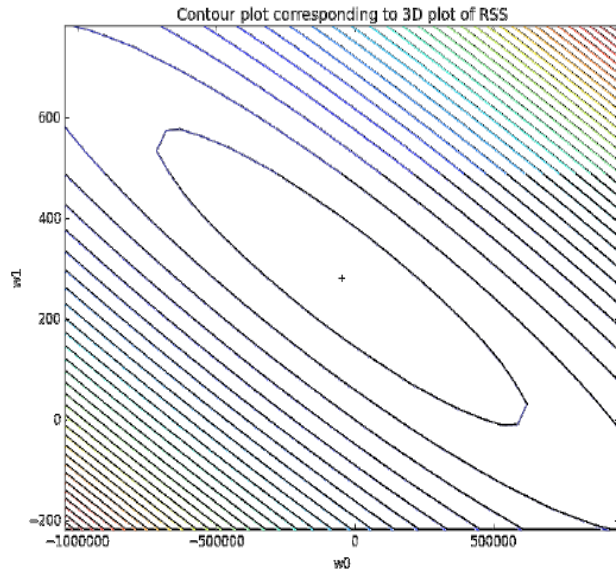
$$\nabla \text{RSS}(w_0, w_1) = \begin{bmatrix} -2 \sum_{i=1}^N [y_i - (w_0 + w_1 x_i)] \\ -2 \sum_{i=1}^N [y_i - (w_0 + w_1 x_i)] x_i \end{bmatrix} = \begin{bmatrix} -2 \sum_{i=1}^N [y_i - \hat{y}_i(w_0, w_1)] \\ -2 \sum_{i=1}^N [y_i - \hat{y}_i(w_0, w_1)] x_i \end{bmatrix}$$

actual house sales observation
predicted value $\hat{y}_i(w_0, w_1)$

Approach 2: gradient descent

31

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



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

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

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

→ w_0 is going to increase

similar intuition for w_1 , but multiply by x_i

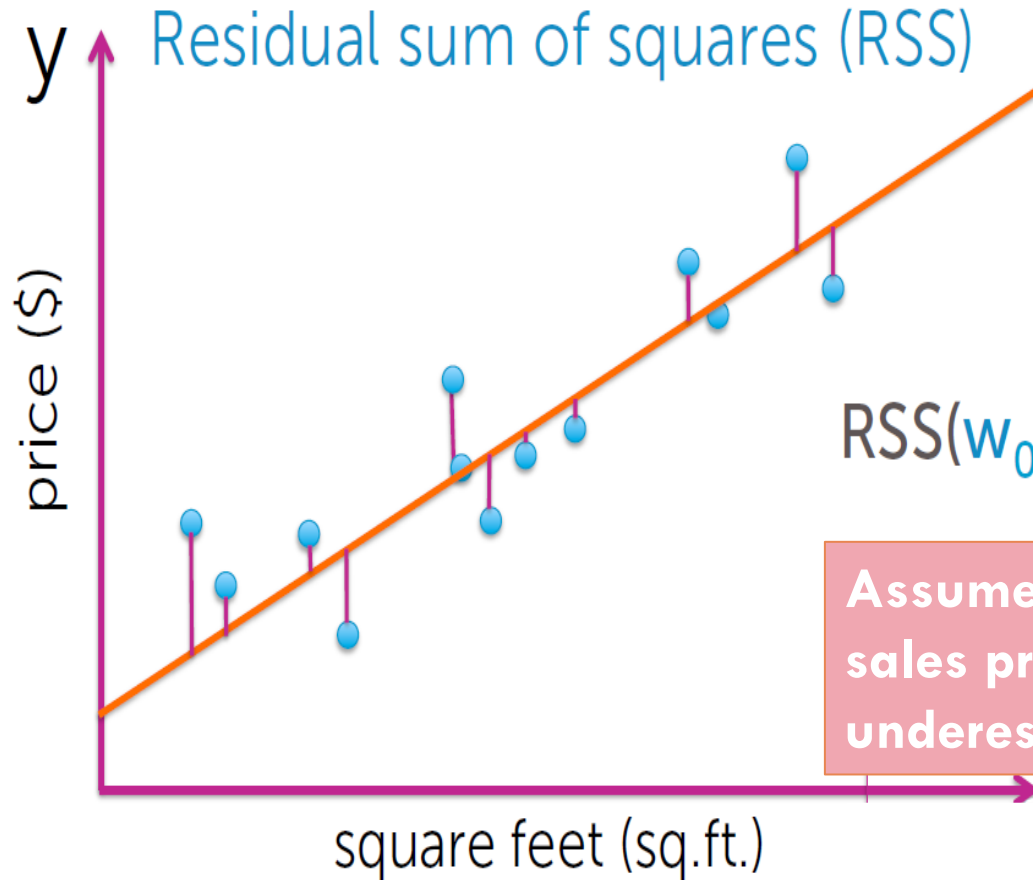
Comparing the approaches

32

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

Symmetric cost function

33

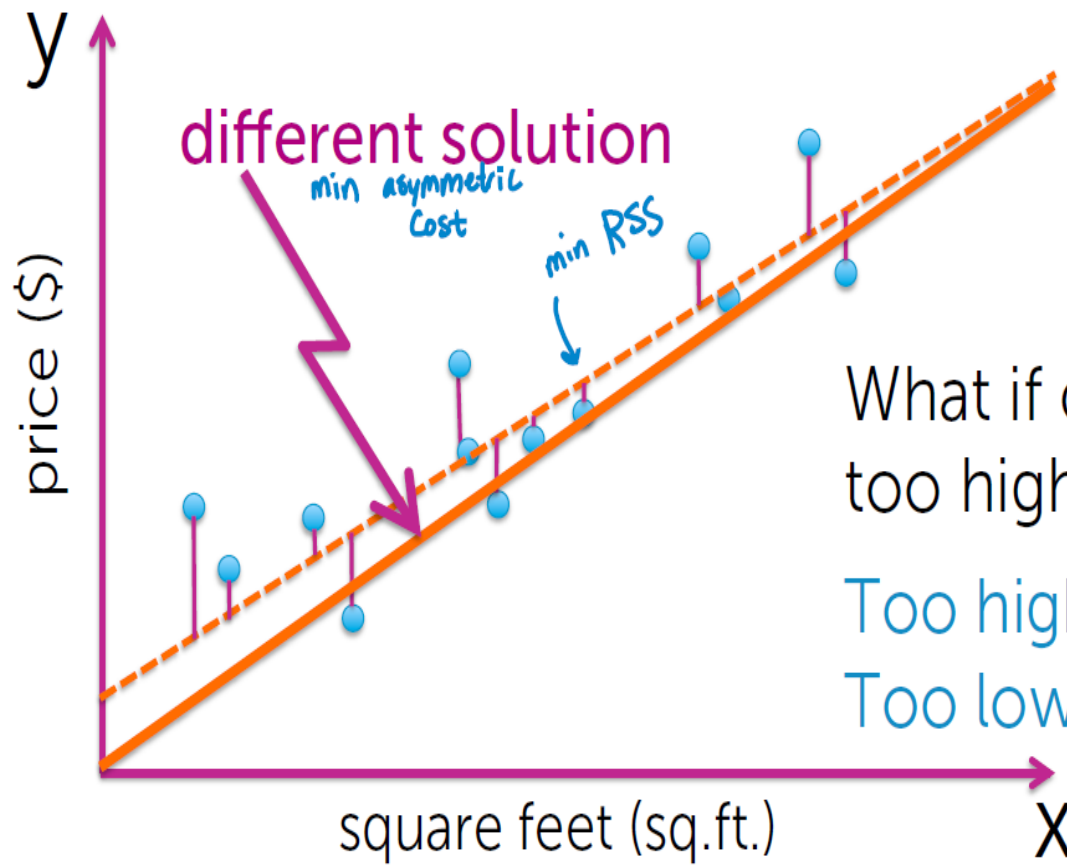


$$\text{RSS}(w_0, w_1) = \sum_{i=1}^N (y_i - [w_0 + w_1 x_i])^2$$

Assumes error of overestimating sales price is the same as underestimating it

Asymmetric cost functions

34



We can weight differently positive and negative errors in RSS calculations.

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

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

Too low \rightarrow offers for lower \$

What you can do now

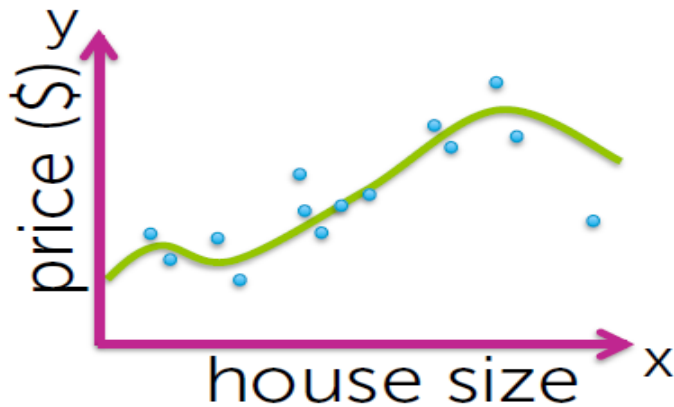
35

- 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

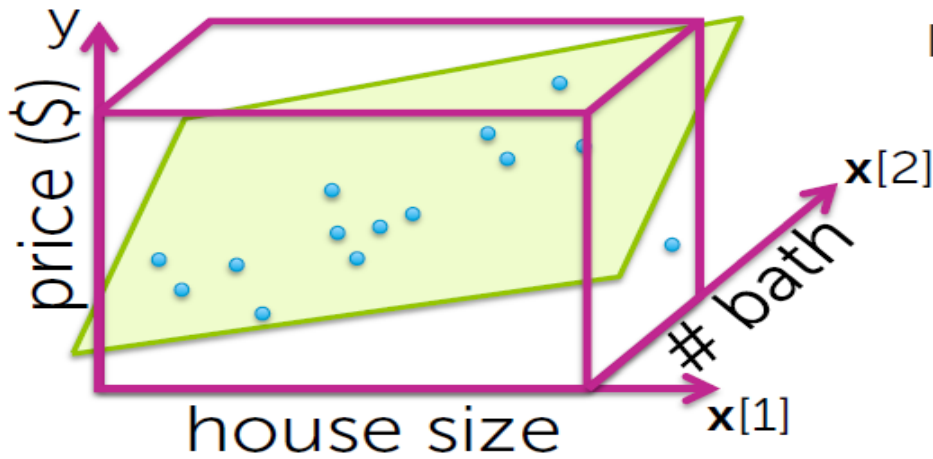
MULTIPLE REGRESSION

Multiple regression

37



Fit more complex relationships than just a line



Incorporate more inputs

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

Polynomial regression

38

Model:

$$y_i = w_0 + w_1 x_i + w_2 x_i^2 + \dots + w_p x_i^p + \epsilon_i$$

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

...

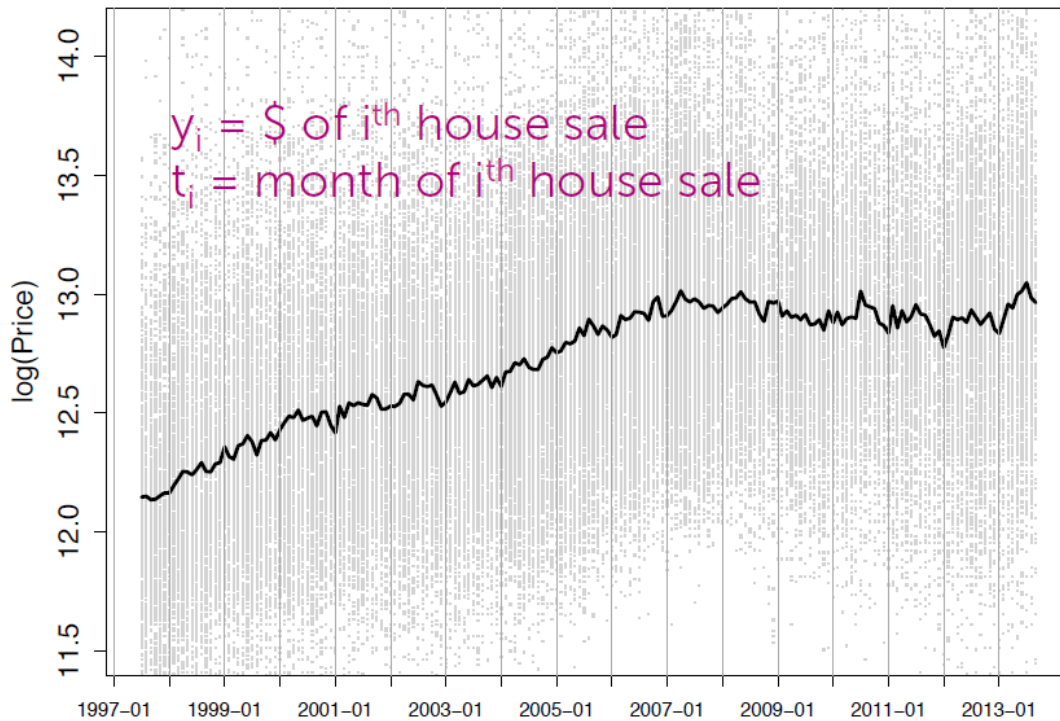
...

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

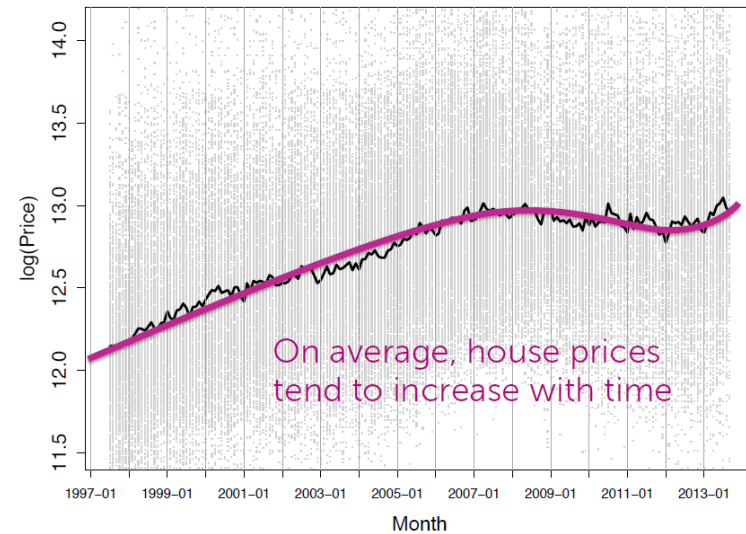
Other functional forms of one input

39

□ Trends in time series



Month ← House sales recorded monthly

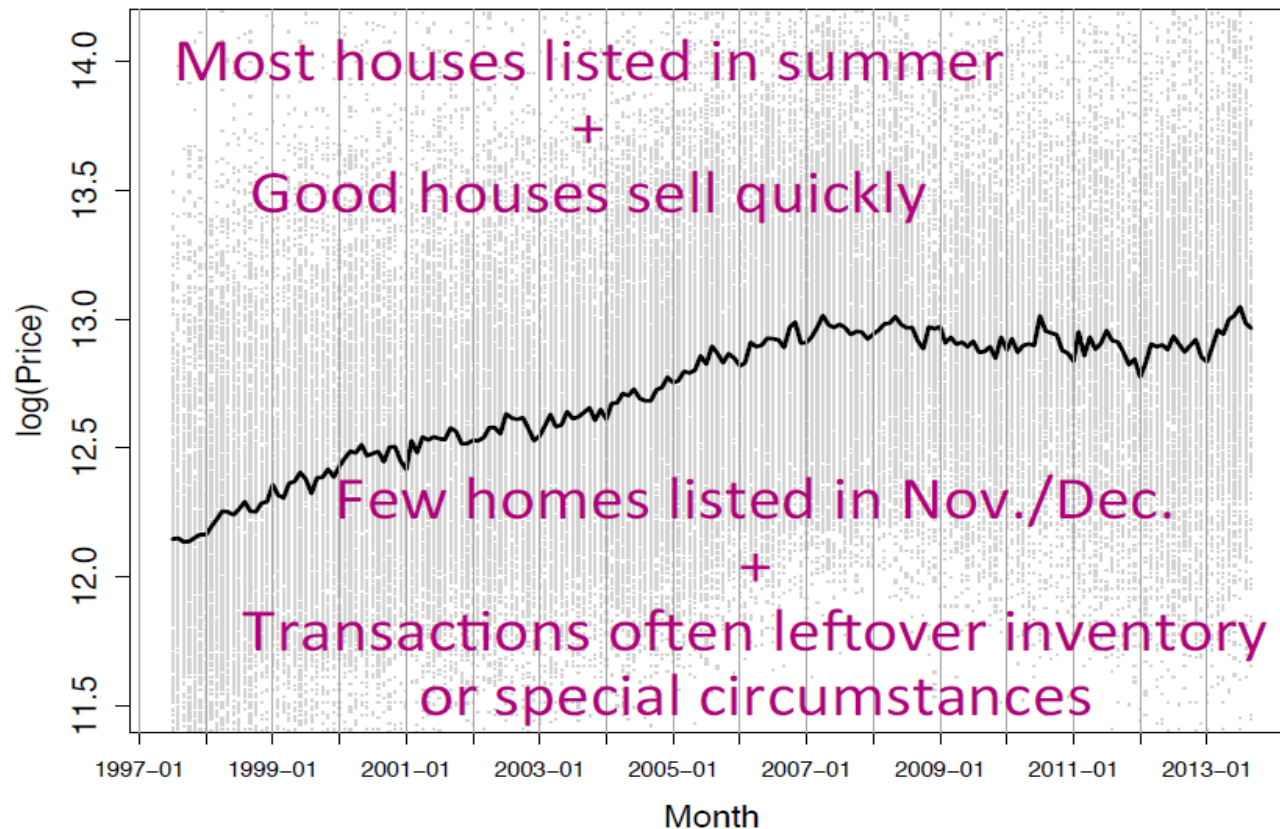


This trend can be modeled with polynomial function.

Other functional forms of one input

40

□ Seasonality



15/10, 22/10, 29/10 2019

Example of detrending

41

Model:

$$y_i = w_0 + w_1 t_i + w_2 \sin(2\pi t_i / 12 - \Phi) + \varepsilon_i$$

Linear trend

Unknown phase/shift

Seasonal component =
Sinusoid with period 12
(resets annually)



Trigonometric identity: $\sin(a - b) = \sin(a)\cos(b) - \cos(a)\sin(b)$

$$\rightarrow \sin(2\pi t_i / 12 - \Phi) = \sin(2\pi t_i / 12)\cos(\Phi) - \cos(2\pi t_i / 12)\sin(\Phi)$$

Example of detrending

42

Equivalently,

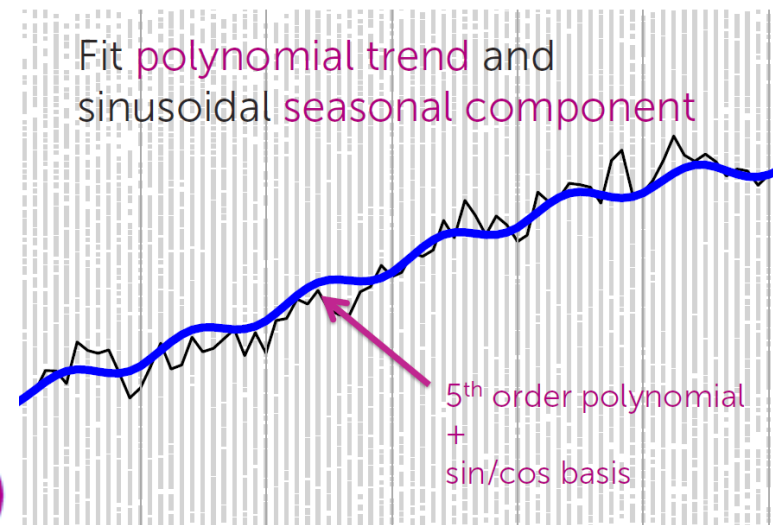
$$y_i = w_0 + w_1 t_i + w_2 \sin(2\pi t_i / 12) + w_3 \cos(2\pi t_i / 12) + \epsilon_i$$

feature 1 = 1 (constant)

feature 2 = t

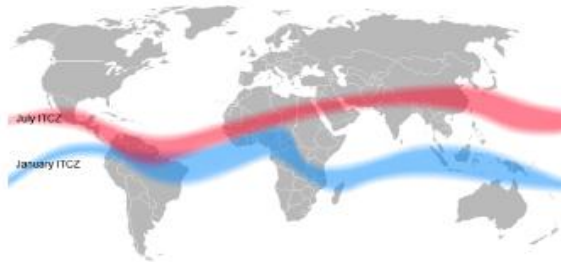
feature 3 = $\sin(2\pi t/12)$

feature 4 = $\cos(2\pi t/12)$

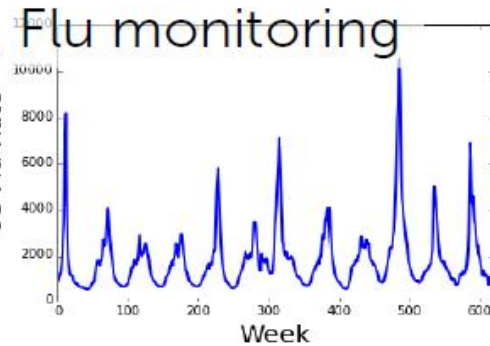


Other examples of seasonality

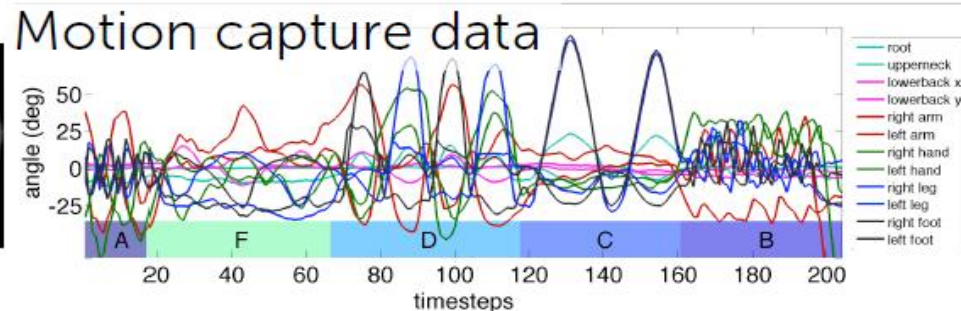
43



Weather modeling
(e.g., temperature, rainfall)



Demand forecasting
(e.g., jacket purchases)



Generic basic expansion

44

Model:

$$y_i = w_0 h_0(x_i) + w_1 h_1(x_i) + \dots + w_D h_D(x_i) + \varepsilon_i$$
$$= \sum_{j=0}^D w_j h_j(x_i) + \varepsilon_i$$

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

feature 2 = $h_1(x)$... e.g., x

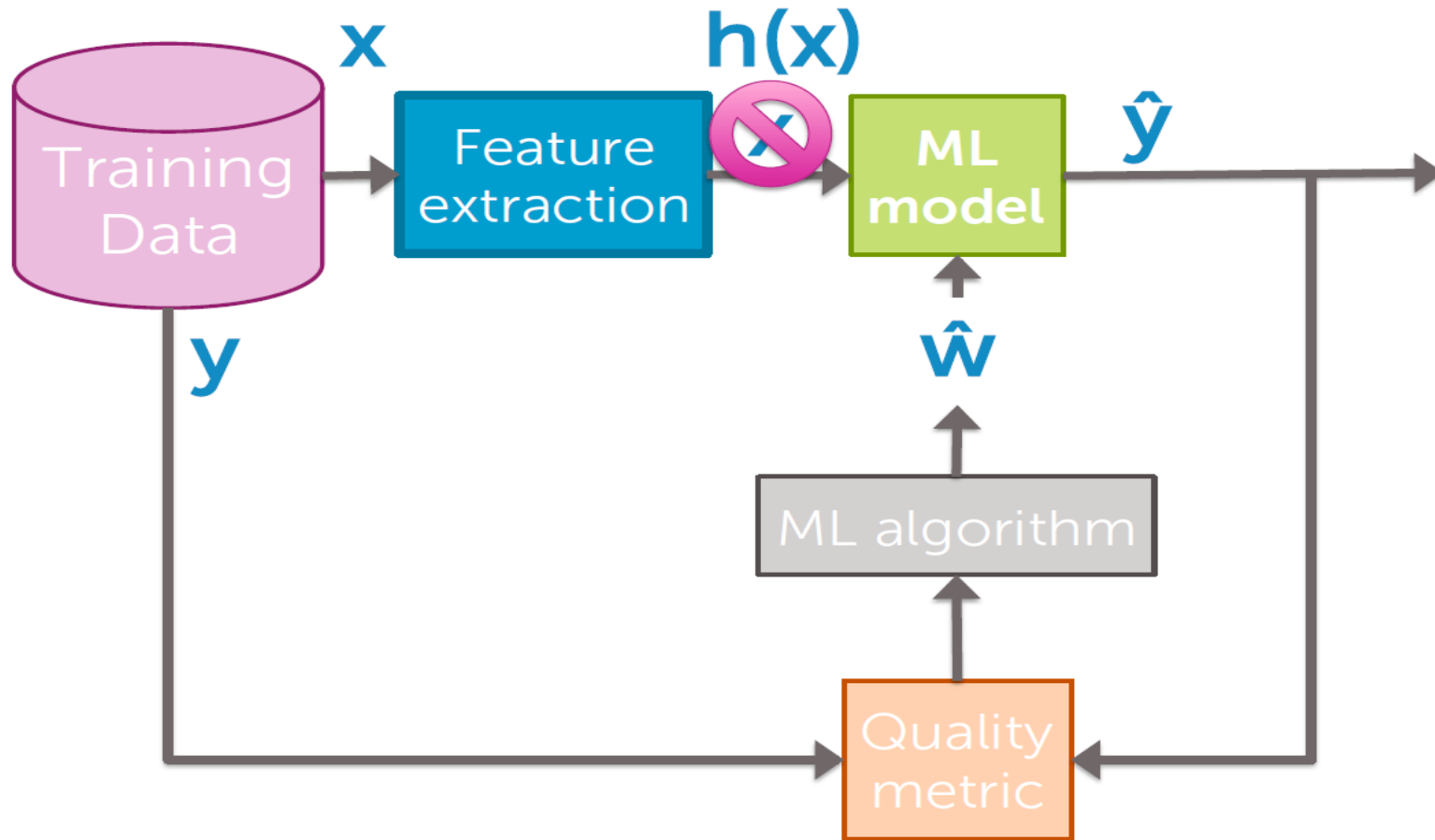
feature 3 = $h_2(x)$... e.g., x^2 or $\sin(2\pi x/12)$

...

feature $D+1$ = $h_D(x)$... e.g., x^p

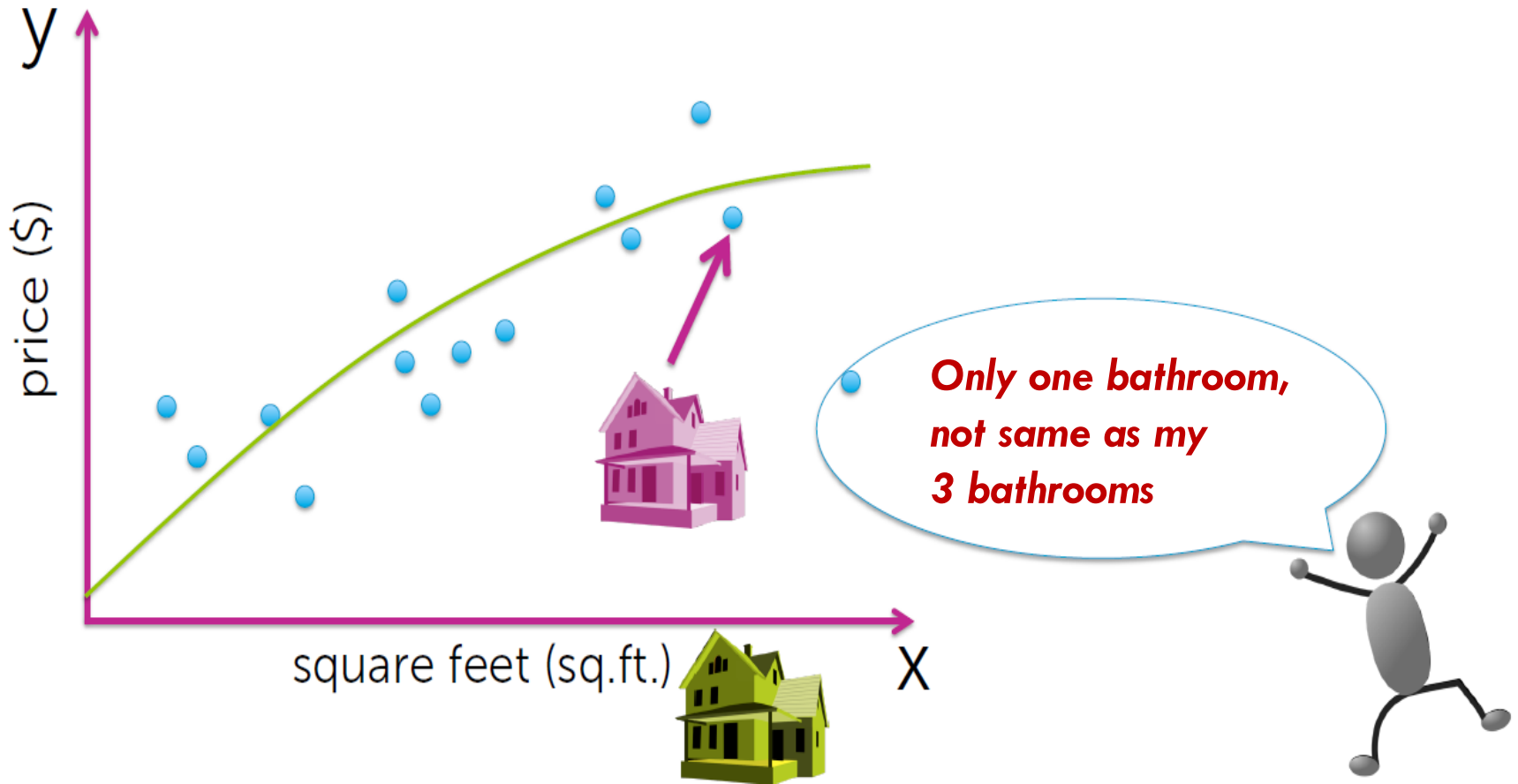
More realistic flow chart

45



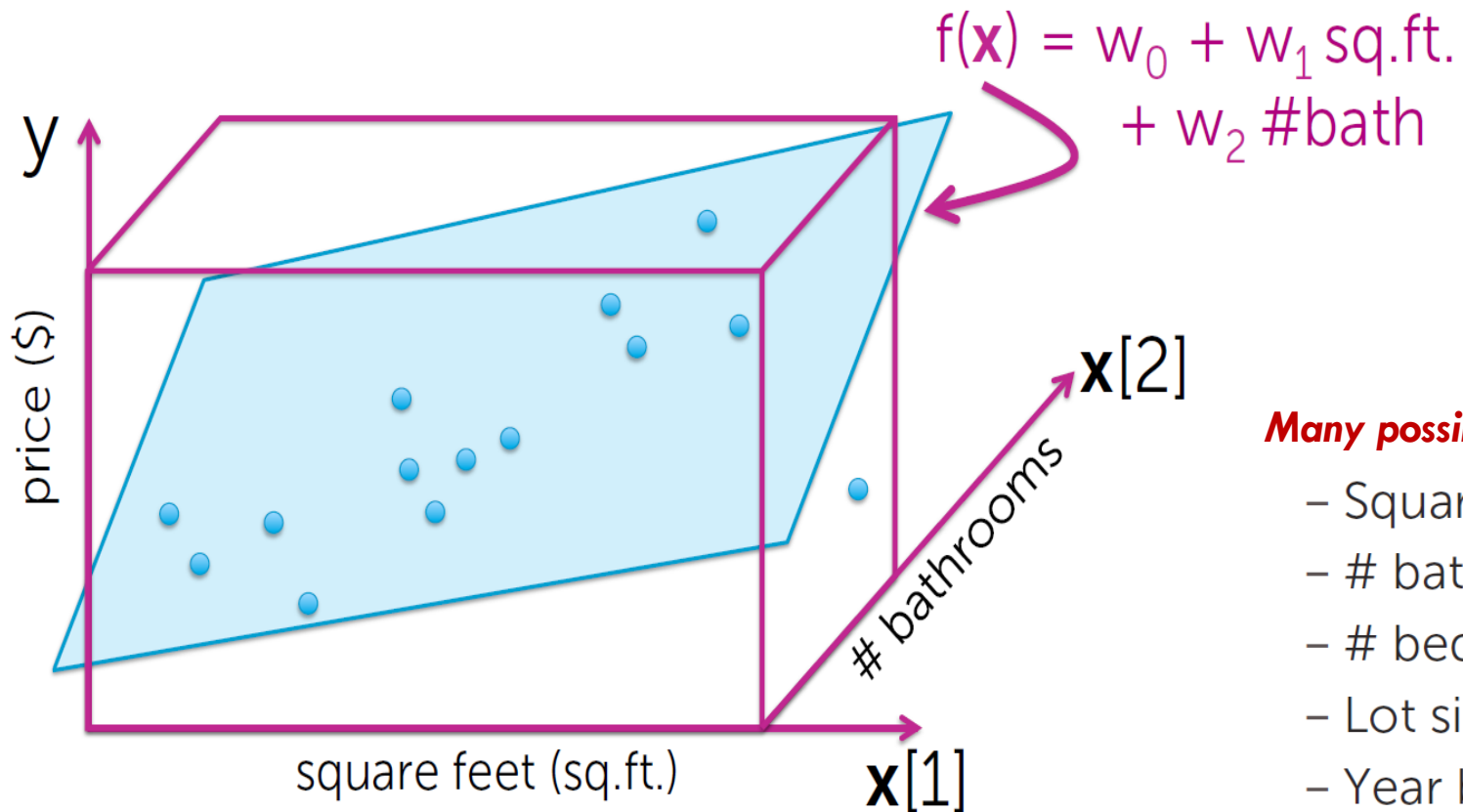
Incorporating multiple inputs

46



Incorporating multiple inputs

47



Many possible inputs

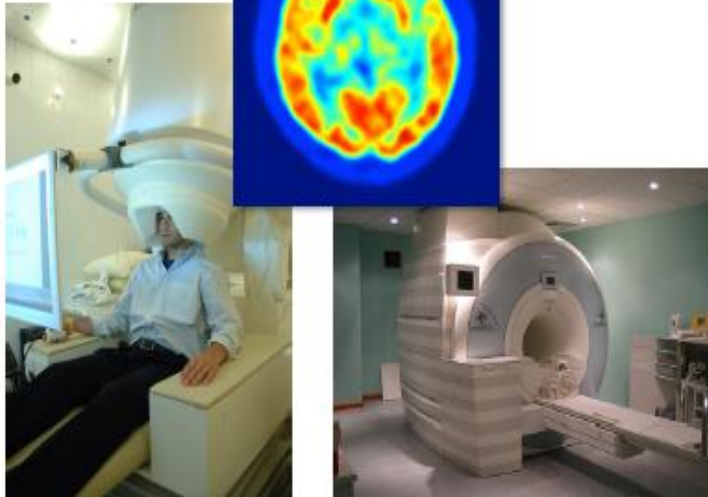
- Square feet
- # bathrooms
- # bedrooms
- Lot size
- Year built
- ...

Reading your brain

48



Whole collection of inputs



Features are
brain region
intensities

General notation

49

Output: y  scalar

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

 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

50

Model:

Noise term



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

feature 1 = 1

feature 2 = $\mathbf{x}[1]$... e.g., sq. ft.

feature 3 = $\mathbf{x}[2]$... e.g., #bath

...

feature $d+1$ = $\mathbf{x}[d]$... e.g., lot size

More generally: D-dimensional curve

51

Model:

$$y_i = w_0 h_0(\mathbf{x}_i) + w_1 h_1(\mathbf{x}_i) + \dots + w_D h_D(\mathbf{x}_i) + \varepsilon_i$$
$$= \sum_{j=0}^D w_j h_j(\mathbf{x}_i) + \varepsilon_i$$

More on notation

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

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

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

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, $\log(\mathbf{x}[7])$ $\mathbf{x}[2]$ = $\log(\#bed) \times \#bath$

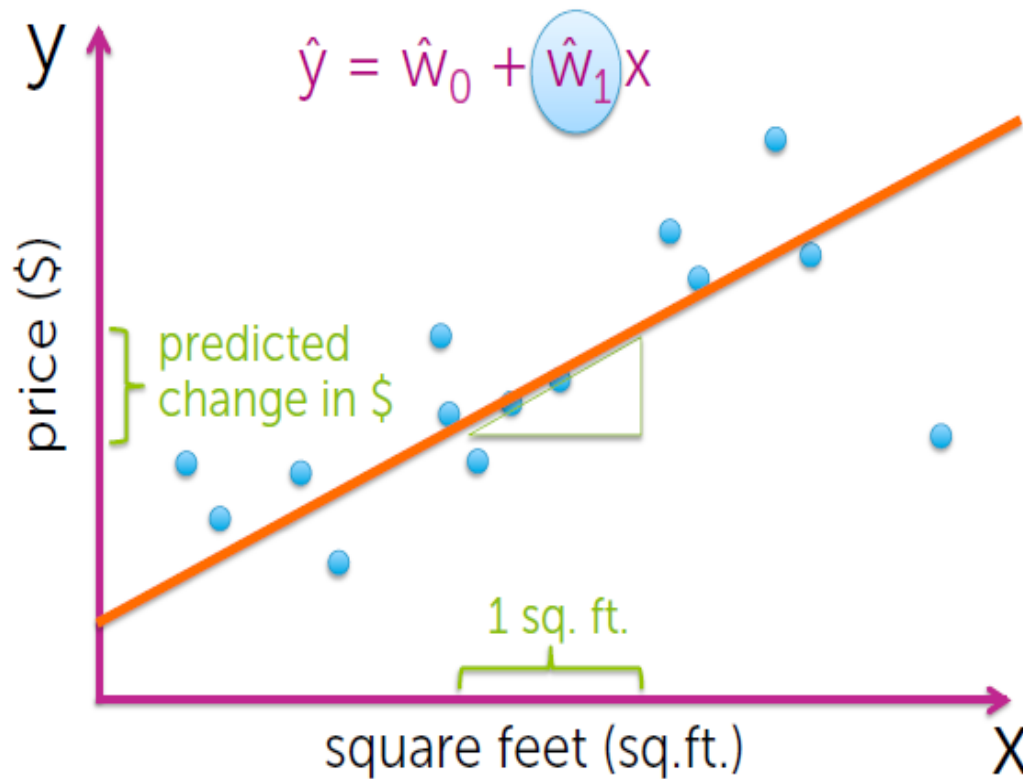
...

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

Interpreting coefficients

52

Simple linear regression



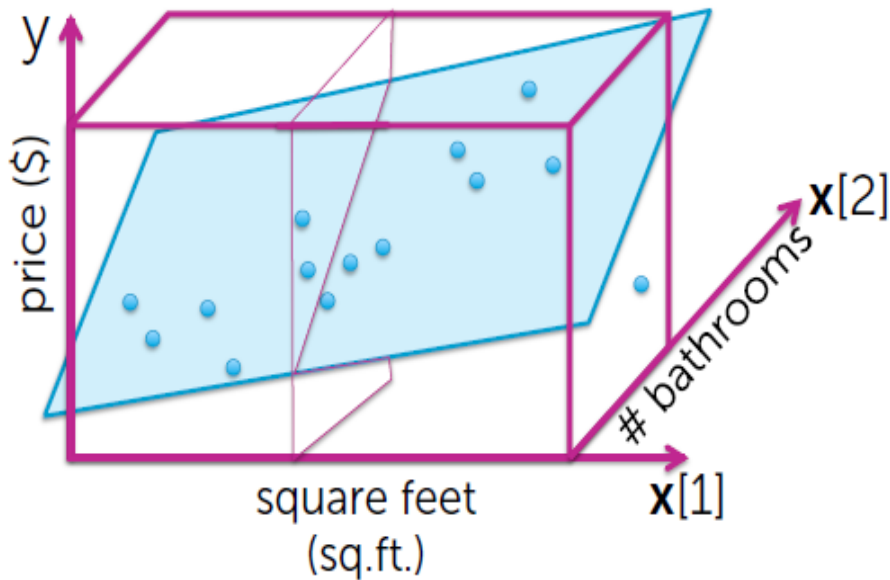
Interpreting coefficients

53

Two linear features

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

fix

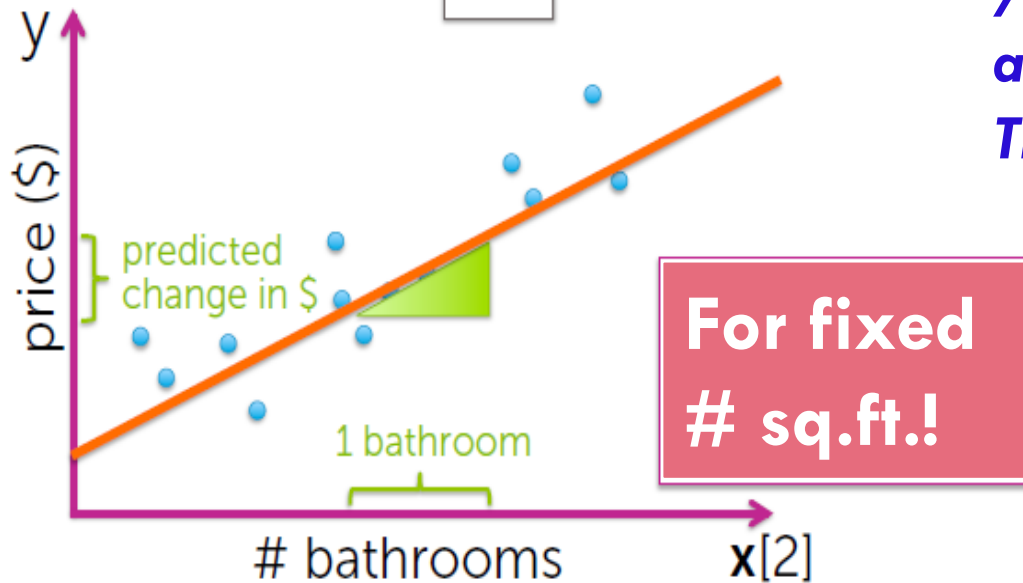


Interpreting coefficients

54

Two linear features

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



But...

**increasing #bathrooms
for fixed #sq.ft will make
your bedrooms smaller
and smaller.**

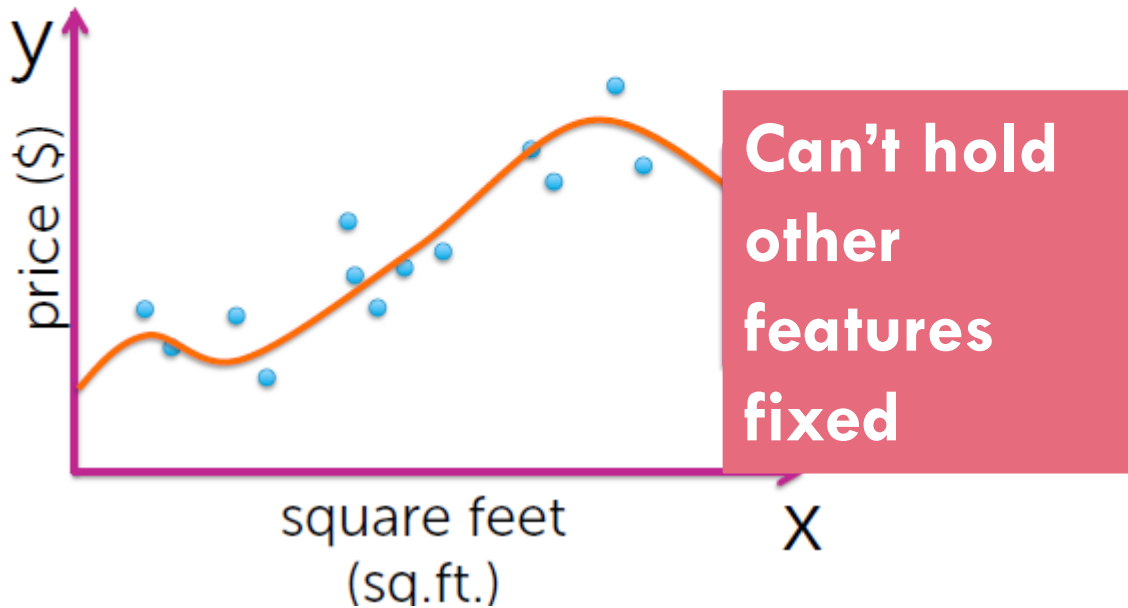
Think about interpretation.

Interpreting coefficients

55

Polynomial regression

$$\hat{y} = \hat{w}_0 + \hat{w}_1 X + \dots + \hat{w}_j X^j + \dots + \hat{w}_p X^p$$



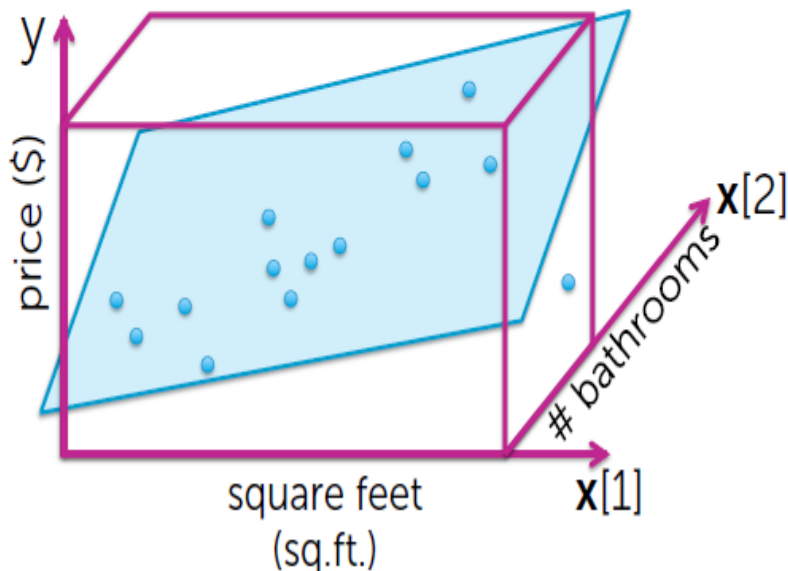
**Then ...
can't interpret
coefficients**

Interpreting coefficients

56

Multiple linear features

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



But...

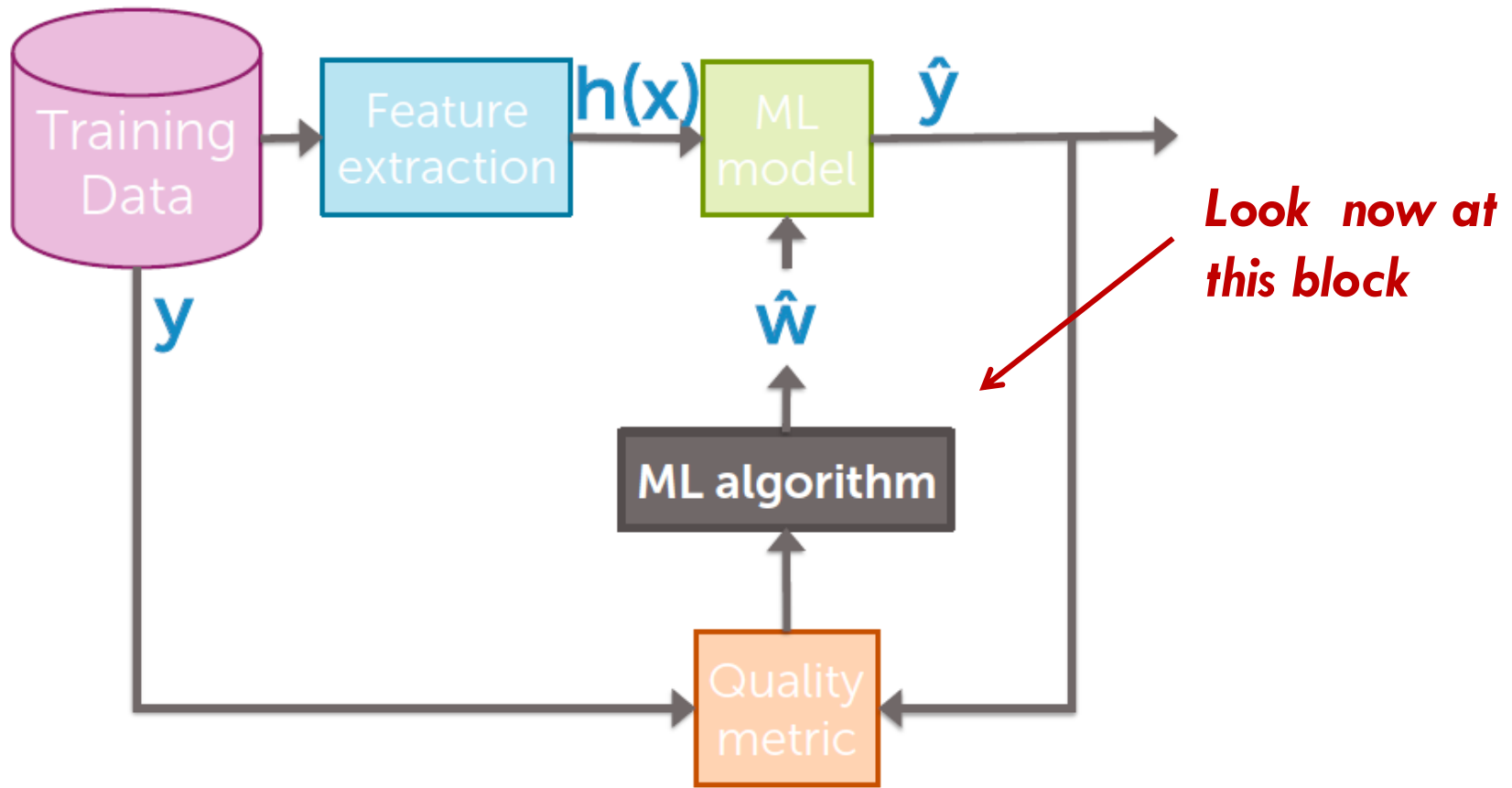
increasing #bedrooms for fixed #sq.ft will make your bedrooms smaller and smaller.

You can end with negative coefficient. Might not be so if you removed #sq.ft from the model.

Think about interpretation in context of the model you put in.

Fitting in D-dimensions

57

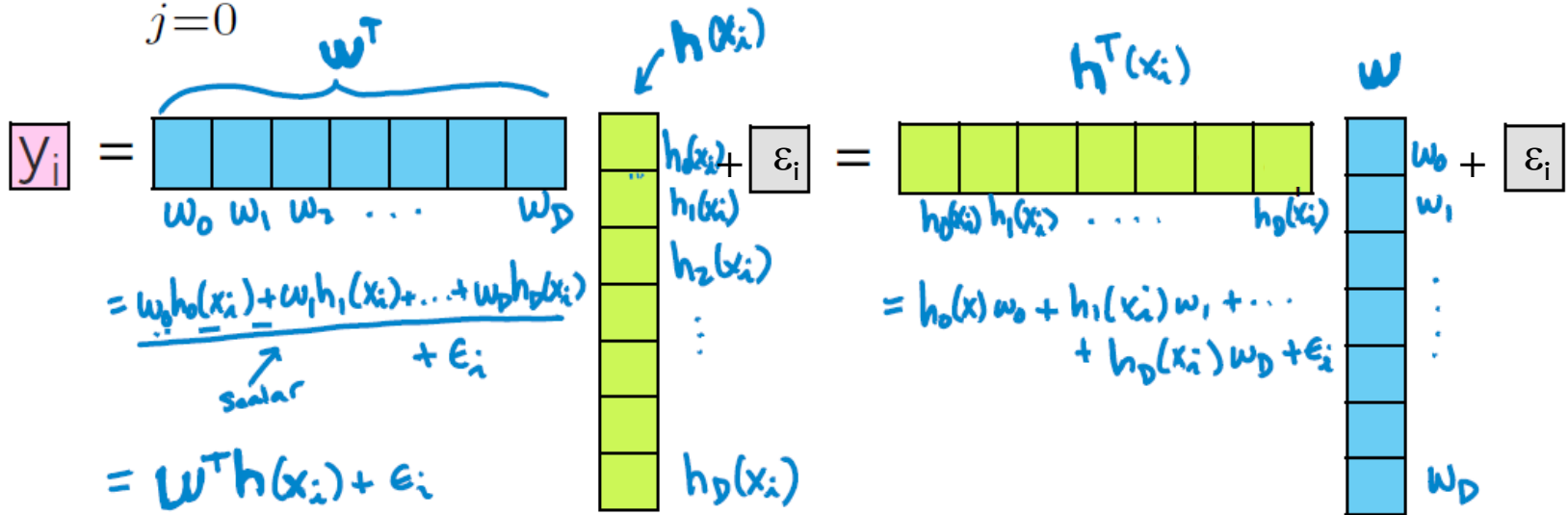


Rewriting in vector notation

58

For observation i

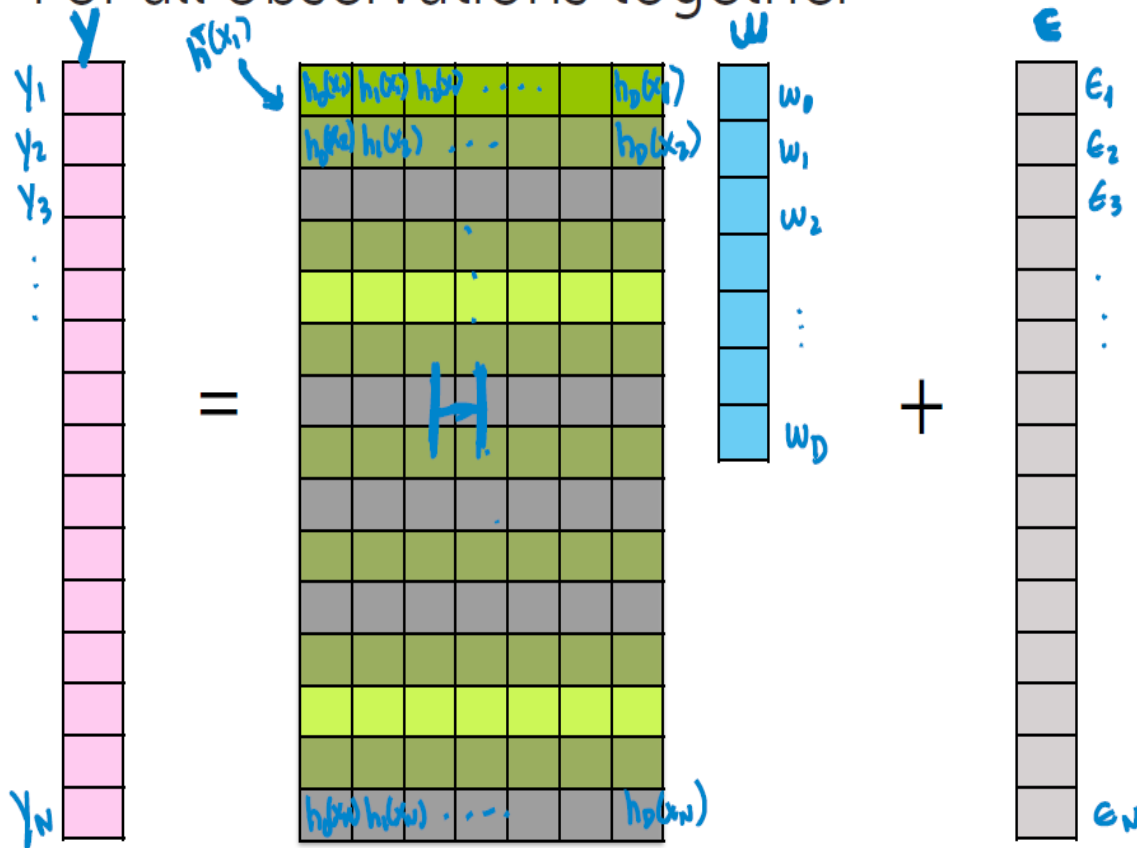
$$y_i = \sum_{j=0}^D w_j h_j(\mathbf{x}_i) + \varepsilon_i$$



Rewriting in matrix notation

59

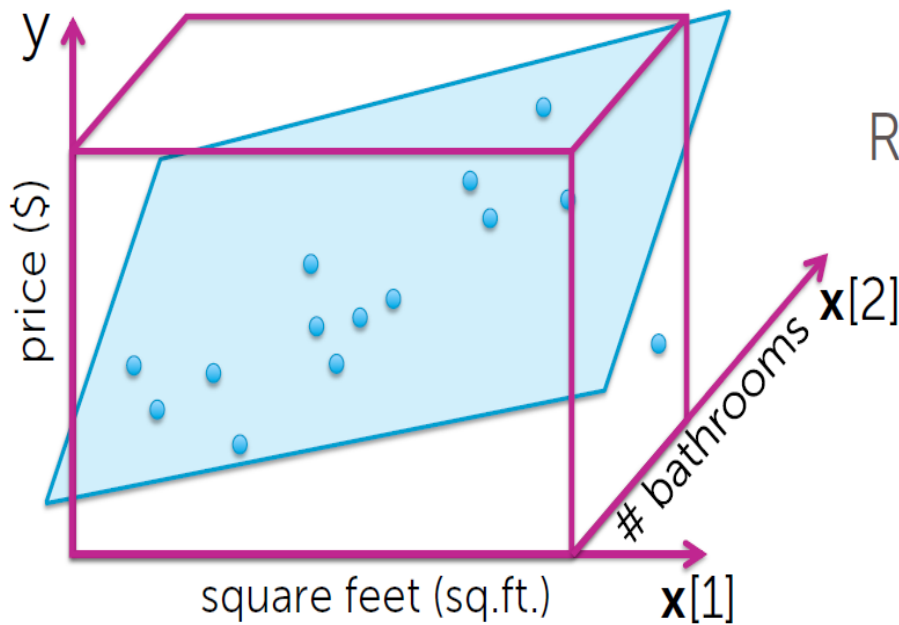
For all observations together



Cost function in D-dimension

61

RSS in vector notation



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

$\hat{y}_i =$

$h^T(x_i)$						
$h_0(x_i) \ h_1(x_i) \ \dots \ h_p(x_i)$						

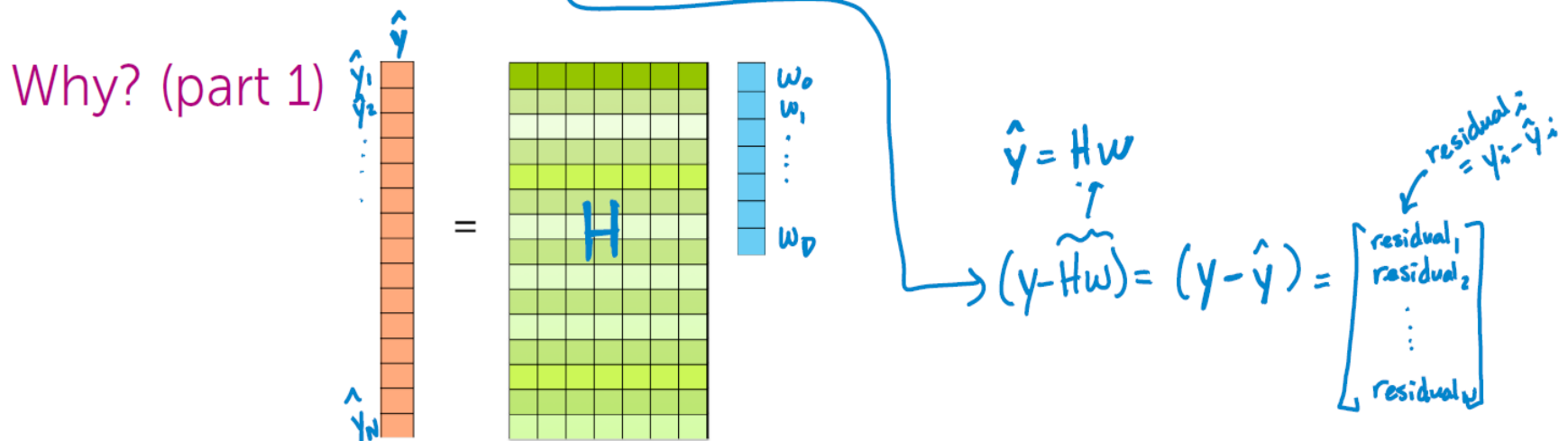
w						
w_0	w_1	w_2	\vdots	\vdots	\vdots	w_D

Cost function in D-dimension

62

RSS in matrix notation

$$\begin{aligned} \text{RSS}(\mathbf{w}) &= \sum_{i=1}^N (y_i - h(\mathbf{x}_i)^T \mathbf{w})^2 \\ &= (\mathbf{y} - \mathbf{H}\mathbf{w})^T (\mathbf{y} - \mathbf{H}\mathbf{w}) \end{aligned}$$



Regression model for D-dimension

63

RSS in matrix notation

$$\begin{aligned} \text{RSS}(\mathbf{w}) &= \sum_{i=1}^N (y_i - h(\mathbf{x}_i)^T \mathbf{w})^2 \\ &= (\mathbf{y} - \mathbf{H}\mathbf{w})^T (\mathbf{y} - \mathbf{H}\mathbf{w}) \end{aligned}$$

Why? (part 2)

residual ₁	residual ₂	residual ₃	...	residual _N	residual ₁
					residual ₂
					residual ₃
					...
					residual _N

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

Regression model for D-dimension

64

Gradient of RSS

$$\begin{aligned}\nabla \text{RSS}(\mathbf{w}) &= \nabla [(\mathbf{y} - \mathbf{H}\mathbf{w})^\top (\mathbf{y} - \mathbf{H}\mathbf{w})] \\ &= -2\mathbf{H}^\top (\mathbf{y} - \mathbf{H}\mathbf{w})\end{aligned}$$

Why? By analogy to 1D case:

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

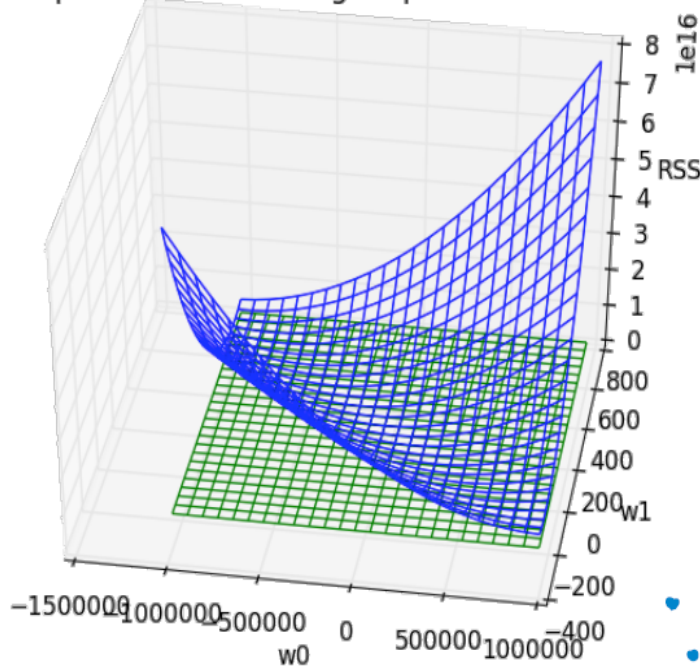
↑
Scalars

Regression model for D-dimension

65

Approach 1: set gradient to zero

3D plot of RSS with tangent plane at minimum



Closed form solution

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

Solve for \mathbf{w} :

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

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

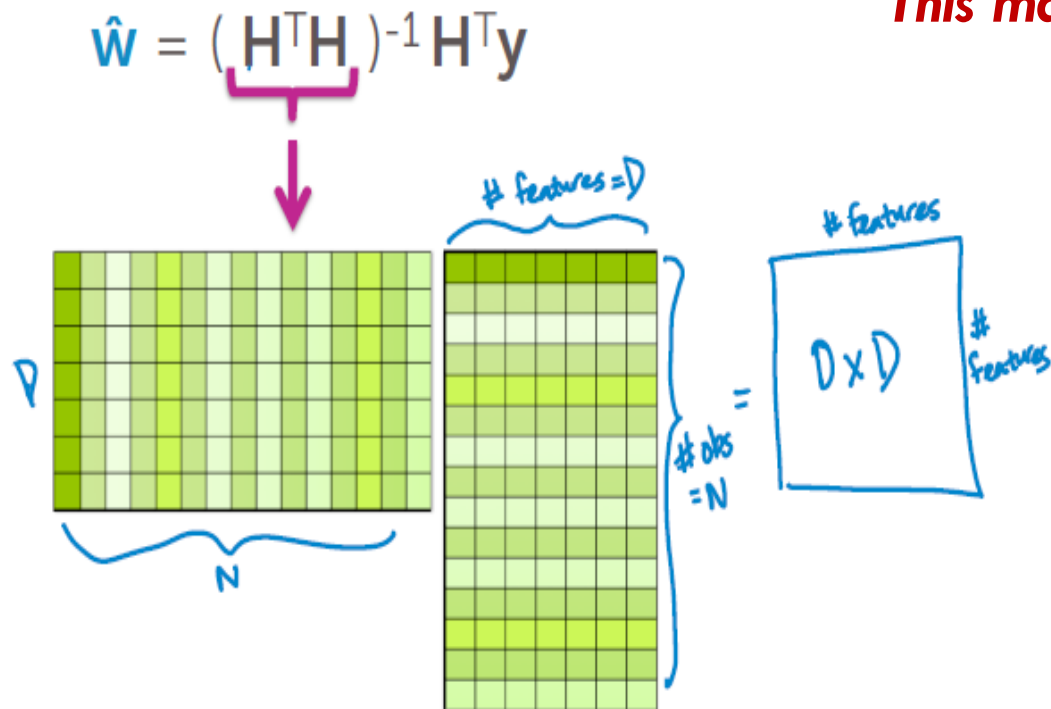
$$\underbrace{(\mathbf{H}^T\mathbf{H})^{-1}}_{\mathbf{I}} \mathbf{H}^T\mathbf{H}\hat{\mathbf{w}} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{y}$$

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

$$\begin{aligned} & \bullet \mathbf{A}^{-1}\mathbf{A} = \mathbf{I} \\ & \bullet \mathbf{I}\mathbf{v} = \mathbf{v} \\ & \bullet \mathbf{I}\mathbf{v} = \mathbf{v} \end{aligned}$$

Closed-form solution

66



This matrix might not be invertible.

Invertible if:
In most cases is $N > D$
really, # of linearly ind. observations

Complexity of inverse:

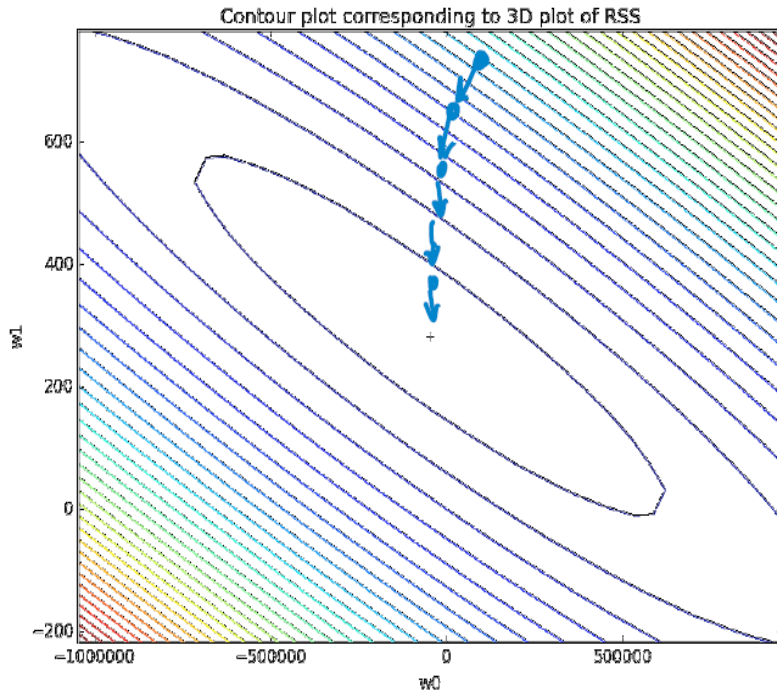
$$O(D^3)$$

This might not be CPU feasible.

Regression model for D-dimension

67

Approach 2: gradient descent



We initialise our solution somewhere and then ...

while not converged

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

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

Gradient descent

68

$$\begin{aligned} \text{RSS}(\mathbf{w}) &= \sum_{i=1}^N (y_i - \mathbf{h}(\mathbf{x}_i)^T \mathbf{w})^2 \\ &= \sum_{i=1}^N (y_i - w_0 h_0(x_i) - w_1 h_1(x_i) - \dots - w_D h_D(x_i))^2 \end{aligned}$$

Partial with respect to w_j

$$\begin{aligned} &\sum_{i=1}^N 2 (y_i - w_0 h_0(x_i) - w_1 h_1(x_i) - \dots - w_D h_D(x_i)) \cdot (-h_j(x_i)) \\ &= -2 \sum_{i=1}^N h_j(x_i) (y_i - \mathbf{h}(\mathbf{x}_i)^T \mathbf{w}) \end{aligned}$$

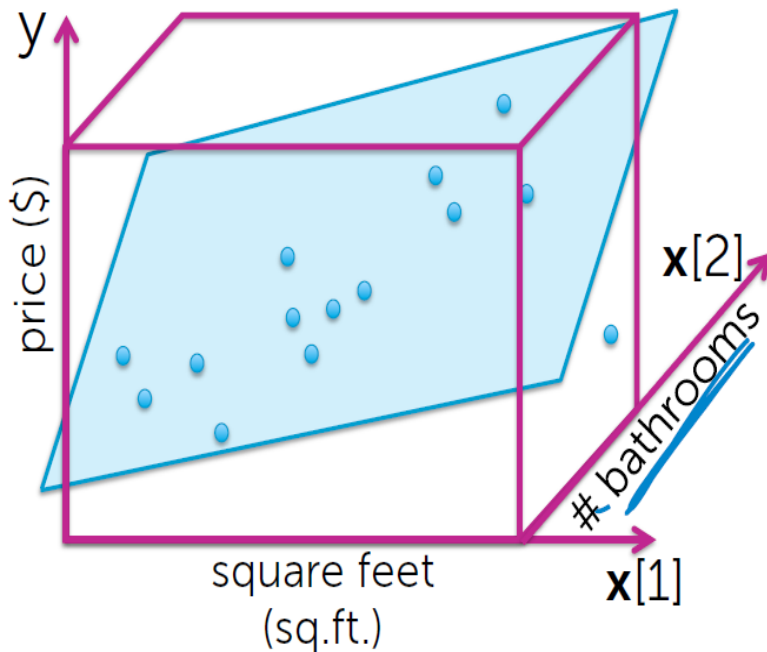
Update to j^{th} feature weight:

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

Regression model for D-dimension

69

Interpreting elementwise



Update to j^{th} feature weight:

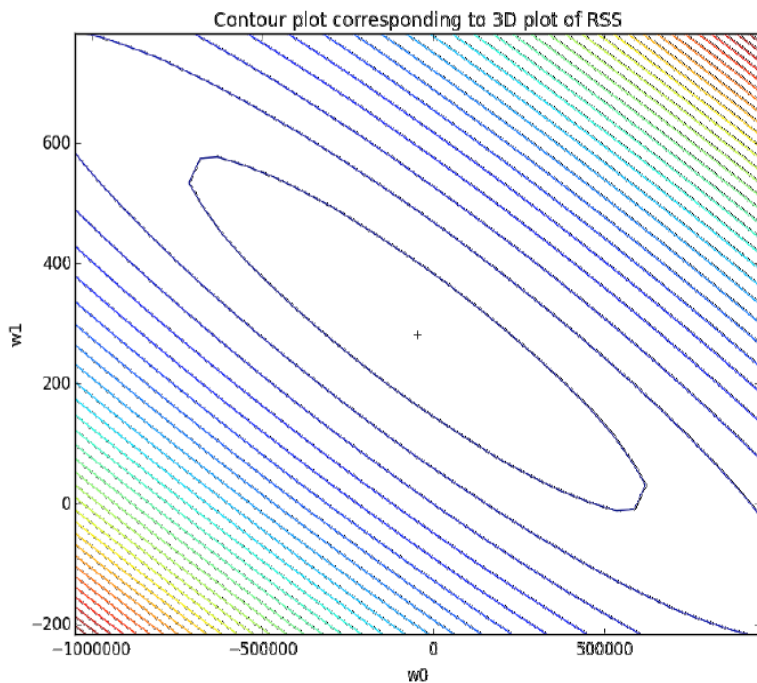
$$w_j^{(t+1)} \leftarrow w_j^{(t)} + 2\eta \sum_{i=1}^N h_j(\mathbf{x}_i) (y_i - \hat{y}_i(\mathbf{w}^{(t)}))$$

If underestimating impact of # bath ($\hat{w}_j^{(t)}$ is too small)
then $(y_i - \hat{y}_i(\mathbf{w}^{(t)}))$ on average
weighted by # bath will be positive
 $\Rightarrow w_j^{(t+1)} > w_j^{(t)}$ (increase)

Summary of gradient descent

70

Extremely useful algorithm in several applications



```
init  $\mathbf{w}^{(1)} = \mathbf{0}$  (or randomly, or smartly),  $t = 1$   
while  $\|\nabla \text{RSS}(\mathbf{w}^{(t)})\| > \epsilon$   
    for  $j = 0, \dots, D$   
         $\text{partial}[j] = -2 \sum_{i=1}^N h_j(\mathbf{x}_i)(y_i - \hat{y}_i(\mathbf{w}^{(t)}))$   
     $\mathbf{w}_j^{(t+1)} \leftarrow \mathbf{w}_j^{(t)} - \eta \text{partial}[j]$   
     $t \leftarrow t + 1$ 
```

Handwritten notes:
- ϵ is tolerance
- The norm in the while loop is $\sqrt{\text{partial}[0]^2 + \dots + \text{partial}[D]^2}$

What you can do now

71

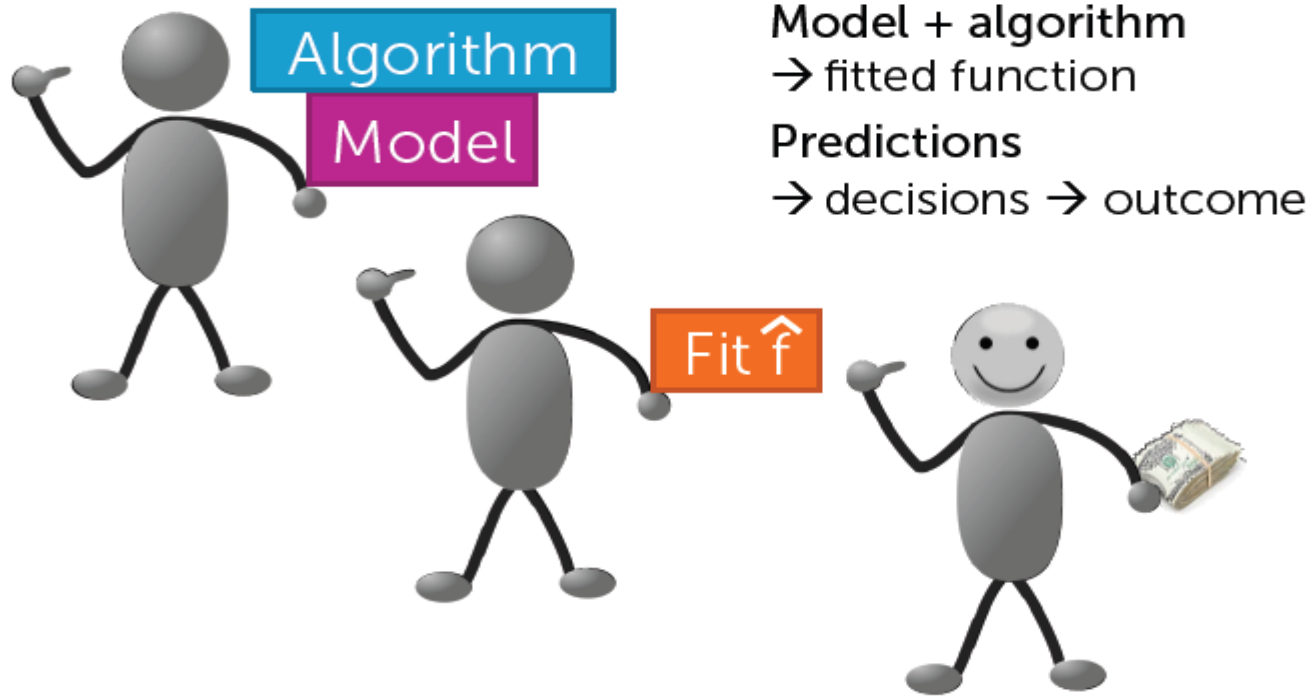
- Describe polynomial regression
- Detrend a time series using trend and seasonal components
- Write a regression model using multiple inputs or features thereof
- Cast both polynomial regression and regression with multiple inputs as regression with multiple features
- Calculate a goodness-of-fit metric (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

ACCESSING PERFORMANCE

Assessing performance

73

Make predictions, get \$, right??



Assessing performance

74

Or, how much am I losing?

Example: Lost \$ due to inaccurate listing price

- Too low → low offers
- Too high → few lookers + no/low offers

How much am I **losing** compared to perfection?

Perfect predictions: Loss = 0

My predictions: Loss = ???

Measuring loss

“Remember that all models are wrong; the practical question is how wrong do they have to be to not be useful.” George Box, 1987.

75

Loss function:

Cost of using \hat{w} at x
when y is true

$$L(y, f_{\hat{w}}(\mathbf{x}))$$

actual value $\hat{f}(\mathbf{x}) =$ predicted value \hat{y}

Examples:

(assuming loss for underpredicting = overpredicting)

Absolute error: $L(y, f_{\hat{w}}(\mathbf{x})) = |y - f_{\hat{w}}(\mathbf{x})|$

Squared error: $L(y, f_{\hat{w}}(\mathbf{x})) = (y - f_{\hat{w}}(\mathbf{x}))^2$

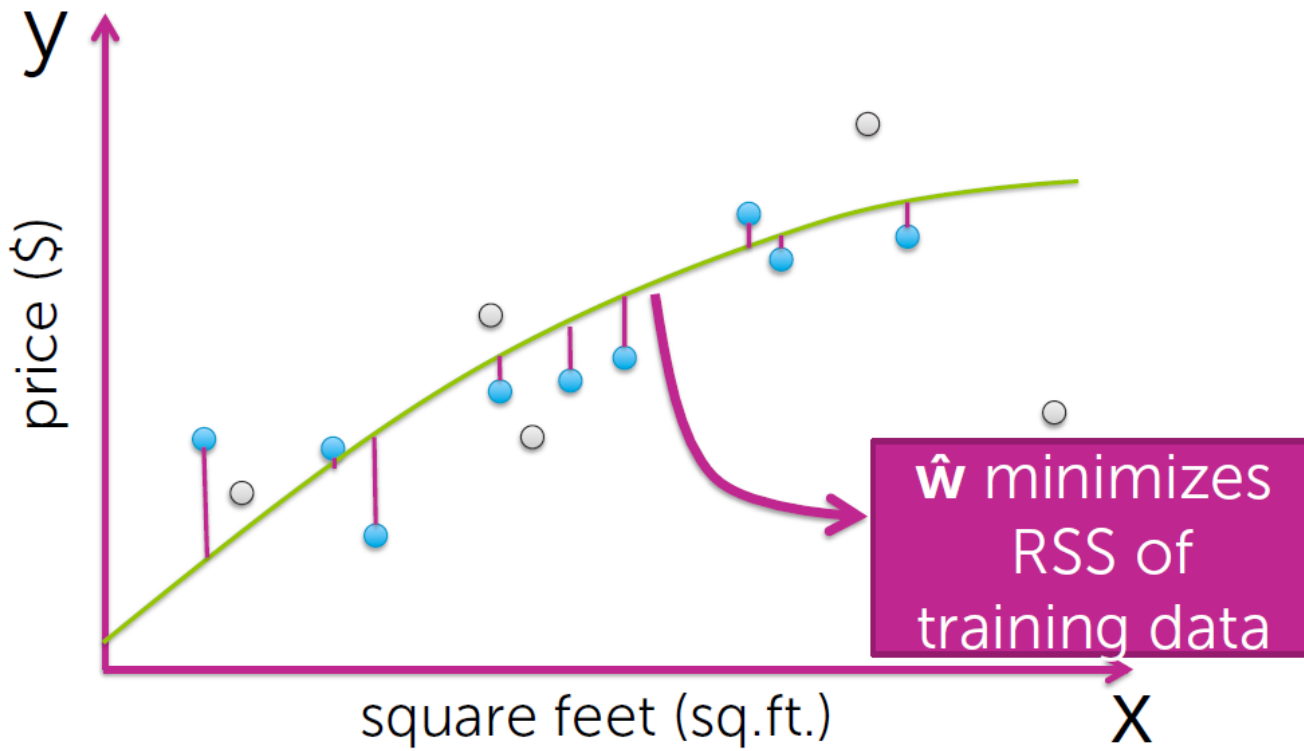
Symmetric loss functions



Accessing the loss

76

Use training data



Compute training error

77

1. Define a loss function $L(y, f_{\hat{\mathbf{w}}}(\mathbf{x}))$
 - E.g., squared error, absolute error, ...

2. Training error

= avg. loss on houses in training set

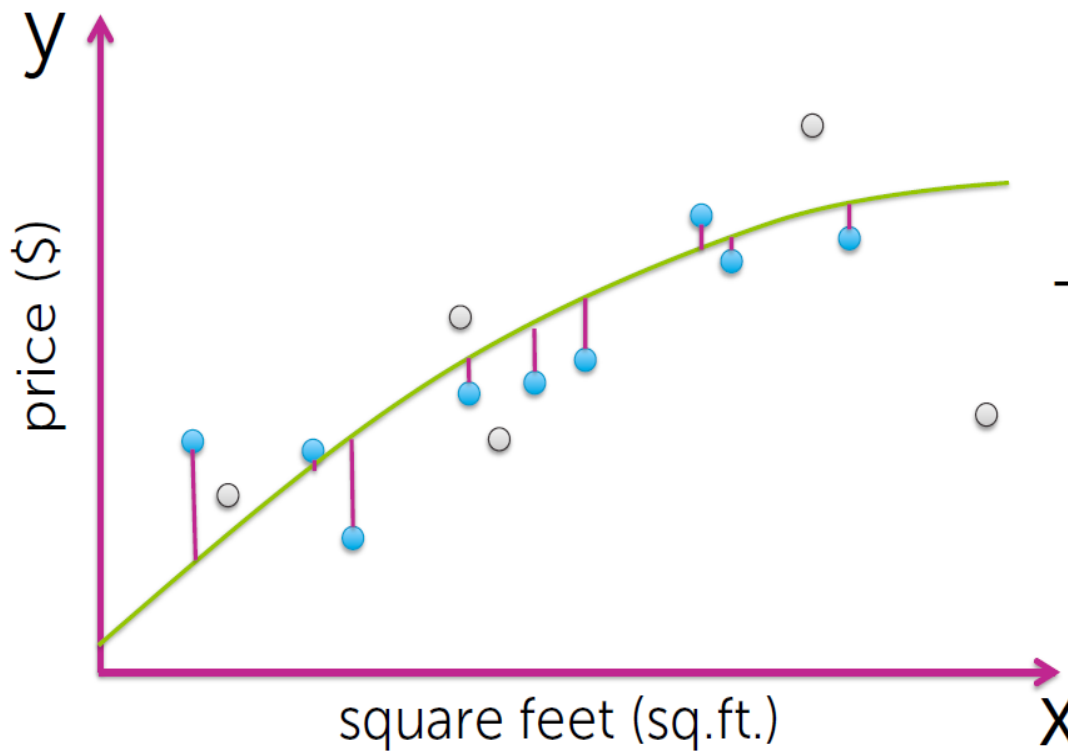
$$= \frac{1}{N} \sum_{i=1}^N L(y_i, f_{\hat{\mathbf{w}}}(\mathbf{x}_i))$$

 fit using training data

Training error

78

Use **squared error loss** $(y - f_{\hat{w}}(x))^2$



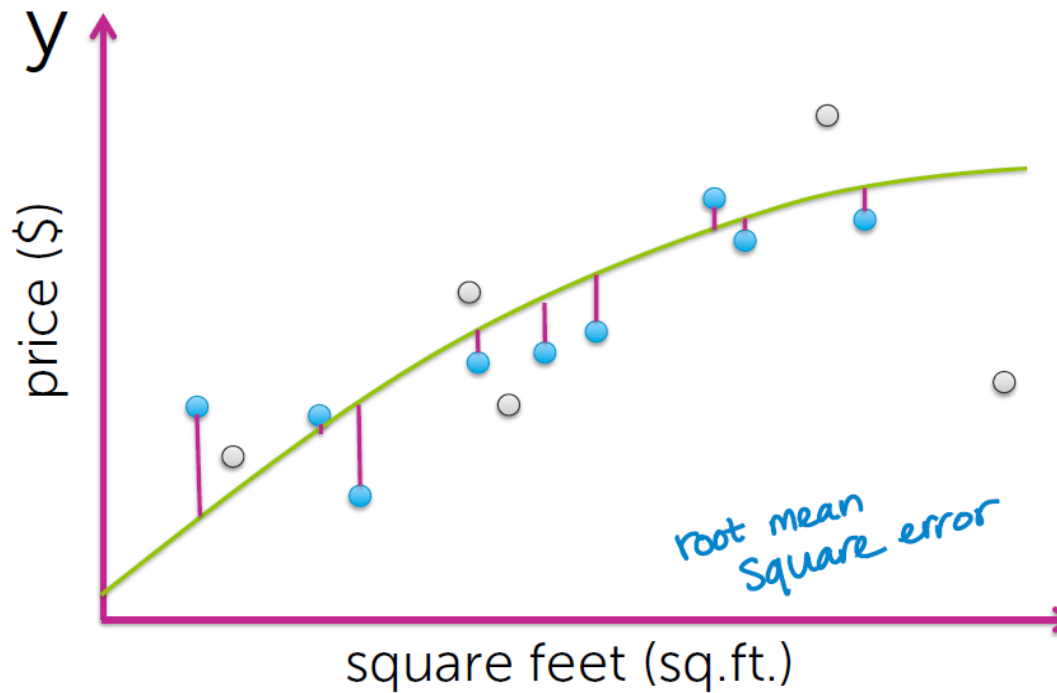
Convention is to take average here

Training error (\hat{w}) = $1/N * [(\$_{\text{train } 1} - f_{\hat{w}}(\text{sq.ft.}_{\text{train } 1}))^2 + (\$_{\text{train } 2} - f_{\hat{w}}(\text{sq.ft.}_{\text{train } 2}))^2 + (\$_{\text{train } 3} - f_{\hat{w}}(\text{sq.ft.}_{\text{train } 3}))^2 + \dots \text{include all training houses}]$

Training error

79

More intuitive is to take RMSE, same units as y

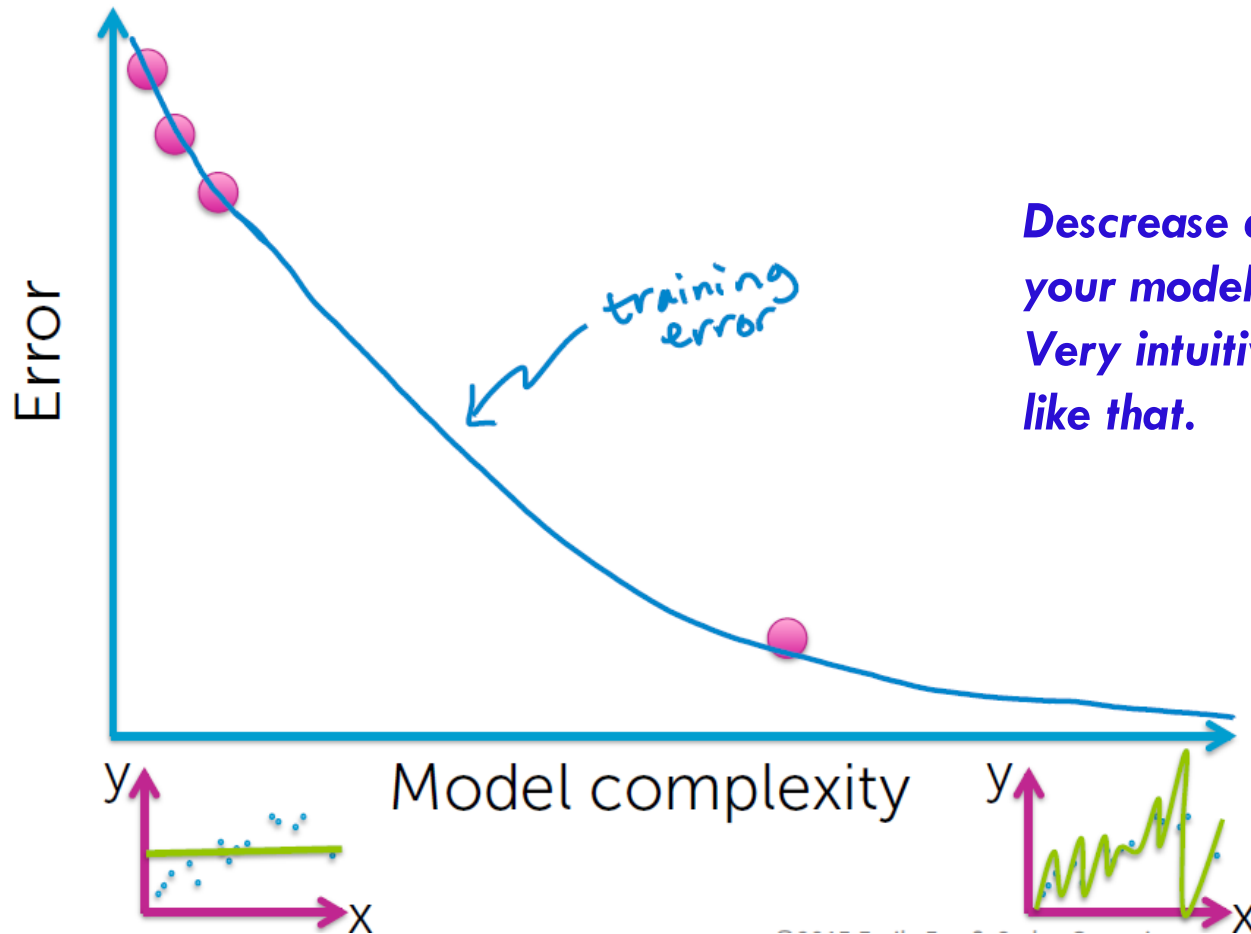


$$\text{Training error } (\hat{\mathbf{w}}) = \frac{1}{N} \sum_{i=1}^N (y_i - f_{\hat{\mathbf{w}}}(\mathbf{x}_i))^2$$

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - f_{\hat{\mathbf{w}}}(\mathbf{x}_i))^2}$$

Training error vs. model complexity

80

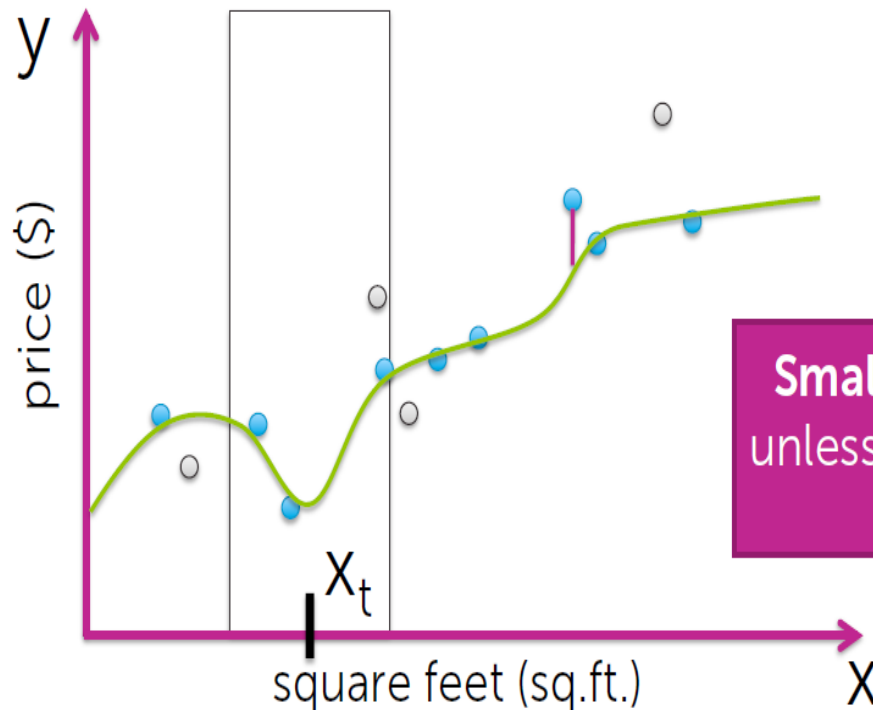


Decrease as you increase your model complexity. Very intuitive why it is like that.

Is training error a good measure?

81

Issue: Training error is overly optimistic
because \hat{w} was fit to training data



Is there something particularly wrong about having x_t square feet ???

Small training error \neq good predictions
unless training data includes everything you might ever see

Generalisation (true) error

82

Really want estimate of loss
over all possible (🏠, \$) pairs

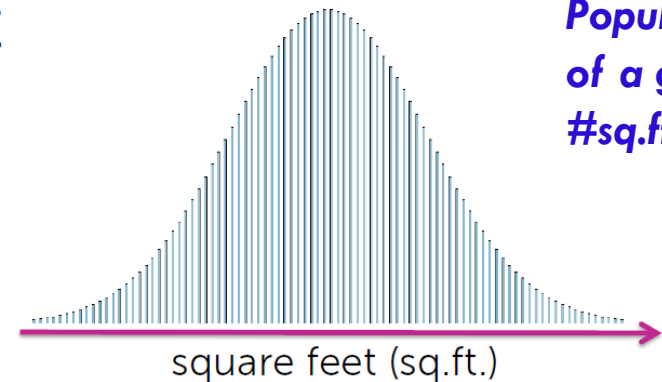


Lots of houses
in neighborhood,
but not in dataset

Distribution over house

83

In our neighborhood, houses of what # sq.ft. (🏠) are we likely to see?



**Popularity
of a given
#sq.ft.**

For houses with a given # sq.ft. (🏠),
what house prices \$ are we likely to see?



For fixed
sq.ft.

Generalisation error definition

84

Really want estimate of loss
over all possible (🏠, \$) pairs

Formally:

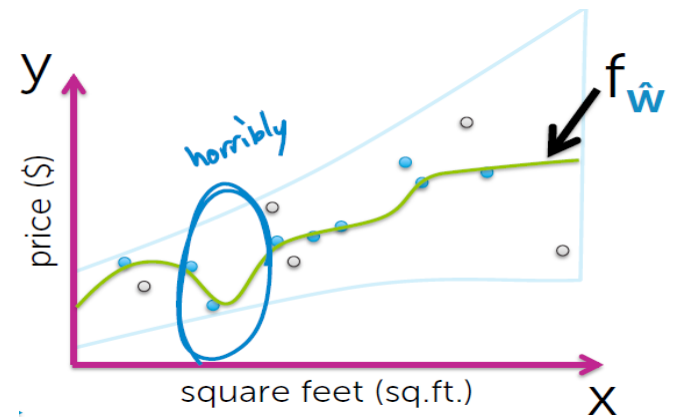
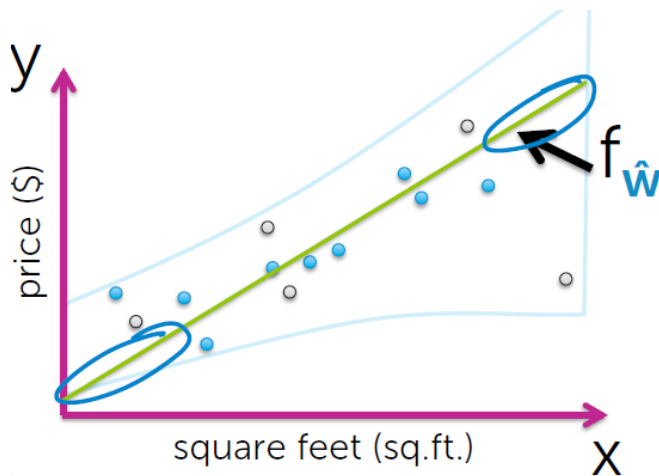
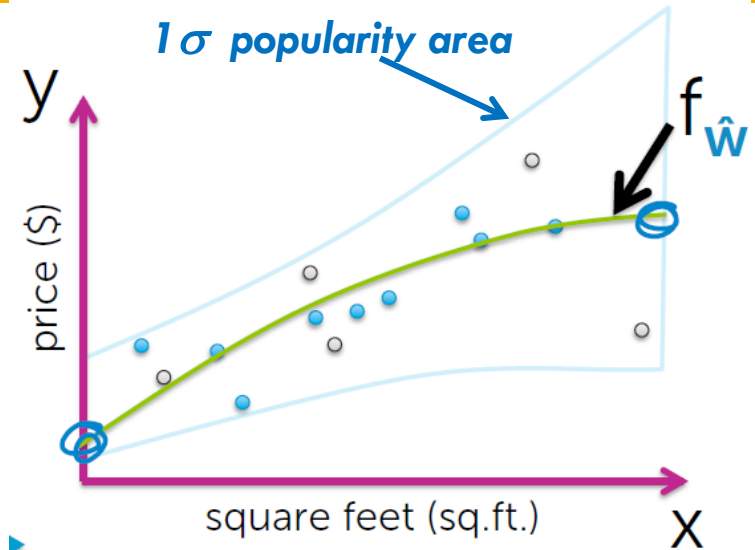
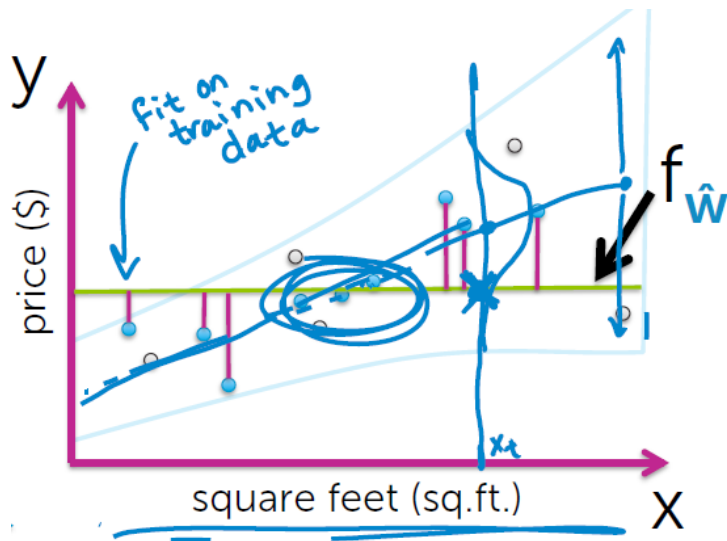
average over all possible
(\mathbf{x}, y) pairs weighted by
how likely each is

$$\text{generalization error} = E_{\mathbf{x}, y} [L(y, f_{\hat{\mathbf{w}}}(\mathbf{x}))]$$

fit using training data

Generalisation error (weighted with popularity) vs model complexity

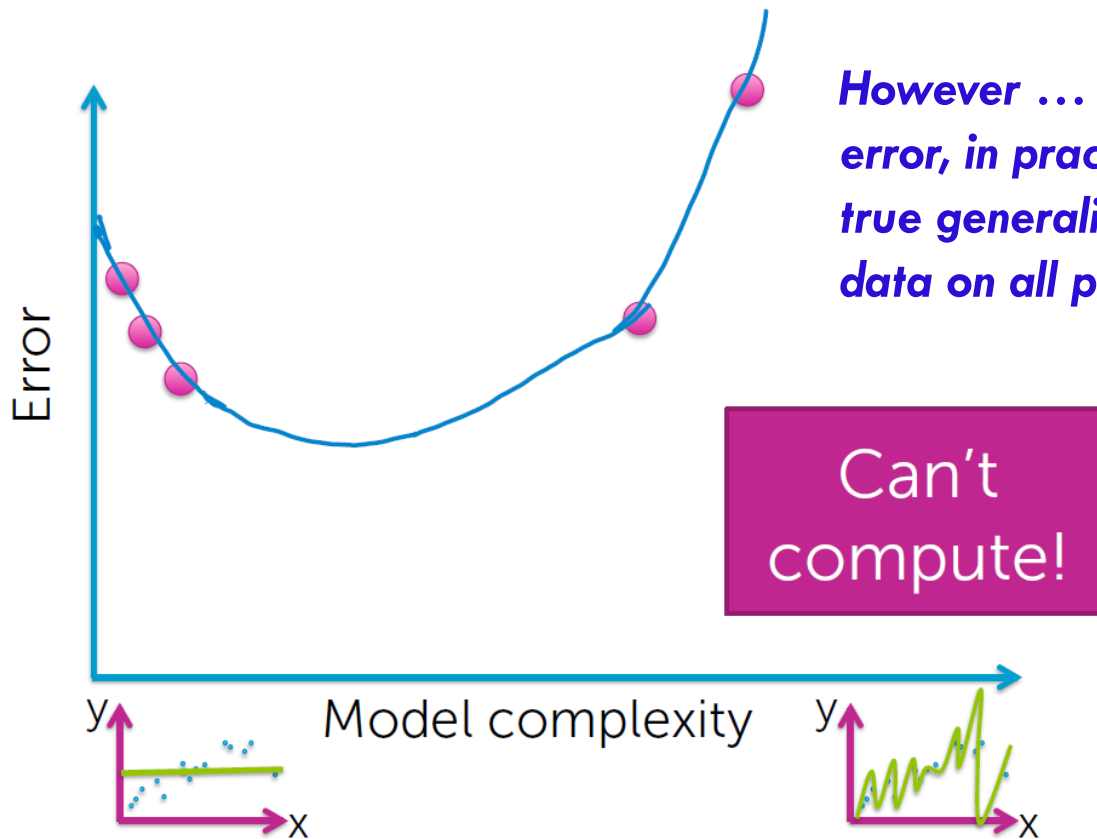
85



15/10, 22/10, 29/10 2019

Generalisation error vs model complexity

86



However ... in contrast to the training error, in practice we cannot really compute true generalisation error. We don't have data on all possible houses in the area.

Can't compute!

Forming a test set

87

Hold out some (🏠, \$) that are *not* used for fitting the model



We want to approximate generalisation error.

Test set: proxy for „everything you might see“

Training set



Test set



Compute test error

88

Test error

= avg. loss on houses in test set

$$= \frac{1}{N_{test}} \sum_{i \text{ in test set}} L(y_i, f_{\hat{w}}(\mathbf{x}_i))$$



test points

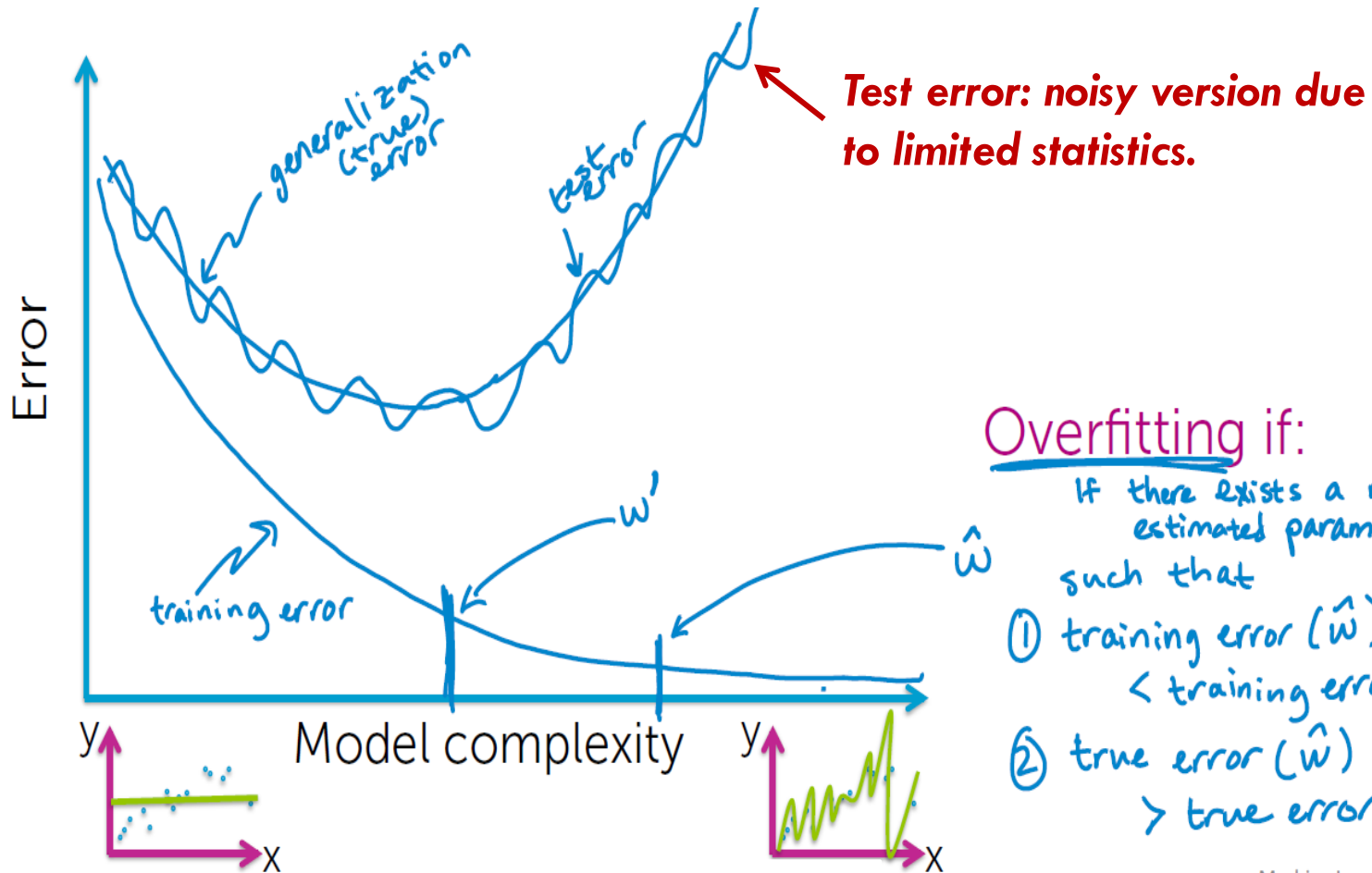


fit using training data

**has never seen
test data!**

Training, true and test error vs. model complexity. Notion of overfitting.

89



Overfitting if:

If there exists a model with estimated params w' such that

- ① training error (\hat{w}) $<$ training error (w')
- ② true error (\hat{w}) $>$ true error (w')

Training/test splits

90



↑
Too few → \hat{w} poorly estimated



↑
Too few → test error bad approximation of generalization error



Typically, just enough test points to form a reasonable estimate of generalization error

If this leaves too few for training, other methods like **cross validation** (will see later...)

Three sources of errors

91

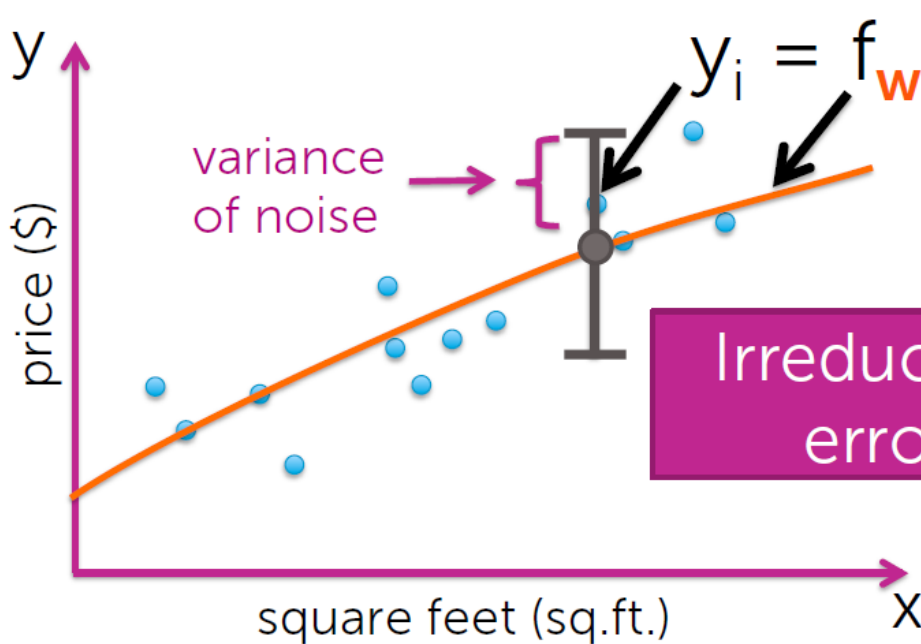
In forming predictions, there are 3 sources of error:

1. Noise
2. Bias
3. Variance

Data are inherently noisy

92

There is some true relationship between sq.ft and value of the house, specific to the given house.



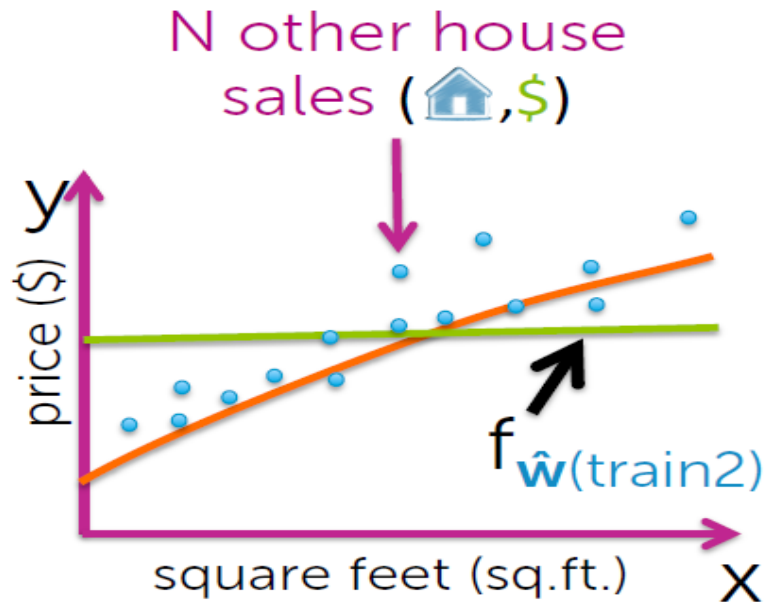
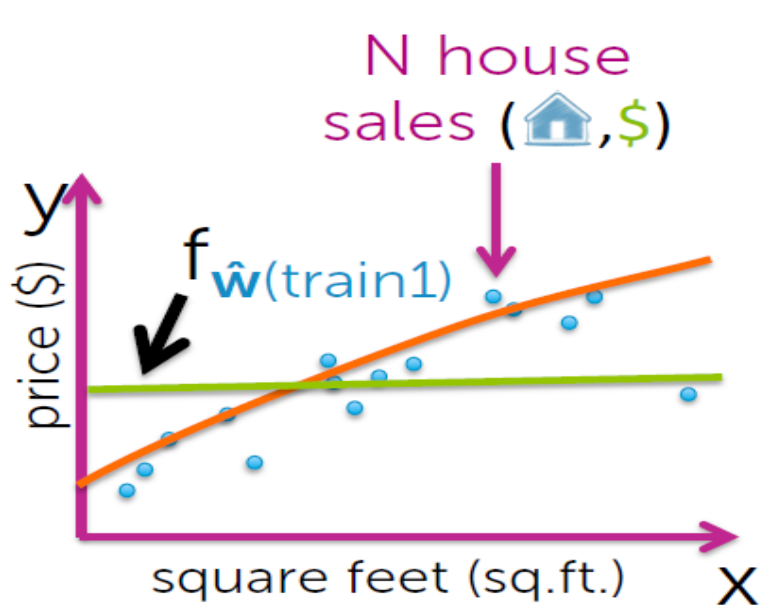
We cannot reduce it by choosing better model or procedure, It is beyond our control.

Bias contribution

93

This contribution we can control.

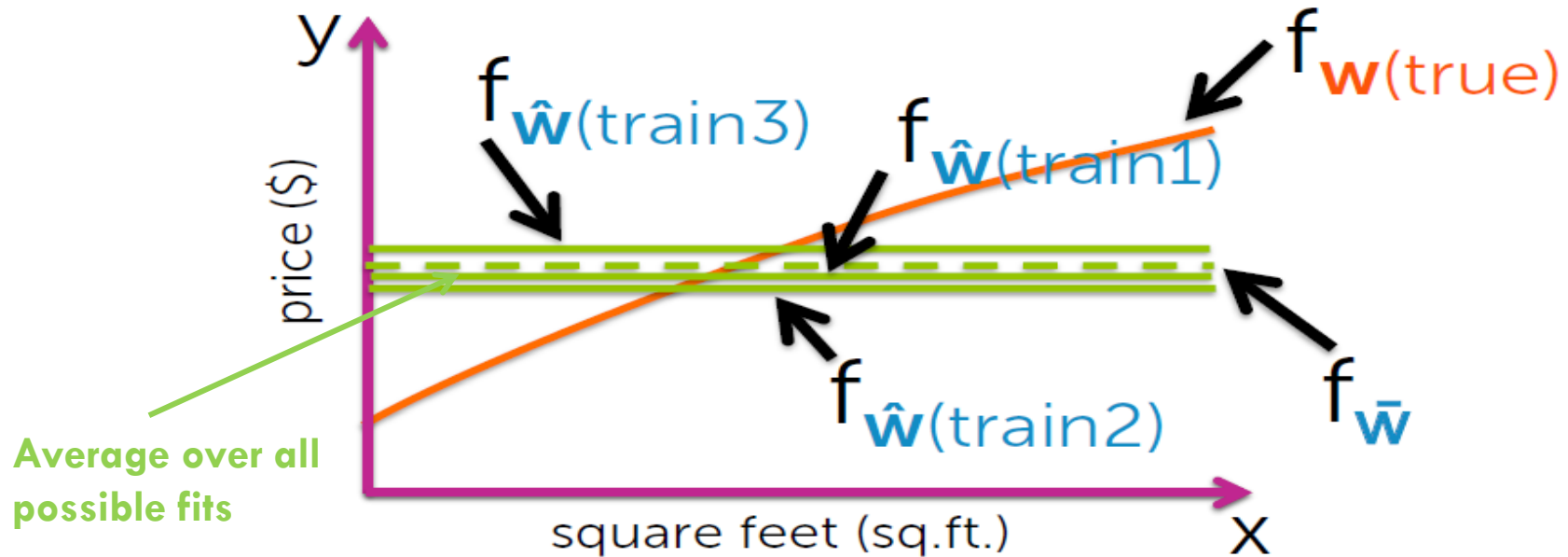
Assume we fit a constant function



Bias contribution

94

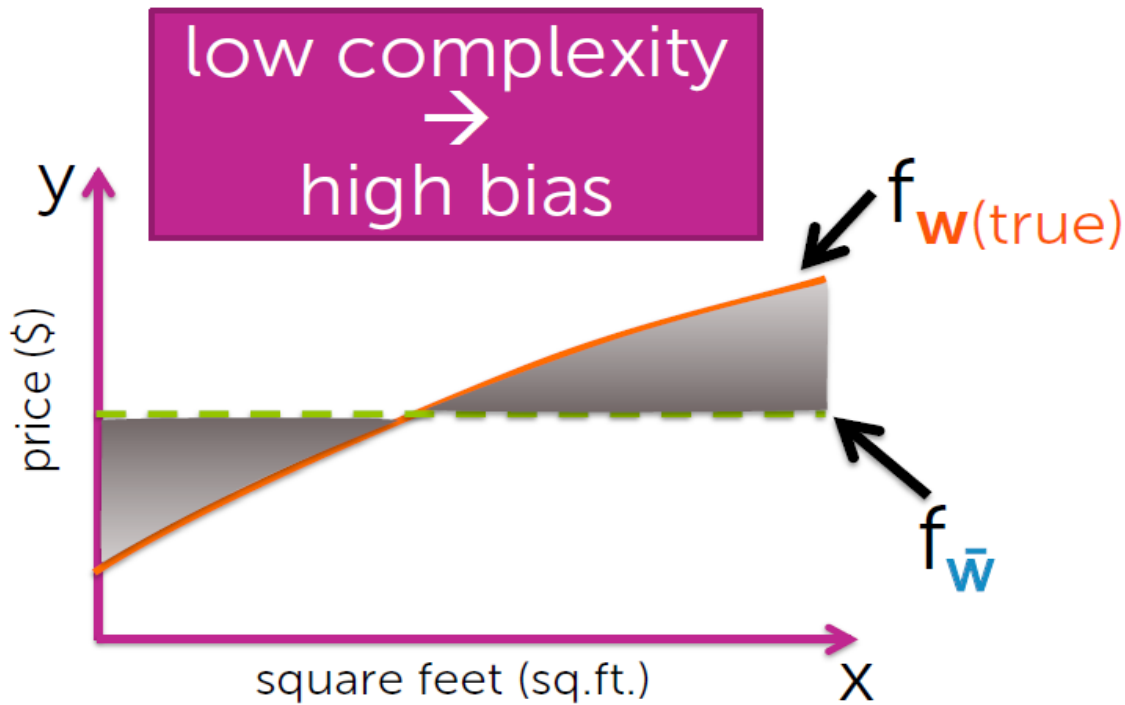
Over all possible size N training sets, what do I expect my fit to be?



Bias contribution

95

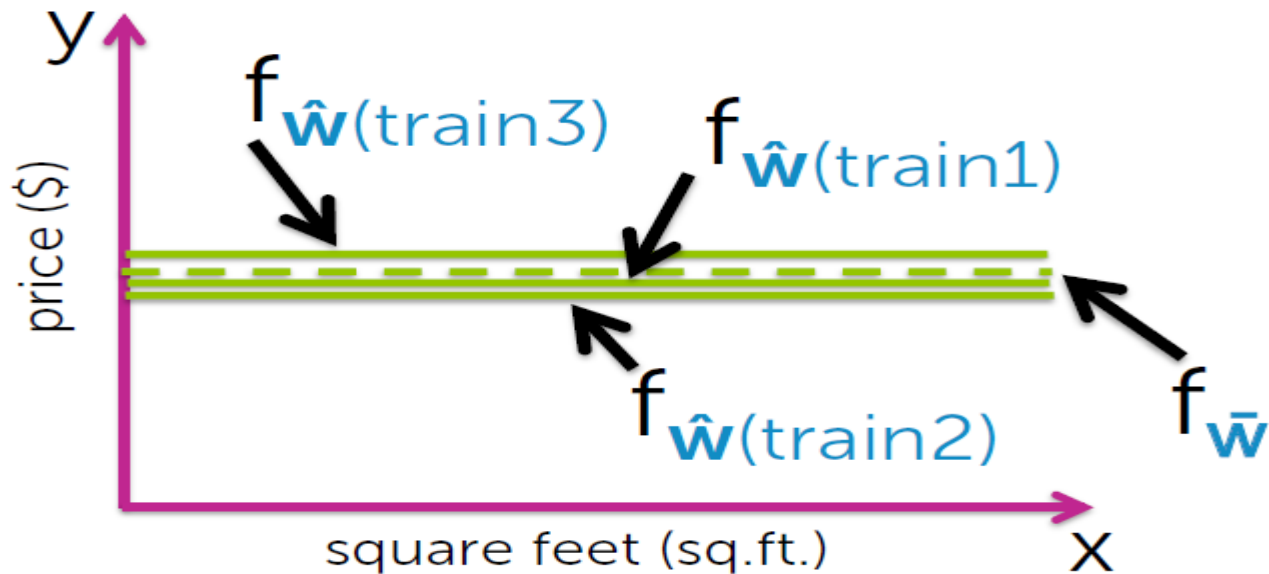
$\text{Bias}(\mathbf{x}) = f_{\mathbf{w}(\text{true})}(\mathbf{x}) - f_{\bar{\mathbf{w}}}(\mathbf{x})$ ← Is our approach flexible enough to capture $f_{\mathbf{w}(\text{true})}$?
If not, error in predictions.



Variance contribution

96

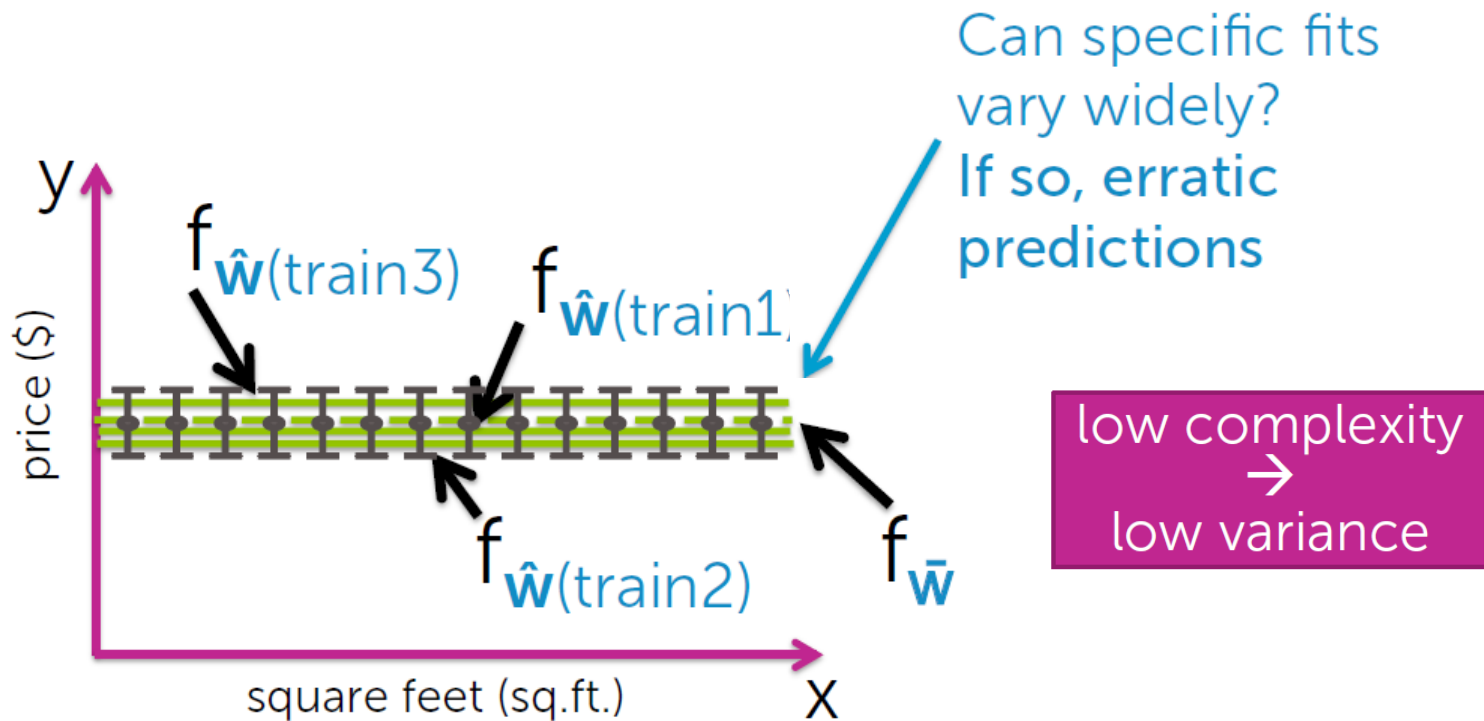
How much do specific fits vary from the expected fit?



Variance contribution

97

How much do specific fits vary from the expected fit?

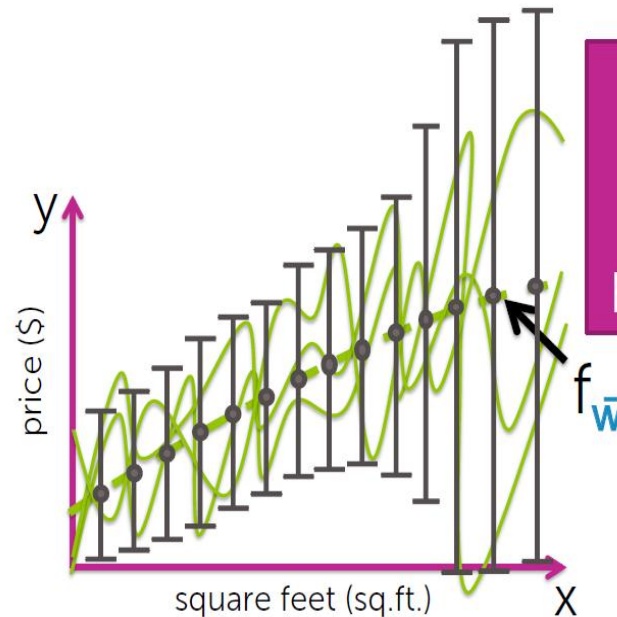
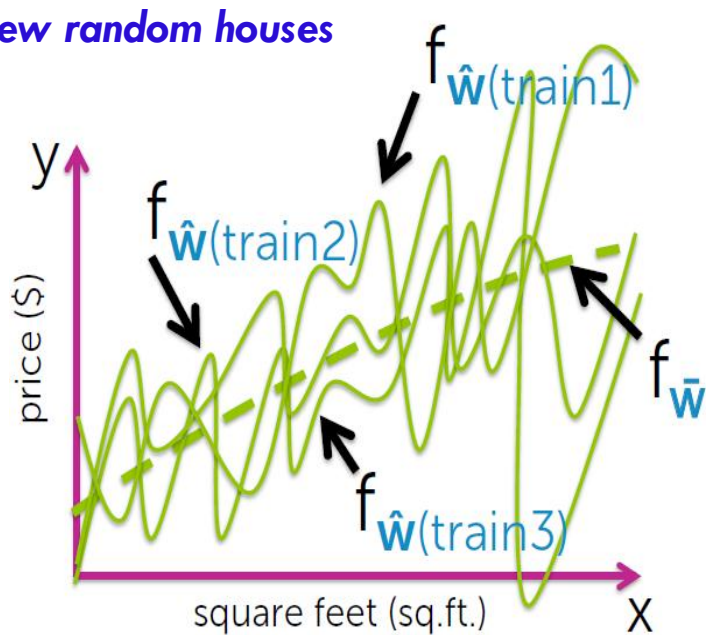


Variance of high complexity models

98

Assume we fit a high-order polynomial

For each train remove
few random houses



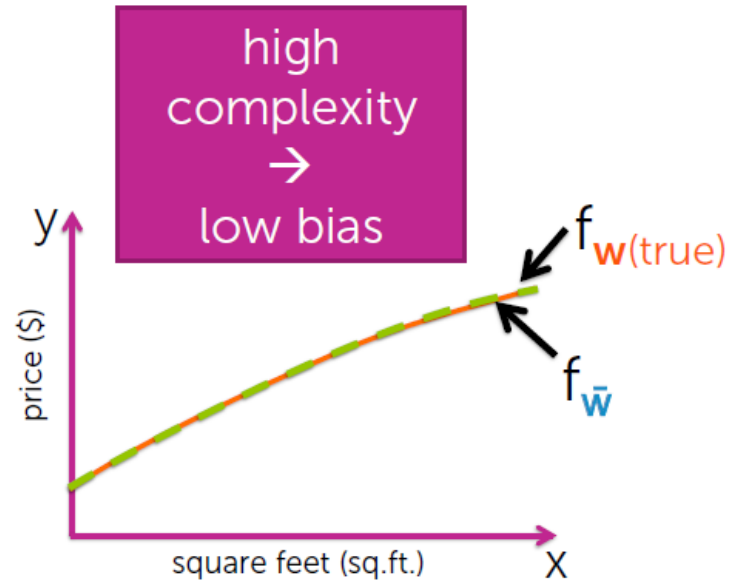
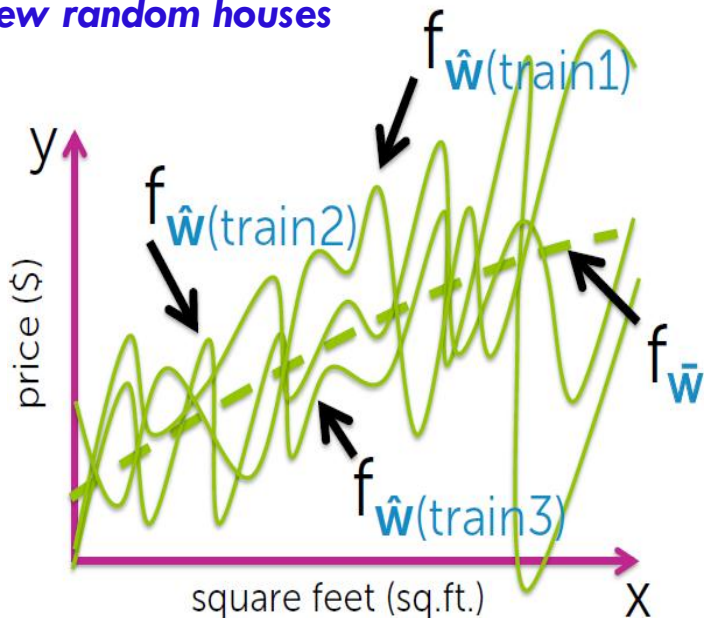
high
complexity
→
high variance

Bias of high complexity models

99

Assume we fit a high-order polynomial

*For each train remove
few random houses*

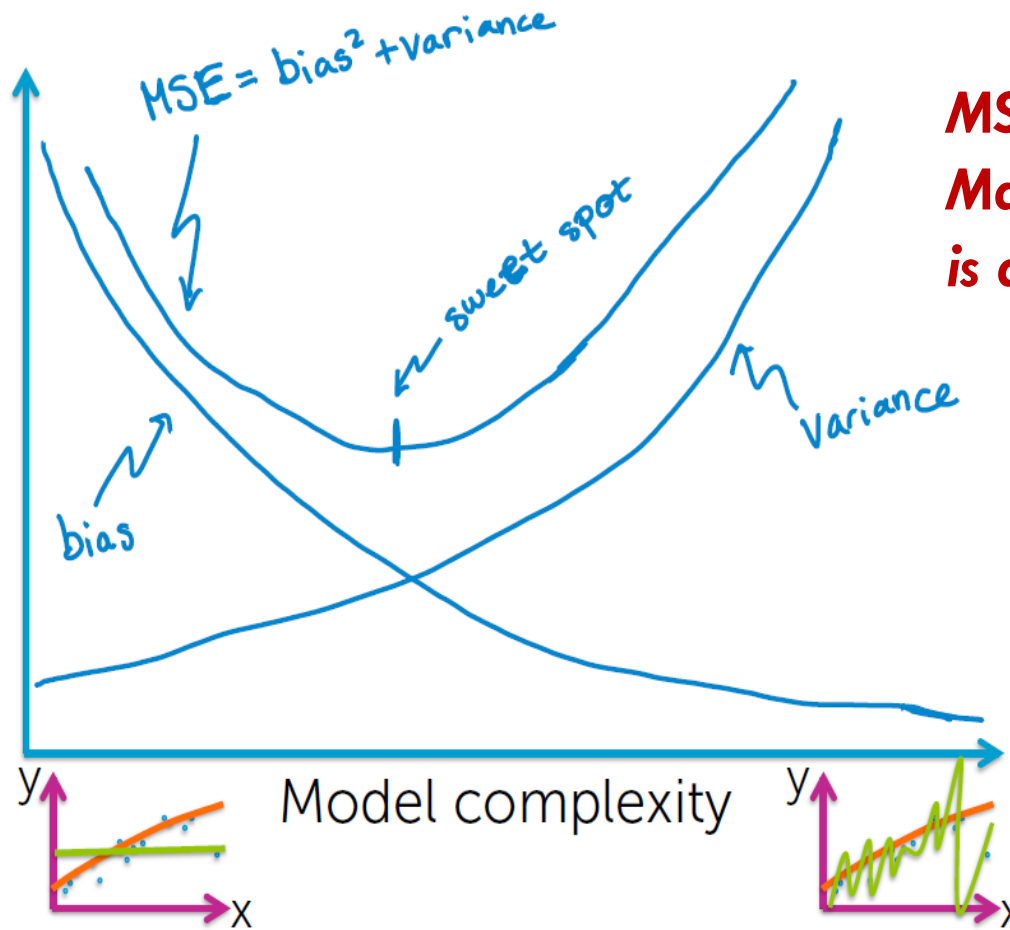


**High complexity models are very flexible,
pick better average trends.**

15/10, 22/10, 29/10 2019

Bias –variance tradeoff

100



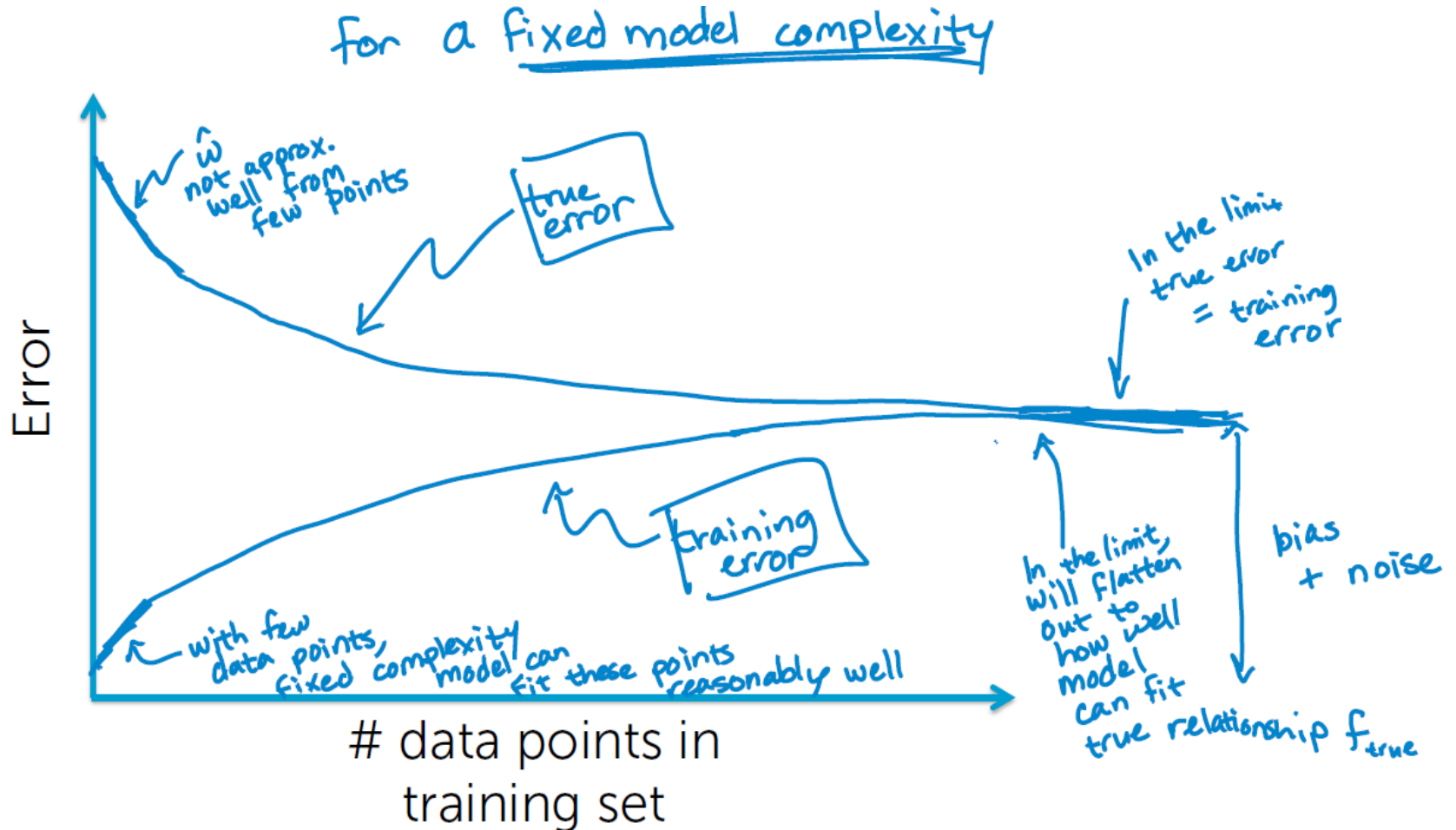
MSE = mean square error
Machine Learning
is all about this tradeoff

But....

Just like with
generalization error,
we cannot compute
bias and variance

Errors vs amount of data

101



The regression/ML workflow

102

1. Model selection

Often, need to choose tuning parameters λ controlling model complexity (e.g. degree of polynomial)

2. Model assessment

Having selected a model, assess the generalization error

Hypothetical implementation

103

Training set

Test set

1. Model selection

For each considered model complexity λ :

- i. Estimate parameters $\hat{\mathbf{w}}_\lambda$ on **training data**
- ii. Assess performance of $\hat{\mathbf{w}}_\lambda$ on **test data**
- iii. Choose λ^* to be λ with **lowest test error**

2. Model assessment

Compute test error of $\hat{\mathbf{w}}_{\lambda^*}$ (fitted model for selected complexity λ^*) to approx. generalization error

Hypothetical implementation

104

Training set

Test set

1. Model selection

For each considered model complexity λ :

- i. Estimate parameters $\hat{\mathbf{w}}_\lambda$ on **training data**
- ii. Assess performance of $\hat{\mathbf{w}}_\lambda$ on **test data**
- iii. Choose λ^* to be λ with **lowest test error**

2. Model assessment

Compute test error of $\hat{\mathbf{w}}_{\lambda^*}$ (fitted model for selected complexity λ^*) to approx. generalization error

Overly optimistic!

Hypothetical implementation

105

Training set

Test set

Issue: Just like fitting $\hat{\mathbf{w}}$ and assessing its performance both on training data

- λ^* was selected to minimize **test error** (i.e., λ^* was fit on test data)
- If test data is not representative of the whole world, then $\hat{\mathbf{w}}_{\lambda^*}$ will typically perform worse than **test error** indicates

Practical implementation

106

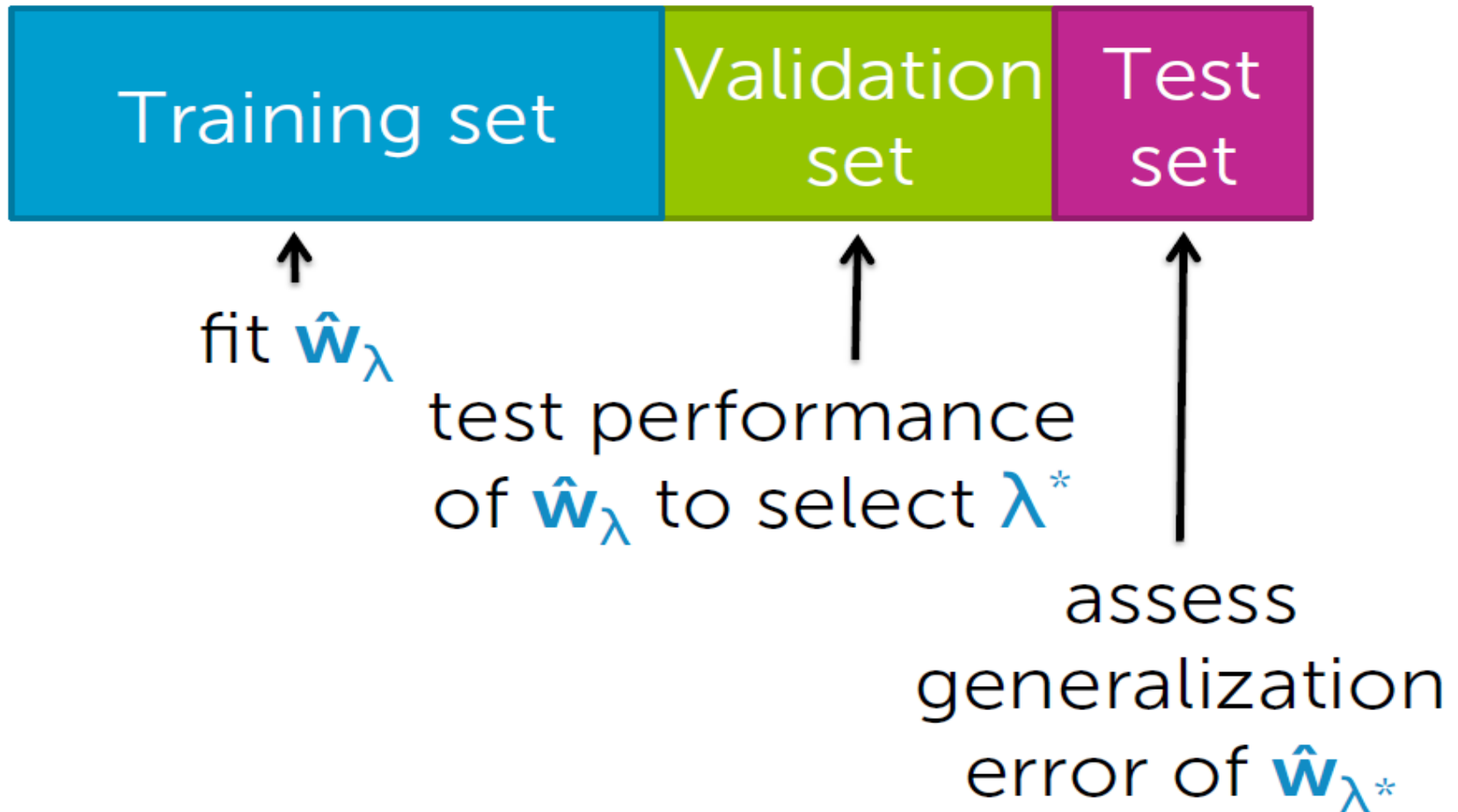


Solution: Create two “test” sets!

1. Select λ^* such that $\hat{\mathbf{w}}_{\lambda^*}$ minimizes error on validation set
2. Approximate generalization error of $\hat{\mathbf{w}}_{\lambda^*}$ using test set

Practical implementation

107



Typical splits

108



80%

10%

10%

50%

25%

25%

What you can do now

109

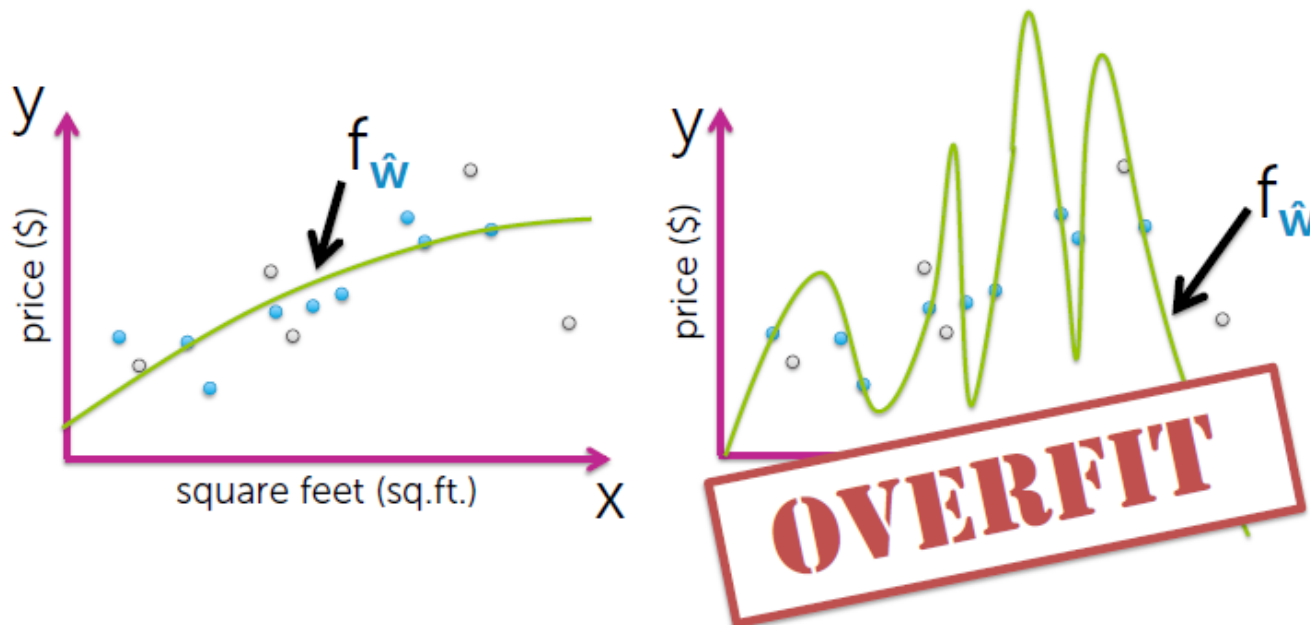
- Describe what a loss function is and give examples
- Contrast training, generalization, and test error
- Compute training and test error given a loss function
- Discuss issue of assessing performance on training set
- Describe tradeoffs in forming training/test splits
- List and interpret the 3 sources of avg. prediction error
 - Irreducible error, bias, and variance
- Discuss issue of selecting model complexity on test data and then using test error to assess generalization error
- Motivate use of a validation set for selecting tuning parameters (e.g., model complexity)
- Describe overall regression workflow

RIDGE REGRESSION

Flexibility of high-order polynomials

111

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



Symptoms for overfitting: often associated with very large value of estimated parameters \hat{w}

Overfitting with many features

112

Not unique to polynomial regression,
but also if **lots of inputs** (d large)

Or, generically,
lots of features (D large)

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

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

How does # of observations influence overfitting?

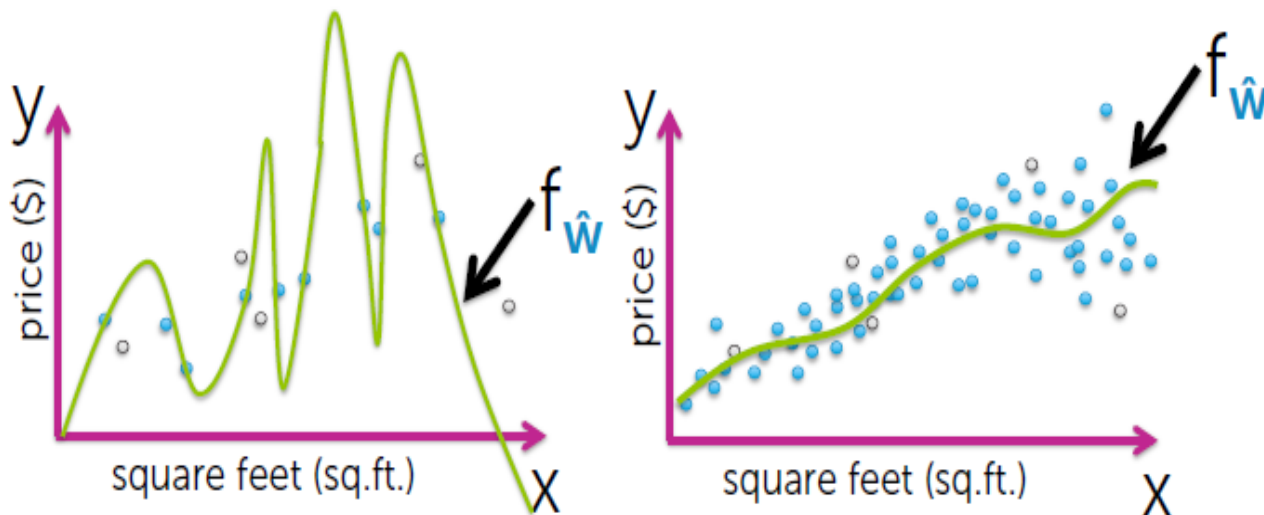
113

Few observations (N small)

→ rapidly overfit as model complexity increases

Many observations (N very large)

→ harder to overfit



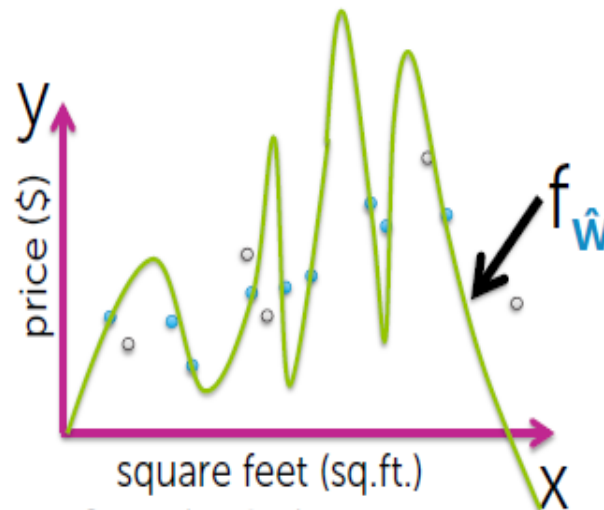
How does # of inputs influence overfitting?

114

1 input (e.g., sq.ft.):

Data must include representative examples of all possible (sq.ft., \$) pairs to avoid overfitting

HARD



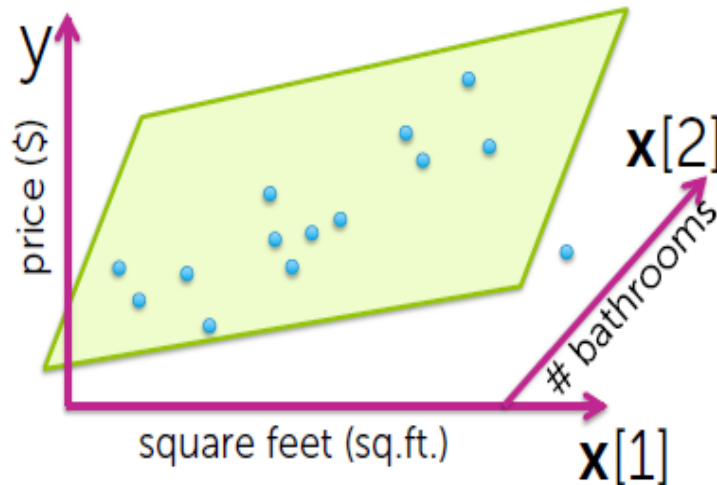
How does # of inputs influence overfitting?

115

d inputs (e.g., sq.ft., #bath, #bed, lot size, year,...):

Data must include examples of all possible (sq.ft., #bath, #bed, lot size, year,...., \$) combos to avoid overfitting

**MUCH!!!
HARDER**

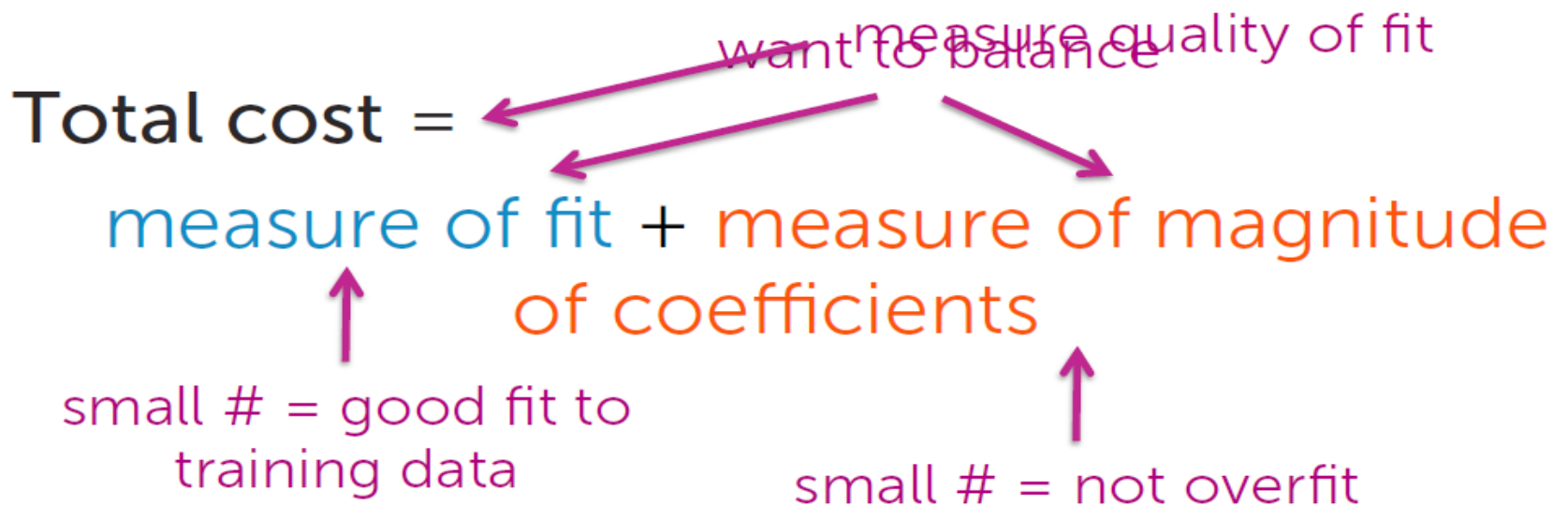


Desire total cost format

117

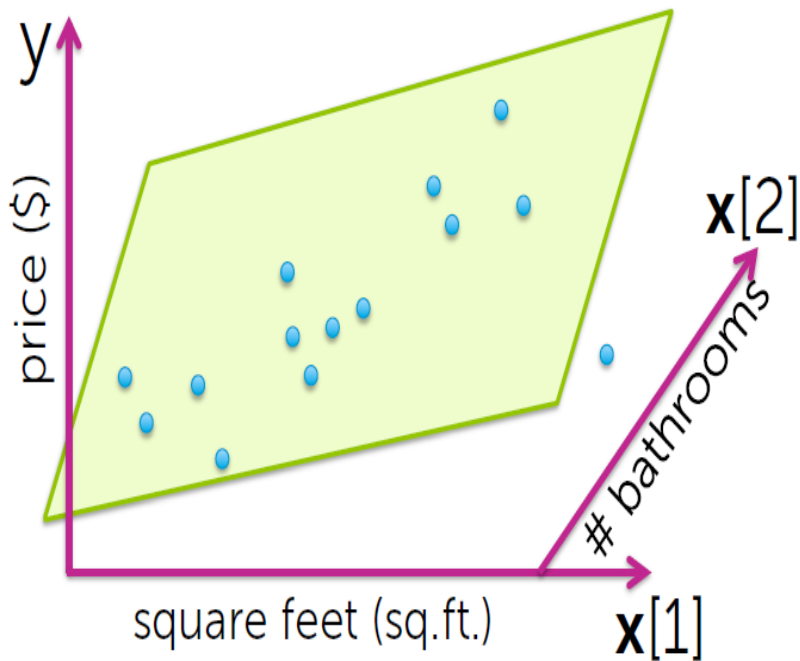
Want to balance:

- i. How well function fits data
- ii. Magnitude of coefficients



Measure of fit to training data

118



$$\begin{aligned} \text{RSS}(\mathbf{w}) &= \sum_{i=1}^N (y_i - h(\mathbf{x}_i)^T \mathbf{w})^2 \\ &= \sum_{i=1}^N (y_i - \hat{y}_i(\mathbf{w}))^2 \end{aligned}$$

pred. value using \mathbf{w}

small RSS \rightarrow model fitting training data well

Measure of magnitude of regression coefficients

119

What summary # is indicative of size of regression coefficients?

- Sum? $w_0 = 1,527,301$ $w_1 = -1,605,253$
 $w_0 + w_1 = \text{small } \#$

← **But ... the coefficients are very large**

- Sum of absolute value?
 $|w_0| + |w_1| + \dots + |w_D| = \sum_{j=0}^D |w_j| \triangleq \|w\|_1$ L_1 norm ... discuss more in next module
- Sum of squares (L_2 norm)
 $w_0^2 + w_1^2 + \dots + w_D^2 = \sum_{j=0}^D w_j^2 \triangleq \|w\|_2^2$ L_2 norm ... focus of this module

Consider specific total cost

120

Total cost =

measure of fit + measure of magnitude
of coefficients

$RSS(\mathbf{w})$ $\|\mathbf{w}\|_2^2$

The diagram illustrates the decomposition of the total cost function. It shows the text 'measure of fit + measure of magnitude of coefficients' in blue and orange. Below this, a purple bracket underlines 'measure of fit' and points to the term $RSS(\mathbf{w})$. Another purple bracket underlines 'measure of magnitude of coefficients' and points to the term $\|\mathbf{w}\|_2^2$.

Consider resulting objectives

121

What if $\hat{\mathbf{w}}$ selected to minimize

$$\text{RSS}(\mathbf{w}) + \lambda \|\mathbf{w}\|_2^2$$

λ tuning parameter = balance of fit and magnitude

Ridge regression
(a.k.a L_2 regularization)

If $\lambda=0$:

reduces to minimizing $\text{RSS}(\mathbf{w})$, as before (old solution) $\rightarrow \hat{\mathbf{w}}^{\text{LS}}$ ← least squares

If $\lambda=\infty$:

For solutions where $\hat{\mathbf{w}} \neq 0$, then total cost is ∞

If $\hat{\mathbf{w}}=0$, then total cost = $\text{RSS}(0)$ \rightarrow solution is $\hat{\mathbf{w}}=0$

If λ in between: Then $0 \leq \|\hat{\mathbf{w}}\|_2^2 \leq \|\hat{\mathbf{w}}^{\text{LS}}\|_2^2$

Ridge regression: bias-variance tradeoff

122

Large λ :

high bias, low variance

(e.g., $\hat{\mathbf{w}} = 0$ for $\lambda = \infty$)

In essence, λ
controls model
complexity

Small λ :

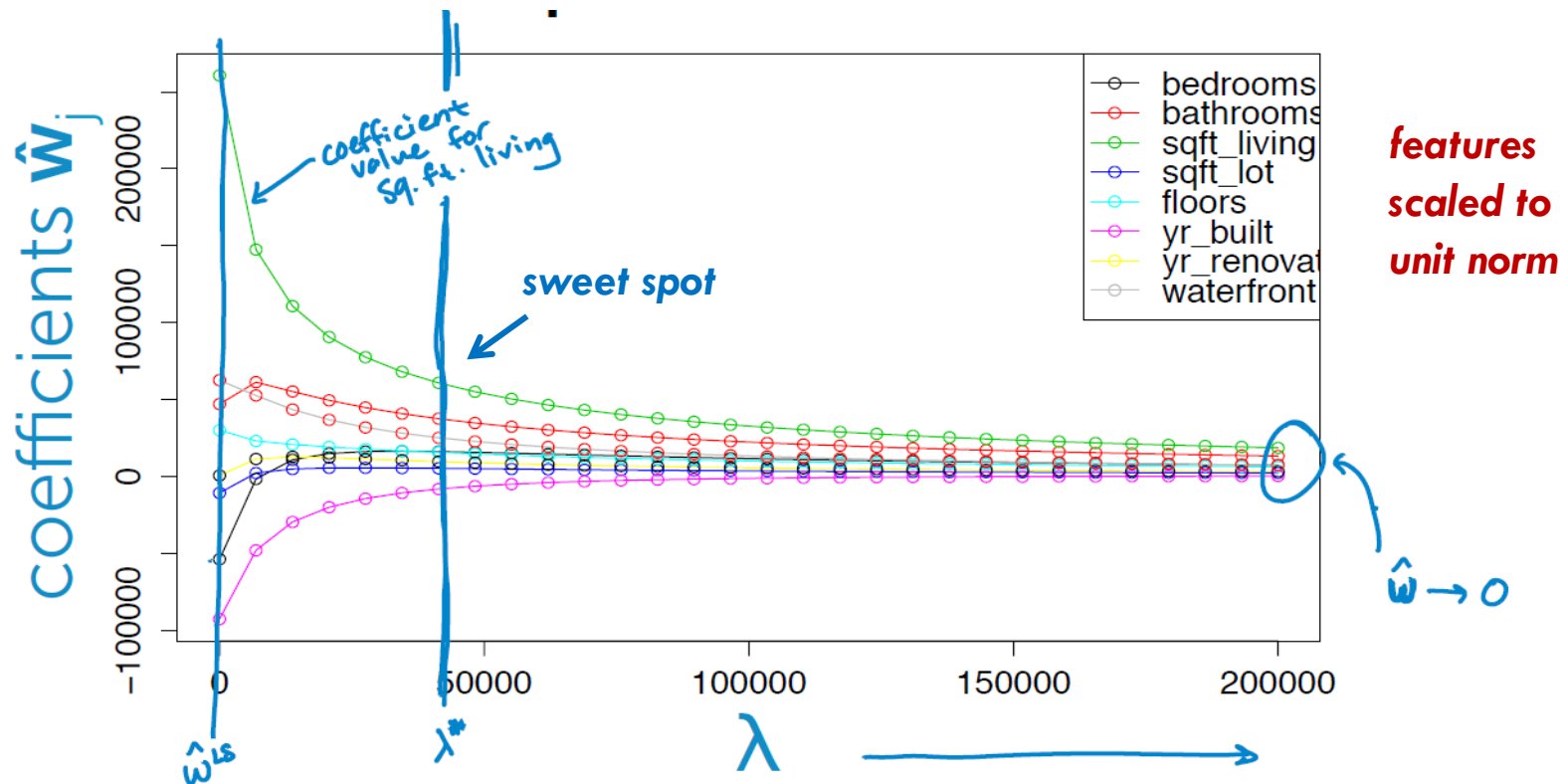
low bias, high variance

(e.g., standard least squares (RSS) fit of
high-order polynomial for $\lambda = 0$)

Ridge regression: coefficients path

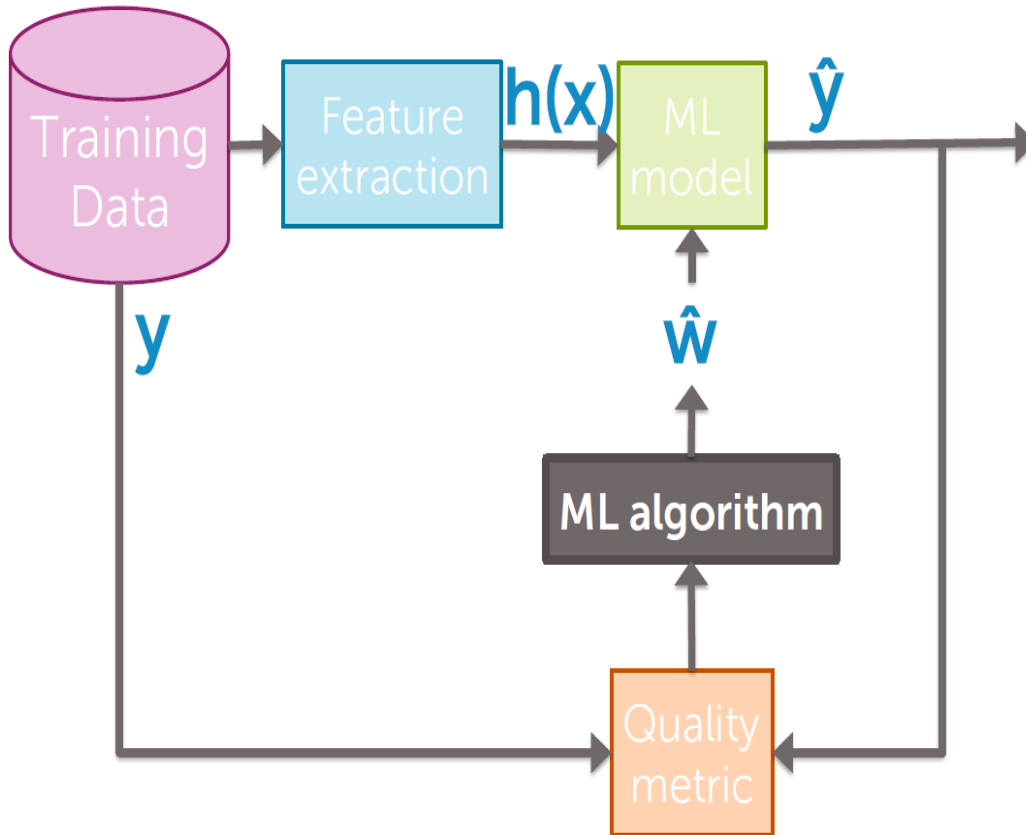
123

What happens if we refit our high-order polynomial, but now using ridge regression?



Flow chart

124



Model for all N observations together

$$\mathbf{y} = \mathbf{H} \mathbf{w} + \boldsymbol{\epsilon}$$

Gradient of ridge regression cost

126

$$\begin{aligned}\nabla [\text{RSS}(\mathbf{w}) + \lambda \|\mathbf{w}\|_2^2] &= \nabla [(\mathbf{y} - \mathbf{H}\mathbf{w})^\top (\mathbf{y} - \mathbf{H}\mathbf{w}) + \lambda \mathbf{w}^\top \mathbf{w}] \\ &= \underbrace{\nabla [(\mathbf{y} - \mathbf{H}\mathbf{w})^\top (\mathbf{y} - \mathbf{H}\mathbf{w})]}_{-2\mathbf{H}^\top (\mathbf{y} - \mathbf{H}\mathbf{w})} + \lambda \underbrace{\nabla [\mathbf{w}^\top \mathbf{w}]}_{2\mathbf{w}}\end{aligned}$$

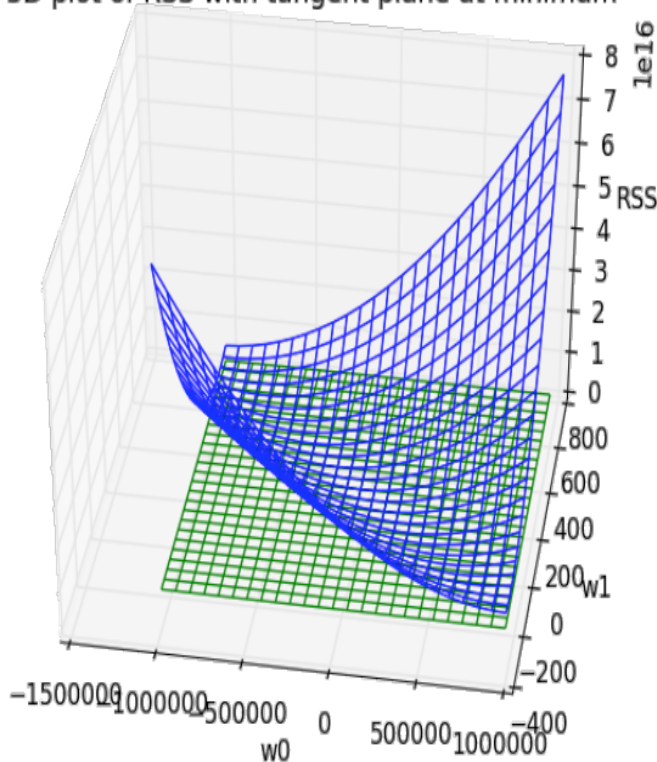
Why? By analogy to 1d case...

$\mathbf{w}^\top \mathbf{w}$ analogous to w^2 and derivative of $w^2 = 2w$

Ridge regression: closed-form solution

127

3D plot of RSS with tangent plane at minimum



$$\nabla \text{cost}(\mathbf{w}) = -2\mathbf{H}^T(\mathbf{y} - \mathbf{H}\mathbf{w}) + 2\lambda\mathbf{I}\mathbf{w} = 0$$

Solve for $\hat{\mathbf{w}}$: $-\mathbf{H}^T\mathbf{y} + \mathbf{H}^T\mathbf{H}\hat{\mathbf{w}} + \lambda\mathbf{I}\hat{\mathbf{w}} = 0$

$$\mathbf{H}^T\mathbf{H}\hat{\mathbf{w}} + \lambda\mathbf{I}\hat{\mathbf{w}} = \mathbf{H}^T\mathbf{y}$$

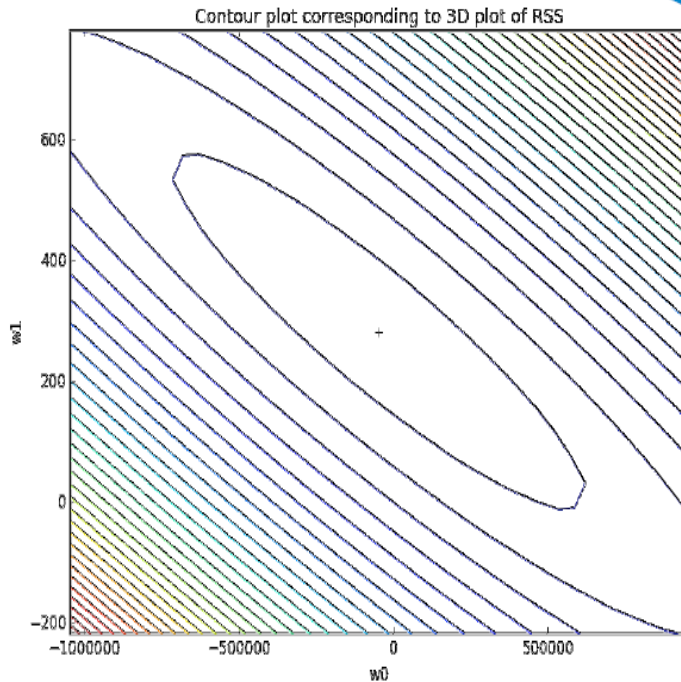
$$(\mathbf{H}^T\mathbf{H} + \lambda\mathbf{I})\hat{\mathbf{w}} = \mathbf{H}^T\mathbf{y}$$

$$\hat{\mathbf{w}}^{\text{ridge}} = (\mathbf{H}^T\mathbf{H} + \lambda\mathbf{I})^{-1}\mathbf{H}^T\mathbf{y}$$

Ridge regression: gradient descent

128

$$\nabla \text{cost}(\mathbf{w}) = -2\mathbf{H}^T(\mathbf{y}-\mathbf{H}\mathbf{w}) + 2\lambda\mathbf{w}$$



Update to j^{th} feature weight:

$$w_j^{(t+1)} \leftarrow \underline{w_j^{(t)}} - \eta *$$

Same as before (from RSS term) \rightarrow $-2 \sum_{i=1}^N (\mathbf{x}_i)(y_i - \hat{y}_i(\mathbf{w}^{(t)}))$

$+ 2\lambda \underline{w_j^{(t)}}$
 new term, comes from the j^{th} component of $2\lambda \mathbf{w}$

Summary of ridge regression algorithm

129

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

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

for $j = 0, \dots, D$

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

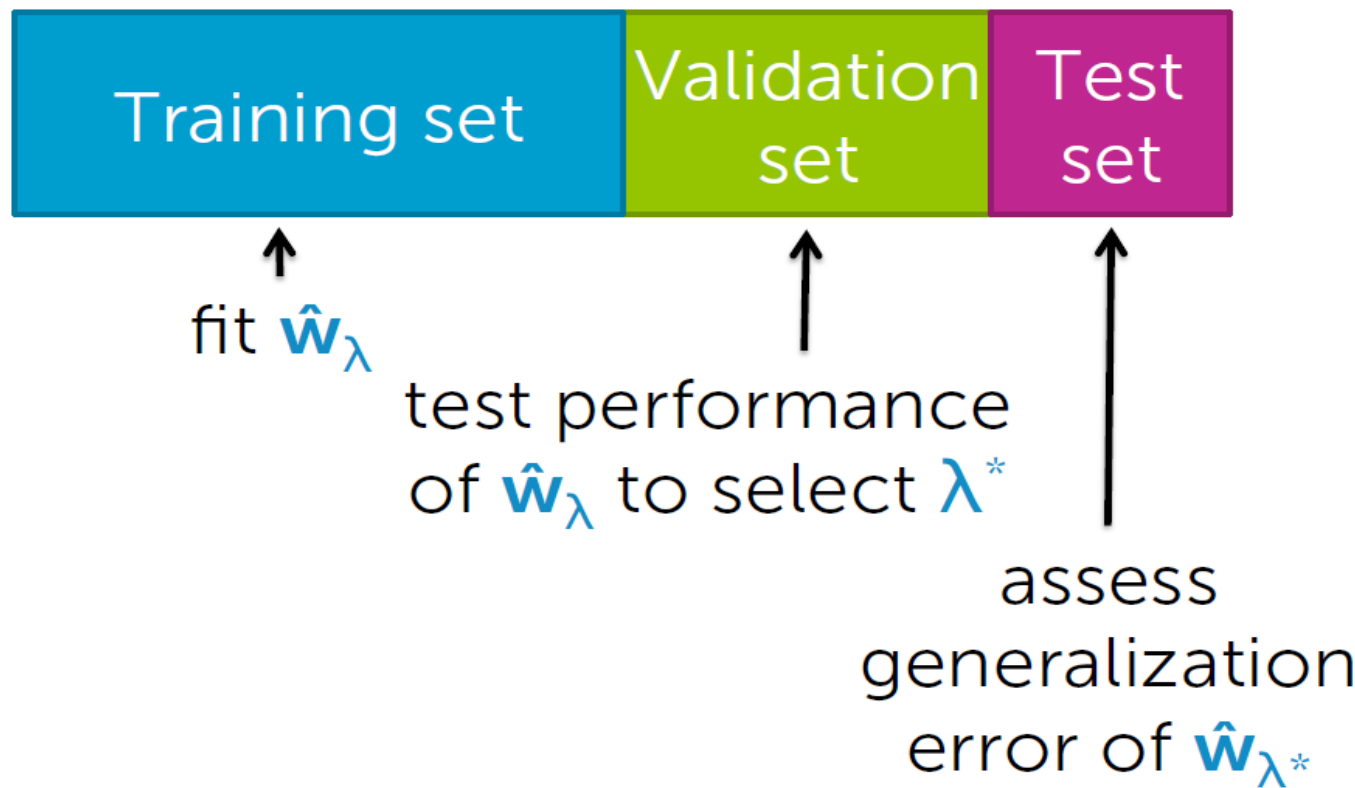
$w_j^{(t+1)} \leftarrow (1 - 2\eta\lambda)w_j^{(t)} - \eta \text{partial}[j]$

$t \leftarrow t + 1$

How to choose λ

130

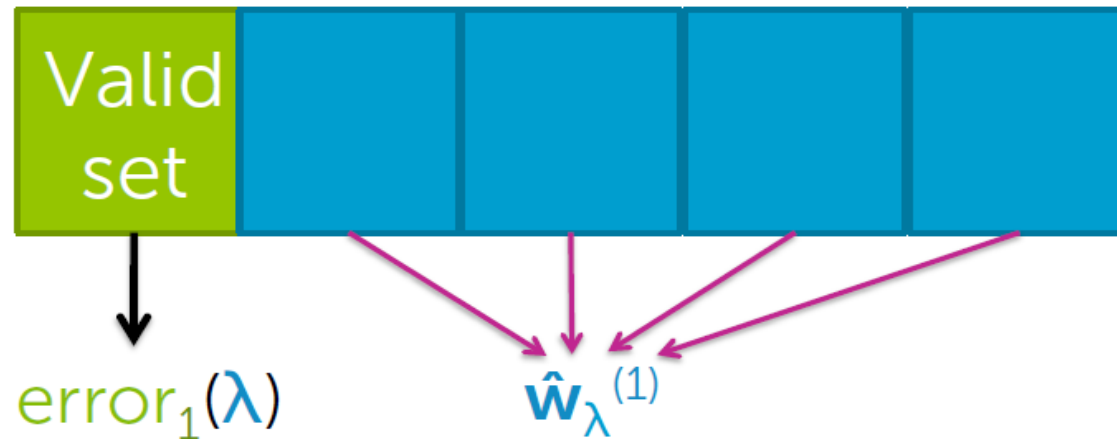
If sufficient amount of data...



How to choose λ

131

K-fold cross validation



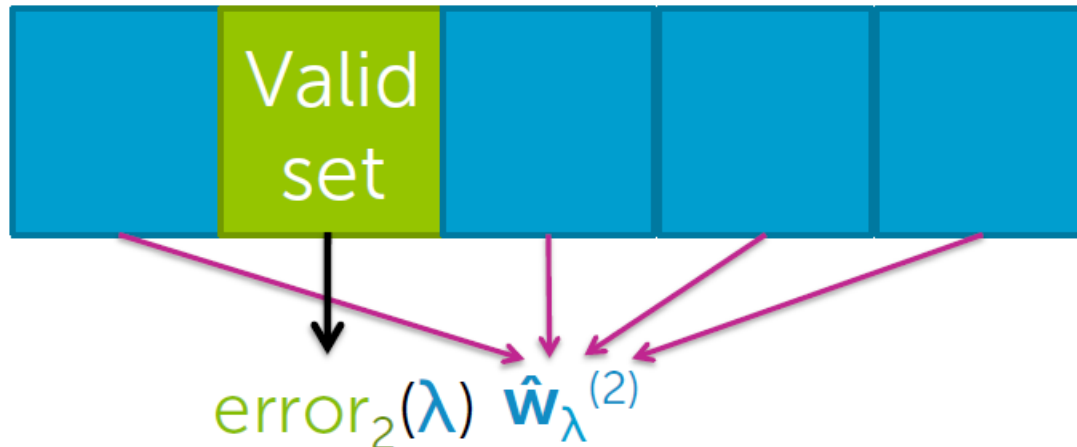
For $k=1, \dots, K$

1. Estimate $\hat{w}_\lambda^{(k)}$ on the training blocks
2. Compute error on validation block: $error_k(\lambda)$

How to choose λ

132

K-fold cross validation



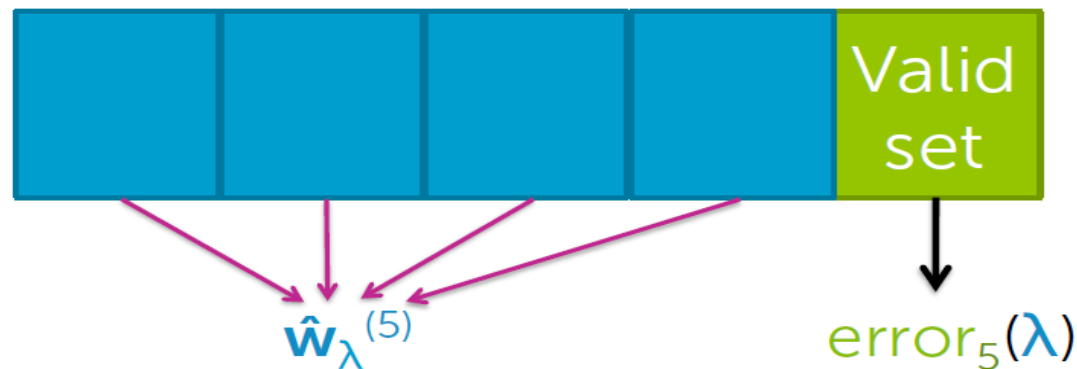
For $k=1, \dots, K$

1. Estimate $\hat{w}_\lambda^{(k)}$ on the training blocks
2. Compute error on validation block: $error_k(\lambda)$

How to choose λ

133

K-fold cross validation



For $k=1, \dots, K$

1. Estimate $\hat{w}_\lambda^{(k)}$ on the training blocks
2. Compute error on validation block: $error_k(\lambda)$

Compute average error: $CV(\lambda) = \frac{1}{K} \sum_{k=1}^K error_k(\lambda)$

How to choose λ

134

K-fold cross validation



Repeat procedure for each choice of λ

Choose λ^* to minimize $CV(\lambda)$

What value of K

135

Formally, the **best approximation** occurs for validation sets of size 1 ($K=N$)

leave-one-out
cross validation

Computationally intensive

- requires computing N fits of model per λ

Typically, $K=5$ or 10

5-fold CV

10-fold CV

How to handle the intercept

136

Recall multiple regression model

Model:

$$\begin{aligned} y_i &= w_0 h_0(\mathbf{x}_i) + w_1 h_1(\mathbf{x}_i) + \dots + w_D h_D(\mathbf{x}_i) + \varepsilon_i \\ &= \sum_{j=0}^D w_j h_j(\mathbf{x}_i) + \varepsilon_i \end{aligned}$$

feature 1 = $h_0(\mathbf{x})$...often 1 (constant)

feature 2 = $h_1(\mathbf{x})$... e.g., $\mathbf{x}[1]$

feature 3 = $h_2(\mathbf{x})$... e.g., $\mathbf{x}[2]$

...

feature D+1 = $h_D(\mathbf{x})$... e.g., $\mathbf{x}[d]$

Do we penalize intercept?

137

Standard ridge regression cost:

$$\text{RSS}(\mathbf{w}) + \lambda \|\mathbf{w}\|_2^2$$

 strength of penalty

Encourages intercept w_0 to also be small

Do we want a small intercept?

Conceptually, not indicative of overfitting...

Do we penalize intercept?

138

□ **Option 1: don't penalize intercept**

Modified ridge regression cost:

$$\text{RSS}(\mathbf{w}_0, \mathbf{w}_{\text{rest}}) + \lambda \|\mathbf{w}_{\text{rest}}\|_2^2$$

□ **Option 2: Center data first**

If data are first **centered about 0**, then favoring small intercept not so worrisome

Step 1: Transform y to have 0 mean

Step 2: Run ridge regression as normal
(closed-form or gradient algorithms)

What you can do now

139

- Describe what happens to magnitude of estimated coefficients when model is overfit
- Motivate form of ridge regression cost function
- Describe what happens to estimated coefficients of ridge regression as tuning parameter λ is varied
- Interpret coefficient path plot
- Estimate ridge regression parameters:
 - In closed form
 - Using an iterative gradient descent algorithm
- Implement K-fold cross validation to select the ridge regression tuning parameter λ

FEATURES SELECTION & LASSO REGRESSION

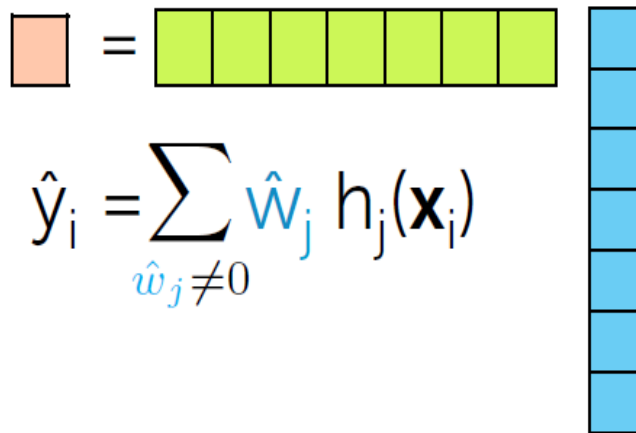
Why features selection?

141

Efficiency:

- If $\text{size}(\mathbf{w}) = 100\text{B}$, each prediction is expensive
- If $\hat{\mathbf{w}}$ **sparse**, computation only depends on # of non-zeros

many zeros


$$\hat{y}_i = \sum_{\hat{w}_j \neq 0} \hat{w}_j h_j(\mathbf{x}_i)$$

Interpretability:

- Which features are relevant for prediction?

Housing application

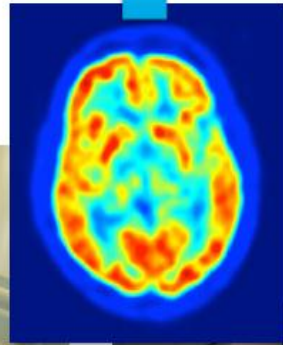


- Lot size
- Single Family
- Year built
- Last sold price
- Last sale price/sqft
- Finished sqft
- Unfinished sqft
- Finished basement sqft
- # floors
- Flooring types
- Parking type
- Parking amount
- Cooling
- Heating
- Exterior materials
- Roof type
- Structure style
- Dishwasher
- Garbage disposal
- Microwave
- Range / Oven
- Refrigerator
- Washer
- Dryer
- Laundry location
- Heating type
- Jetted Tub
- Deck
- Fenced Yard
- Lawn
- Garden
- Sprinkler System
- ⋮

Sparcity

143

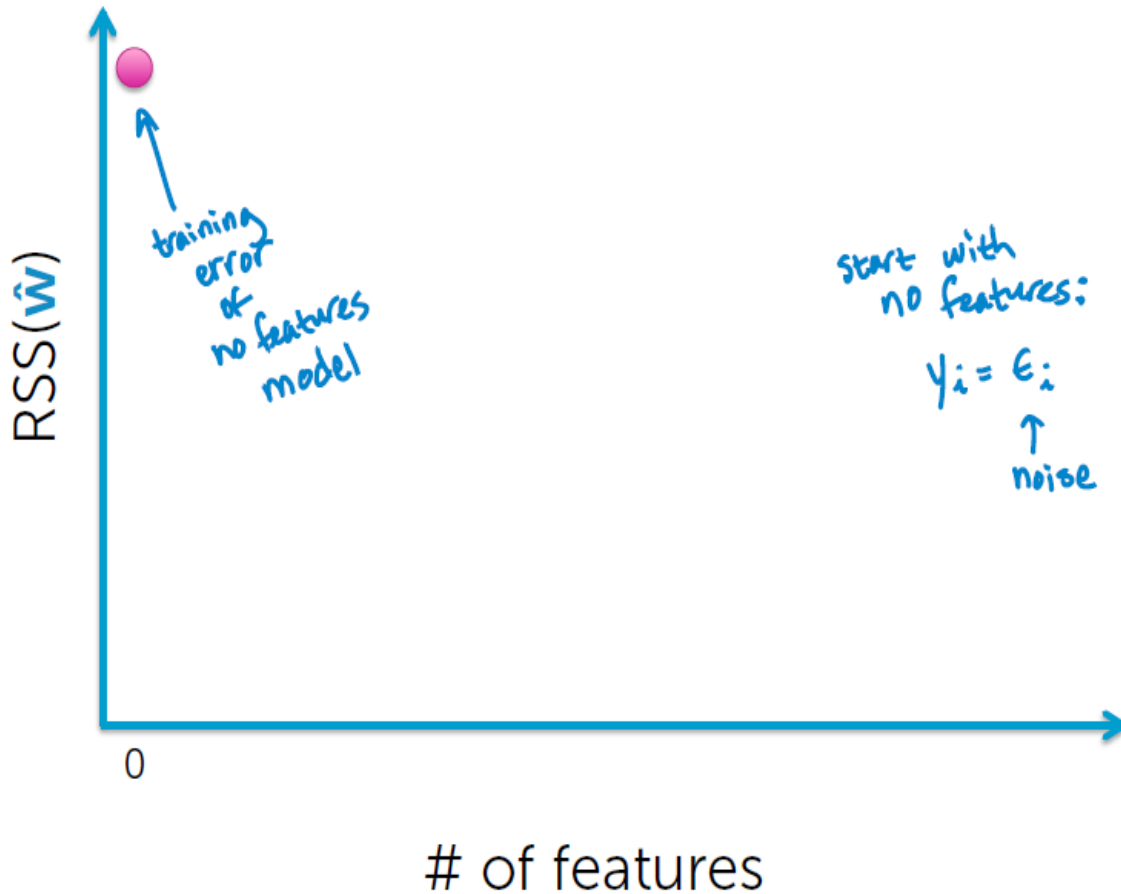
Reading your mind



Activity in which
brain regions
can predict
happiness?

Find best model of size: 0

144

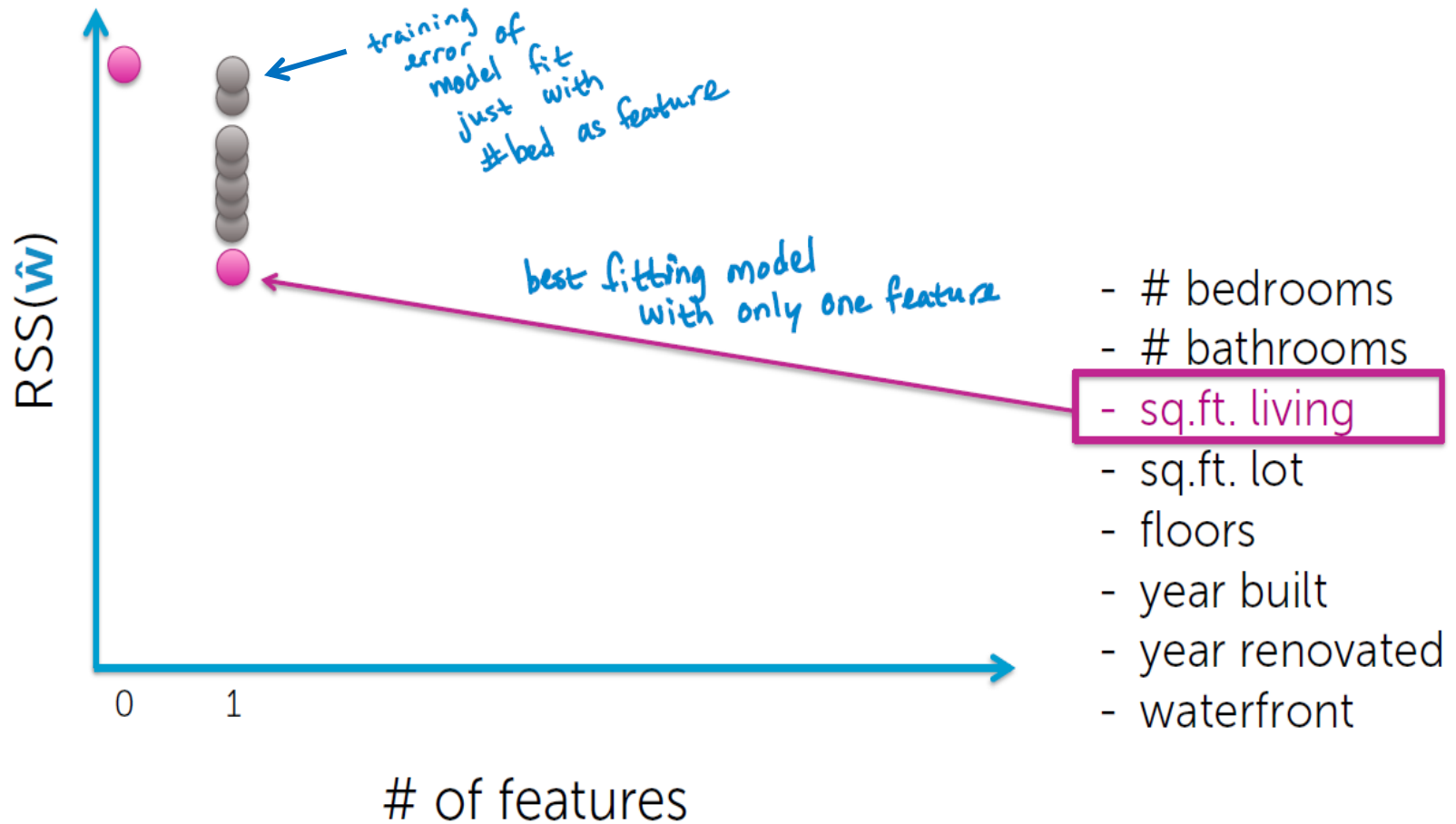


start with
no features:
 $y_i = \epsilon_i$
↑
noise

- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Find best model of size: 1

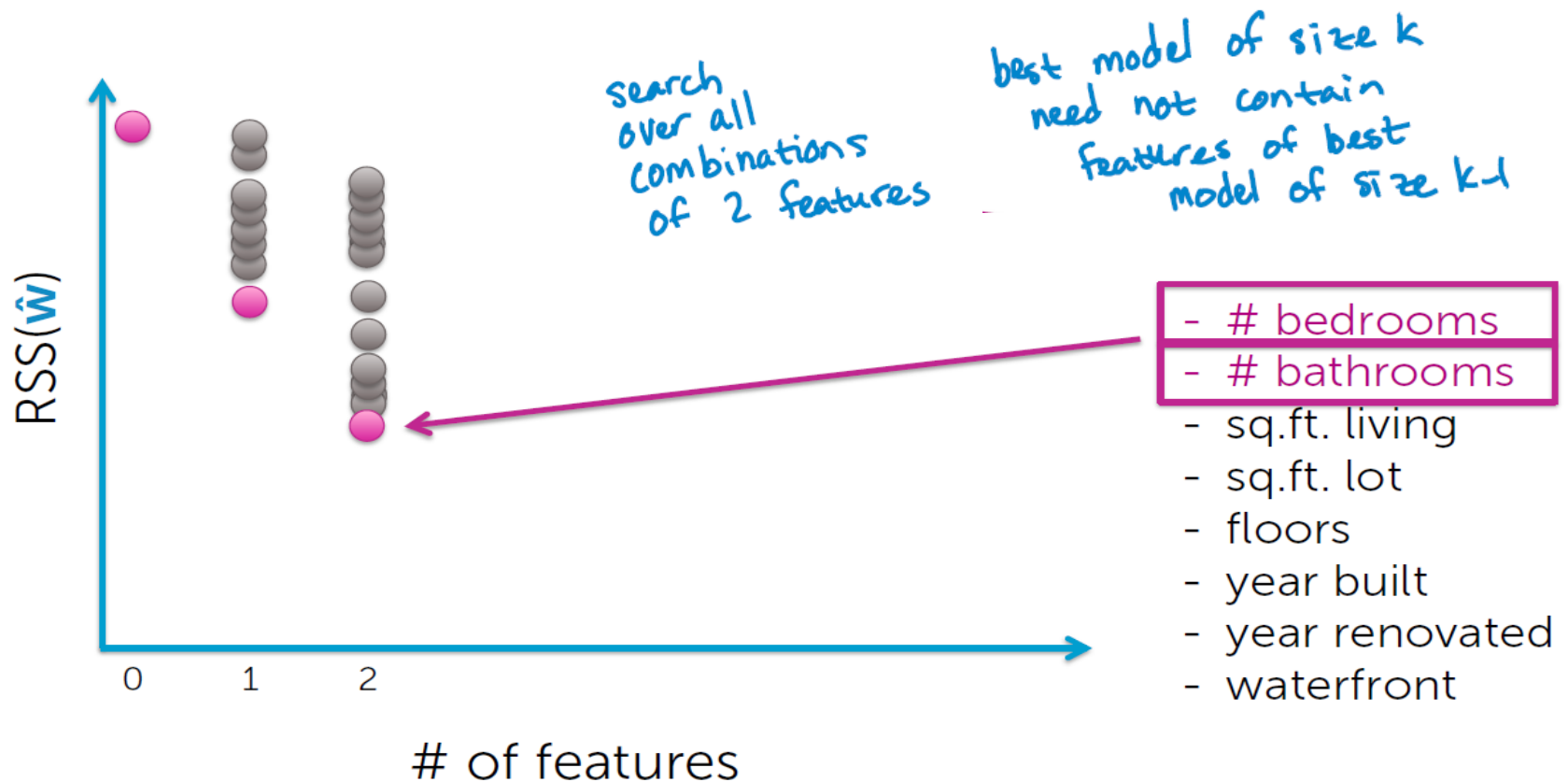
145



Find best model of size: 2

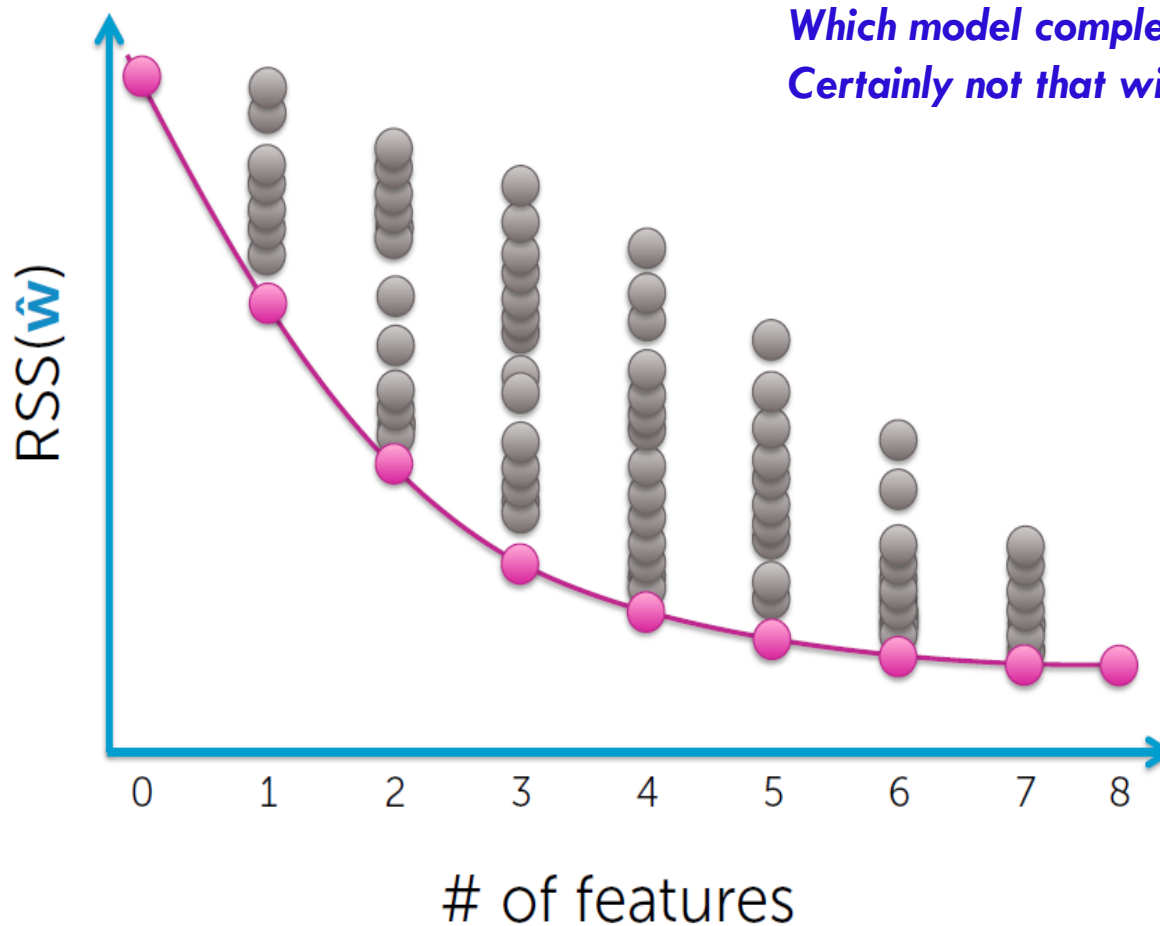
146

Note: not necessarily nested!



Find best model of size: N

147



Which model complexity to choose?

Certainly not that with the smallest training error!

- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Choosing model complexity

148

Option 1: Assess on validation set

Option 2: Cross validation

Option 3+: Other metrics for penalizing model complexity like BIC...

Complexity of „all subsets”

149

How many models were evaluated?

- each indexed by features included

$$y_i = \varepsilon_i$$

$$y_i = w_0 h_0(\mathbf{x}_i) + \varepsilon_i$$

$$y_i = w_1 h_1(\mathbf{x}_i) + \varepsilon_i$$

⋮

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

⋮

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

feature 0 ← 0 if "no"
1 if "yes"
feature 1 ... feature D

[0 0 0 ... 0 0 0]

[1 0 0 ... 0 0 0]

[0 1 0 ... 0 0 0]

⋮

[1 1 0 ... 0 0 0]

⋮

[1 1 1 ... 1 1 1]

2 2 2 ··· 2

2^{D+1}

$$2^8 = 256$$

$$2^{30} = 1,073,741,824$$

$$2^{1000} = 1.071509 \times 10^{301}$$

$$2^{100B} = \text{HUGE!!!!!!}$$

Typically,
computationally
infeasible

Greedy algorithm

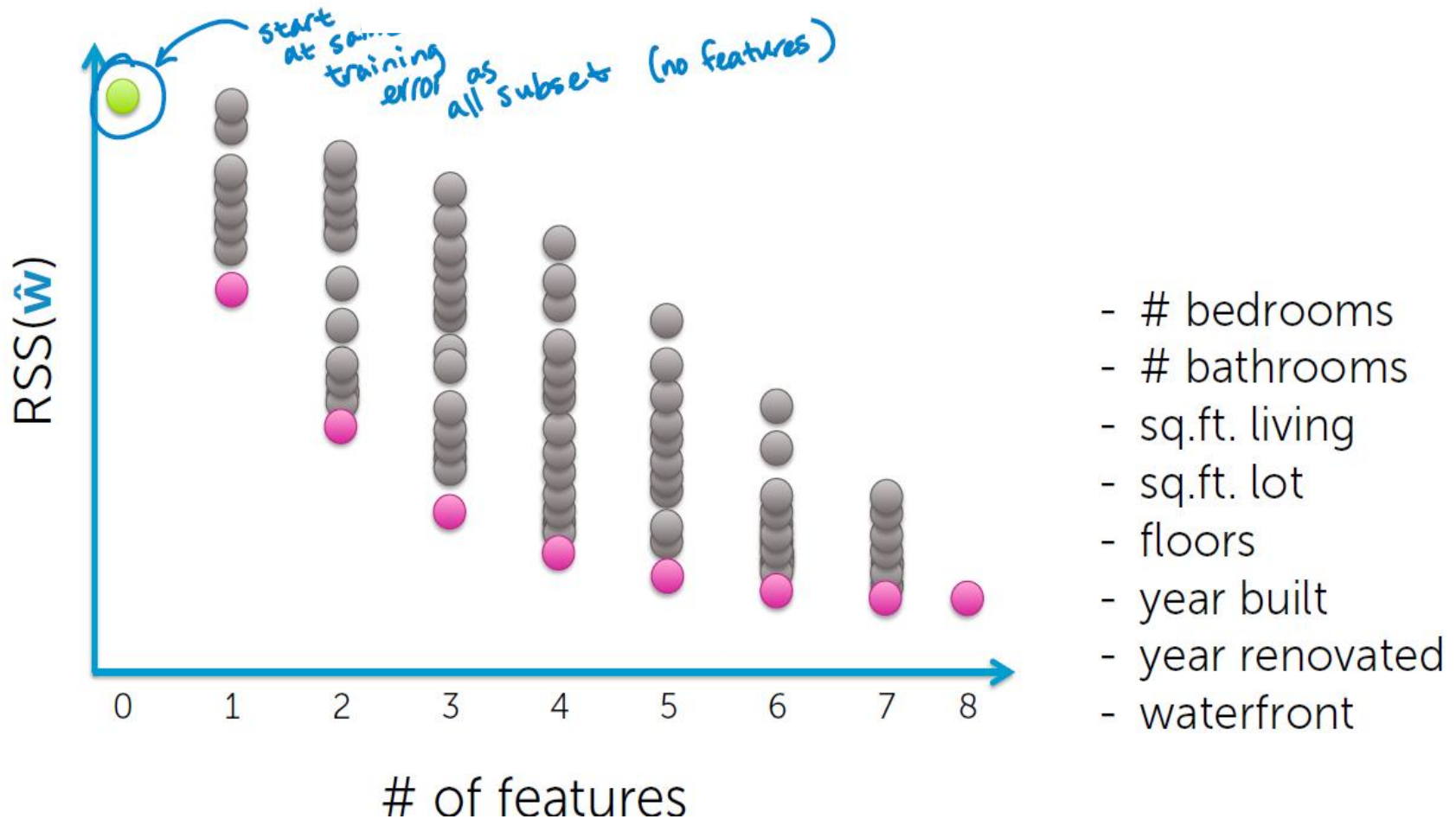
150

Forward stepwise algorithm

1. Pick a dictionary of features $\{h_0(\mathbf{x}), \dots, h_D(\mathbf{x})\}$
 - e.g., polynomials for linear regression
2. Greedy heuristic:
 - i. Start with empty set of features $F_0 = \emptyset$
(or simple set, like just $h_0(\mathbf{x})=1 \rightarrow y_i = w_0 + \epsilon_i$)
 - ii. Fit model using current feature set F_t to get $\hat{\mathbf{w}}^{(t)}$
 - iii. Select next best feature $h_{j^*}(\mathbf{x})$
 - e.g., $h_j(\mathbf{x})$ resulting in lowest training error when learning with $F_t + \{h_j(\mathbf{x})\}$
 - iv. Set $F_{t+1} \leftarrow F_t + \{h_{j^*}(\mathbf{x})\}$
 - v. Recurse

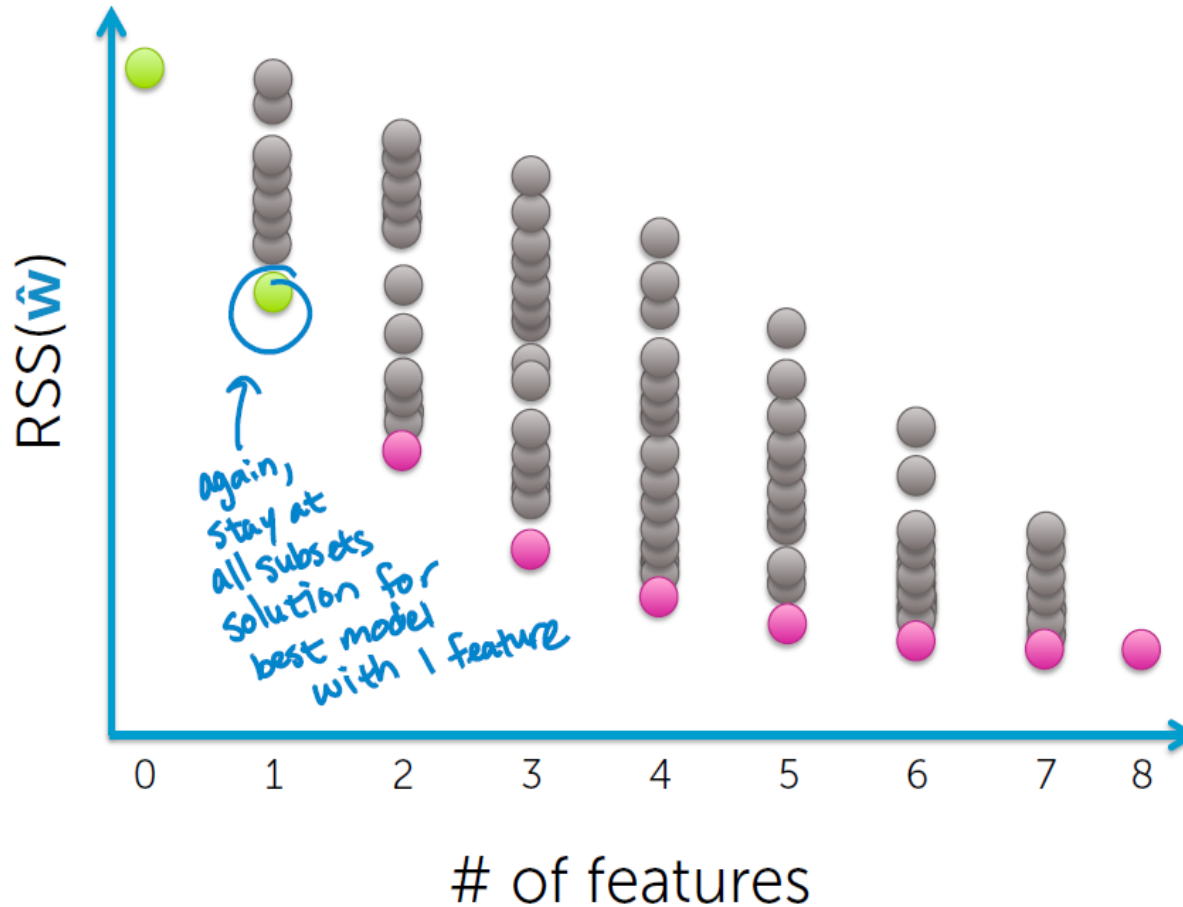
Visualizing greedy algorithm

151



Visualizing greedy algorithm

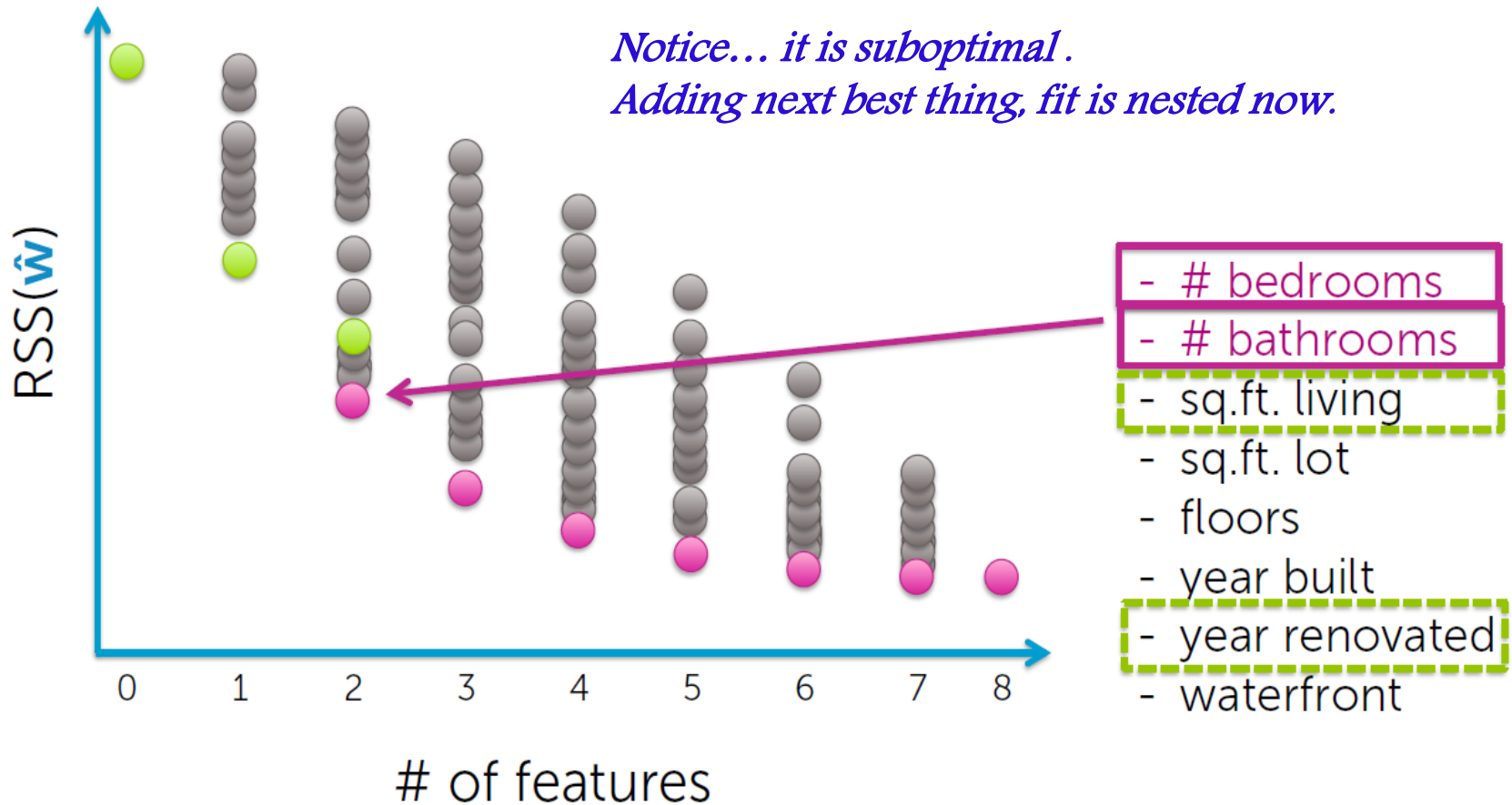
152



- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

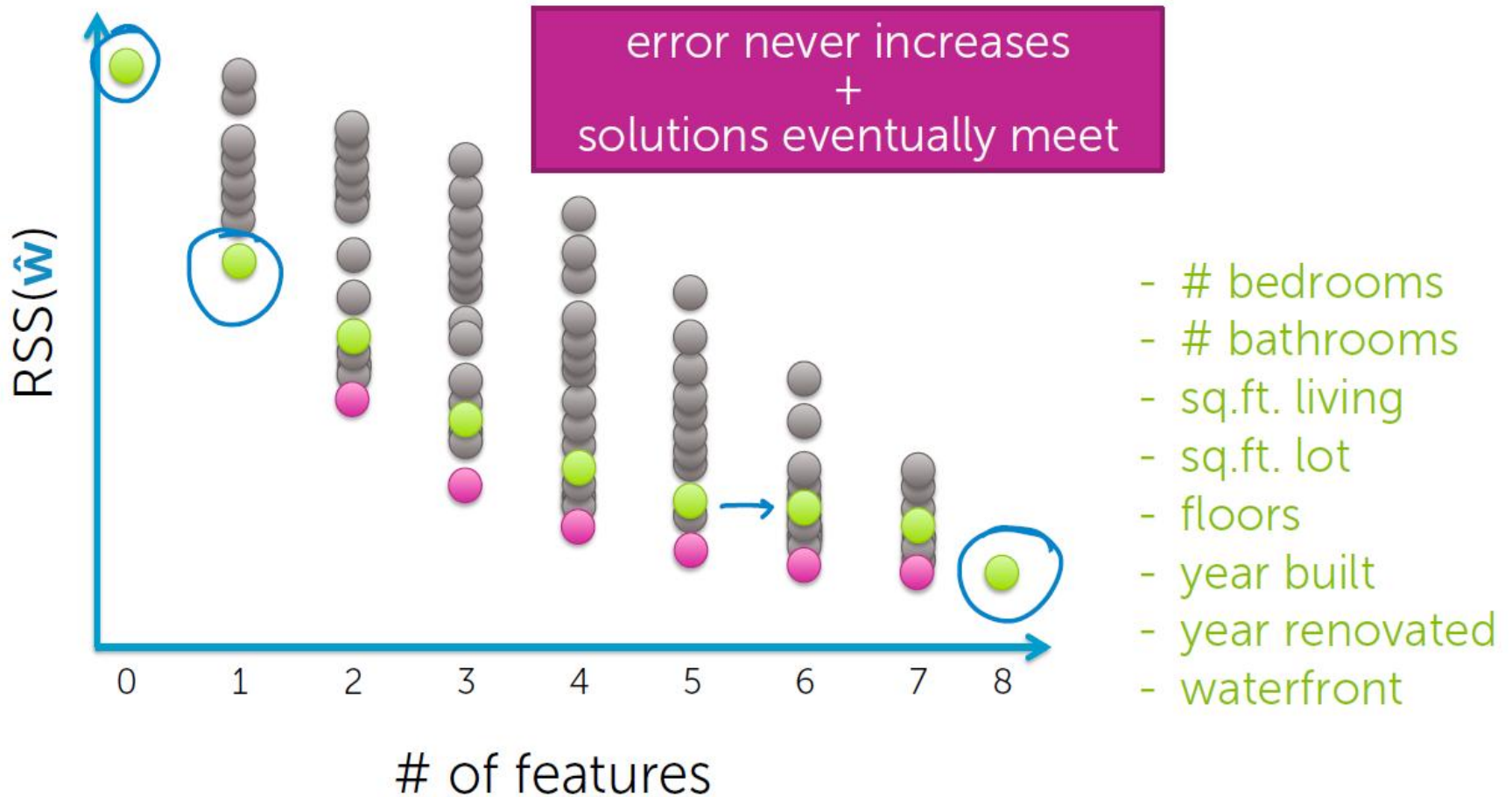
Visualizing greedy algorithm

153



Visualizing greedy algorithm

154



When do we stop?

155

When **training error** is low enough?

No!

When **test error** is low enough?

No!

Use validation set or cross validation!

Complexity of forward stepwise

156

How many models were evaluated?

- 1st step, D models
- 2nd step, $D-1$ models (add 1 feature out of $D-1$ possible)
- 3rd step, $D-2$ models (add 1 feature out of $D-2$ possible)
- ...

How many steps?

- Depends
- At most D steps (to full model)

$$O(D^2) \ll 2^D$$

for large D

Other greedy algorithms

157

Instead of starting from simple model and always growing...

Backward stepwise:

Start with full model and iteratively remove features least useful to fit

Combining forward and backward steps:

In forward algorithm, insert steps to remove features no longer as important

Lots of other variants, too.

Using regularisation for features selection

158

Instead of searching over a **discrete** set of solutions, can we use **regularization**?

- Start with full model (all possible features)
- “Shrink” some coefficients *exactly to 0*
 - i.e., knock out certain features
- Non-zero coefficients indicate “selected” features

Thresholding ridge coefficients?

159

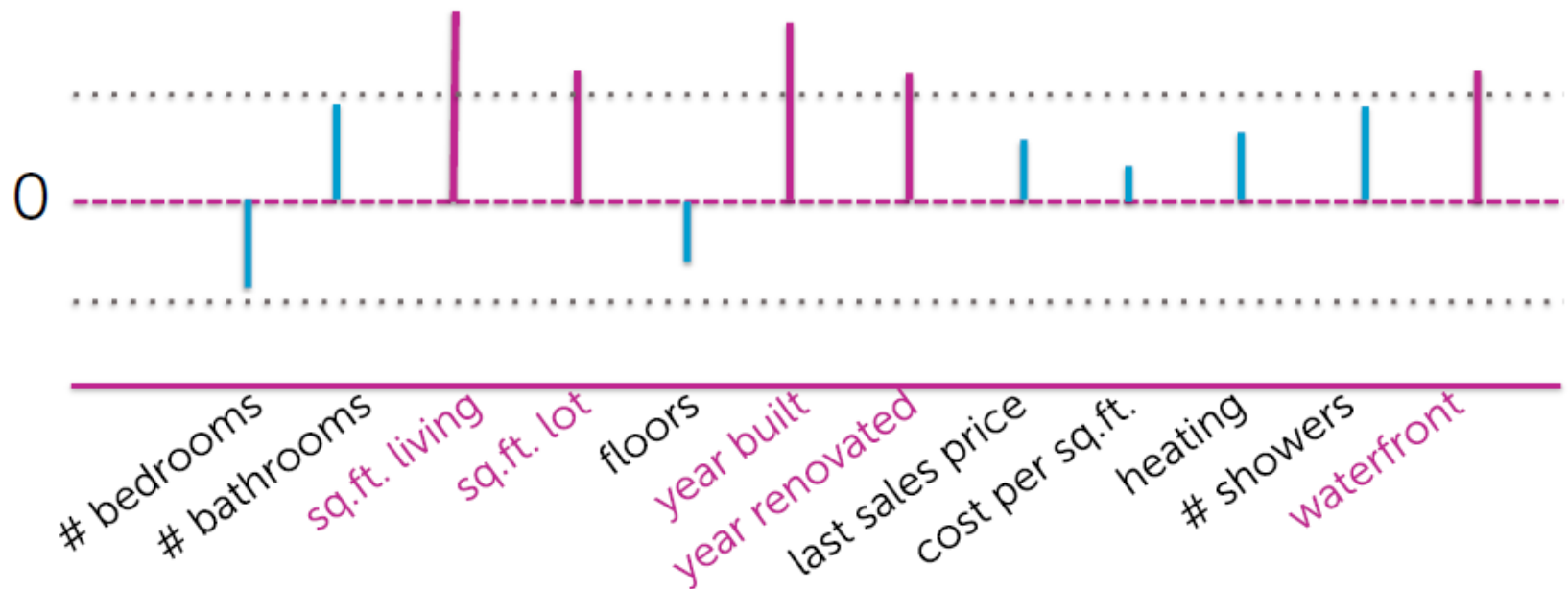
Why don't we just set small ridge coefficients to 0?



Thresholding ridge coefficients?

160

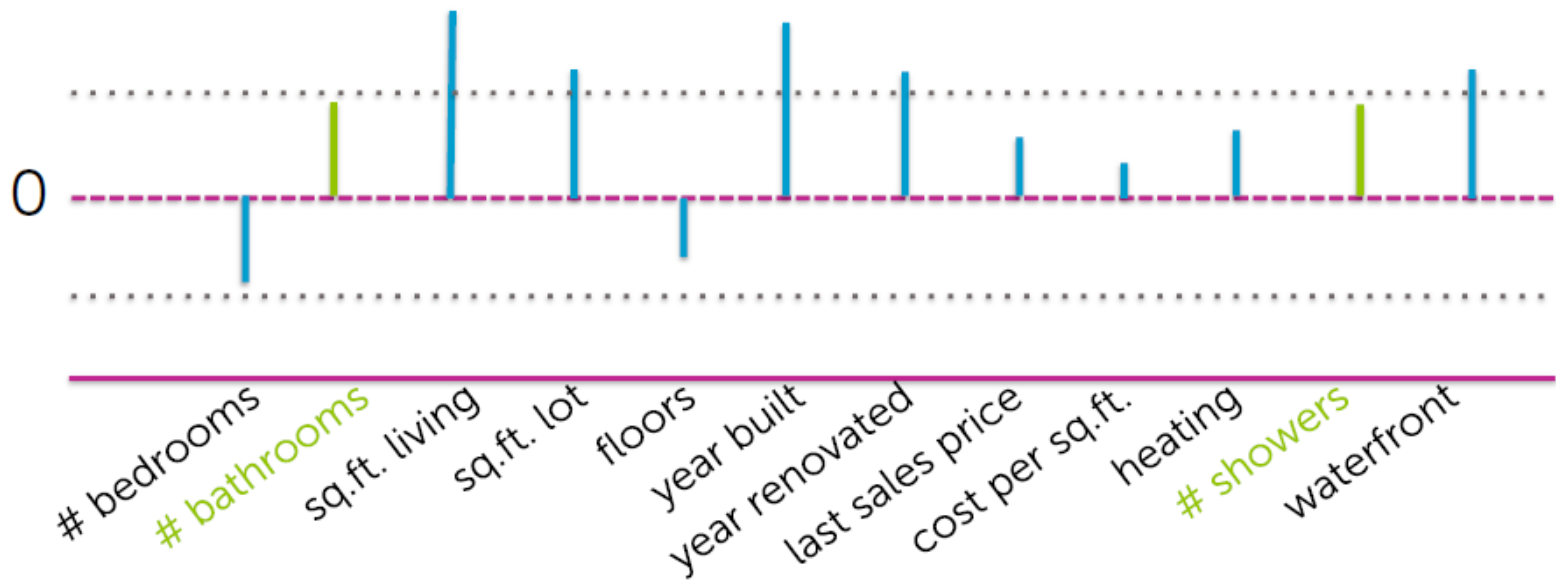
Selected features for a given threshold value



Thresholding ridge coefficients?

161

Let's look at two related features...

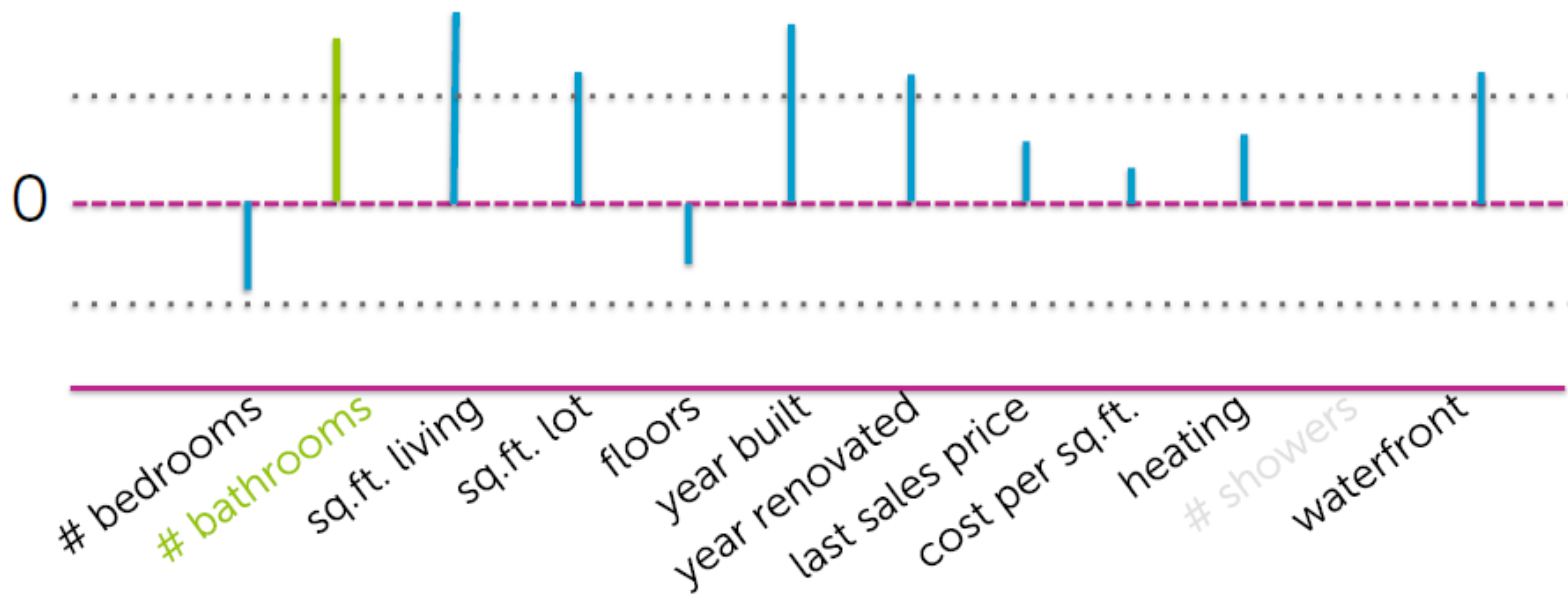


Nothing measuring bathrooms was included!

Thresholding ridge coefficients?

162

If only one of the features had been included...



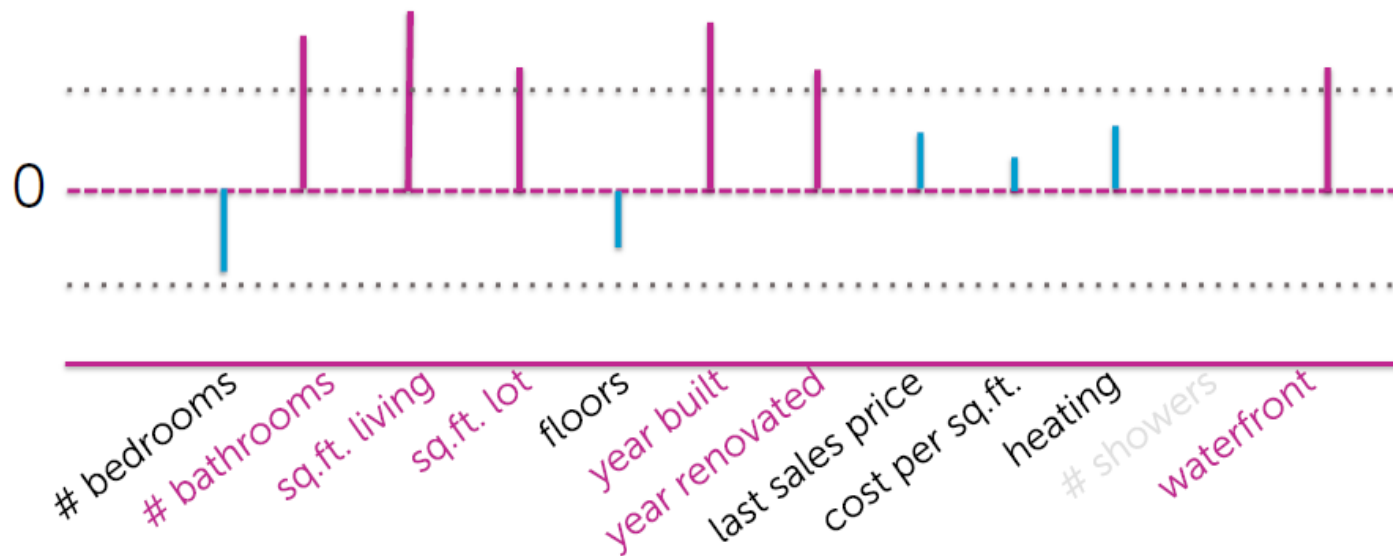
Remember:

this is linear model. If we assume that #showers = #bathrooms and remove one of them from the model, coefficients will sum up.

Thresholding ridge coefficients?

163

Would have included bathrooms in selected model



Can regularization lead directly to sparsity?

Try this cost instead of ridge ...

164

Total cost =

measure of fit + λ measure of magnitude
of coefficients

RSS(\mathbf{w})

$$\|\mathbf{w}\|_1 = |w_0| + \dots + |w_D|$$

Lasso regression
(a.k.a. L_1 regularized regression)


Leads to
sparse
solutions!

Lasso regression

165

Just like ridge regression, solution is governed by a continuous parameter λ

$$\text{RSS}(\mathbf{w}) + \lambda \|\mathbf{w}\|_1$$

 tuning parameter = balance of fit and sparsity

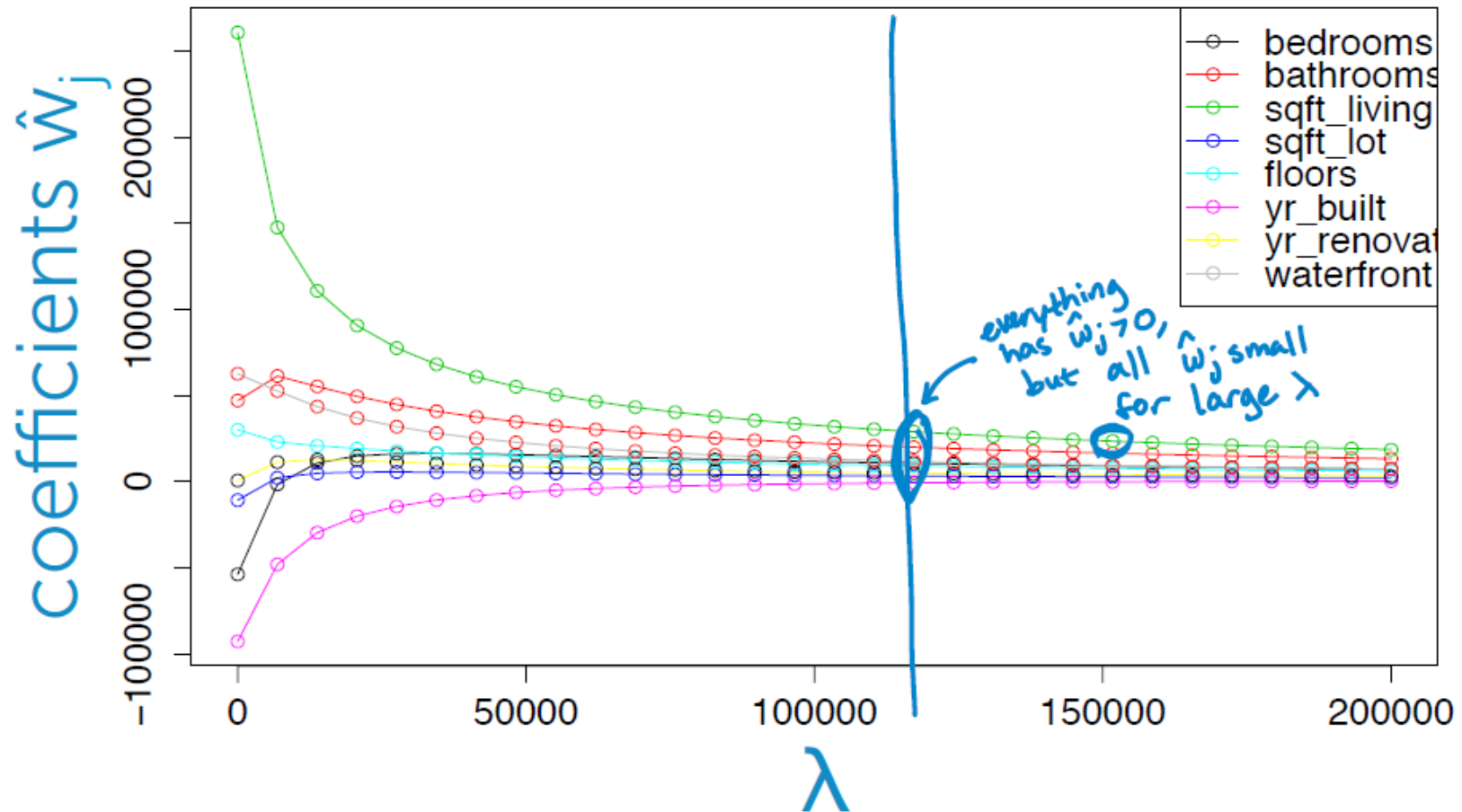
If $\lambda=0$: $\hat{\mathbf{w}}^{\text{lasso}} = \hat{\mathbf{w}}^{\text{LS}}$ (unregularized solution)

If $\lambda=\infty$: $\hat{\mathbf{w}}^{\text{lasso}} = \mathbf{0}$

If λ in between: $\mathbf{0} \leq \|\hat{\mathbf{w}}^{\text{lasso}}\|_1 \leq \|\hat{\mathbf{w}}^{\text{LS}}\|_1$

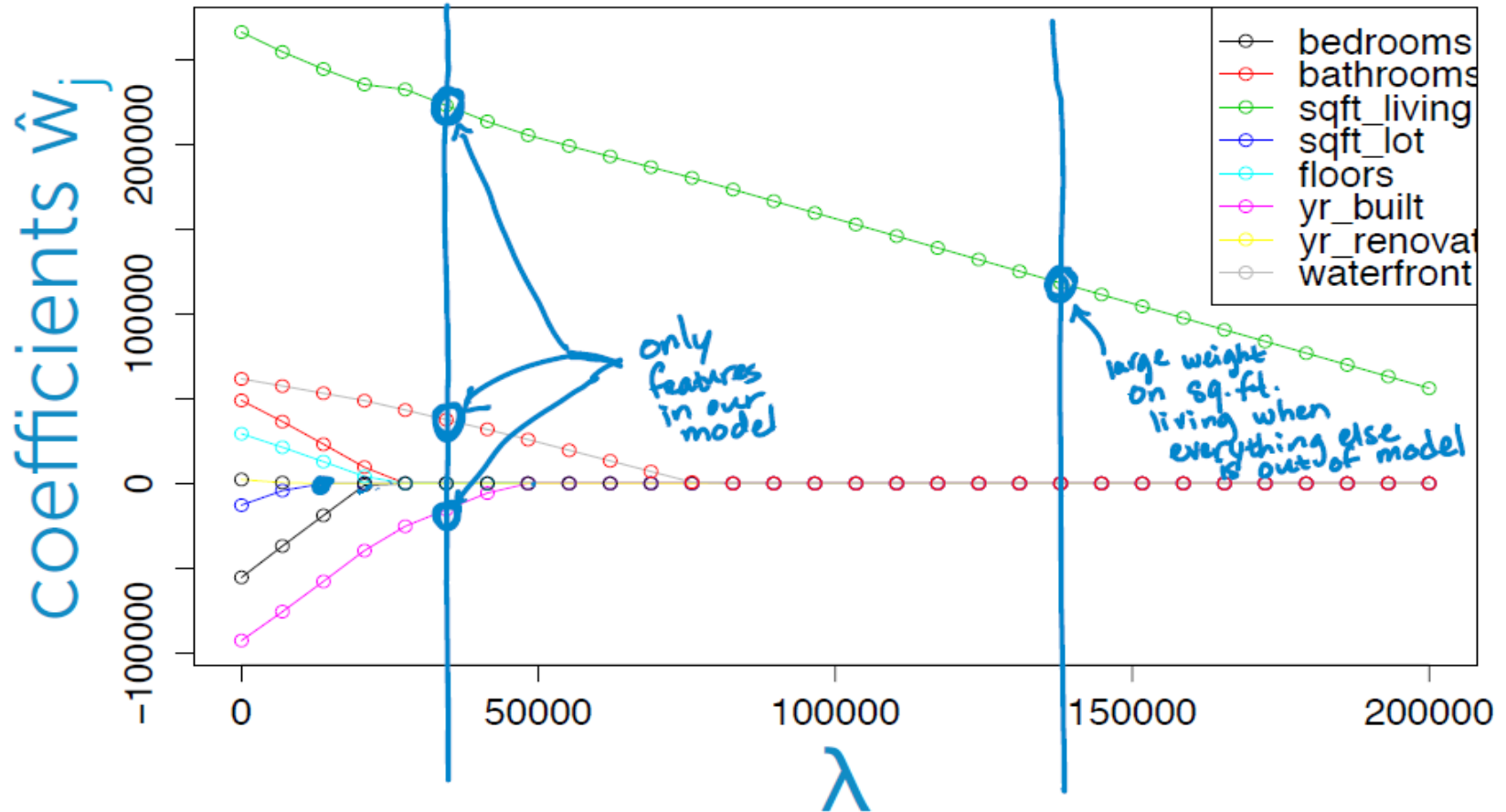
Coefficient path: ridge

166



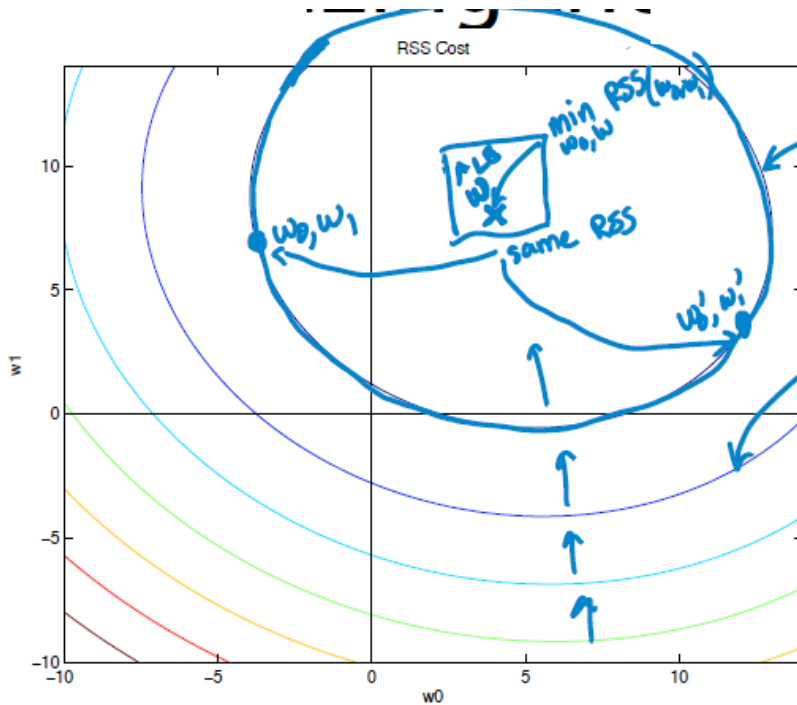
Coefficient path: lasso

167



Visualising ridge cost in 2D

168



$RSS(w_0, w_1) = \text{const}_1$

$RSS(w_0, w_1) = \text{const}_2 > \text{const}_1$
 \vdots

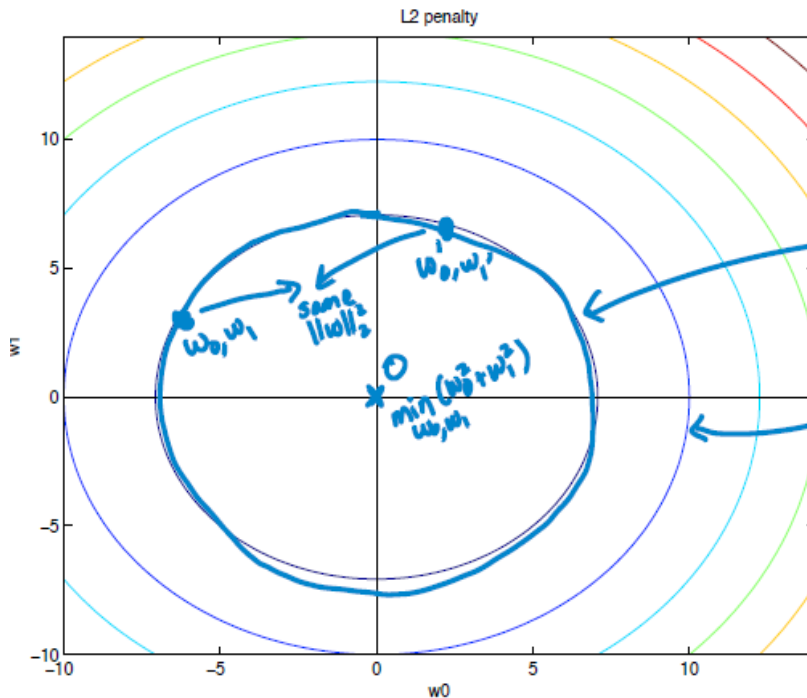
$\sum_i y_i^2 + w_0^2 z_0^2 + w_1^2 z_1^2 + \text{cross terms} = \text{constant}$
ellipse

2 features for visualization sake

$$RSS(\mathbf{w}) + \lambda \|\mathbf{w}\|_2^2 = \sum_{i=1}^N (y_i - w_0 h_0(\mathbf{x}_i) - w_1 h_1(\mathbf{x}_i))^2 + \lambda (w_0^2 + w_1^2)$$

Visualising ridge cost in 2D

169



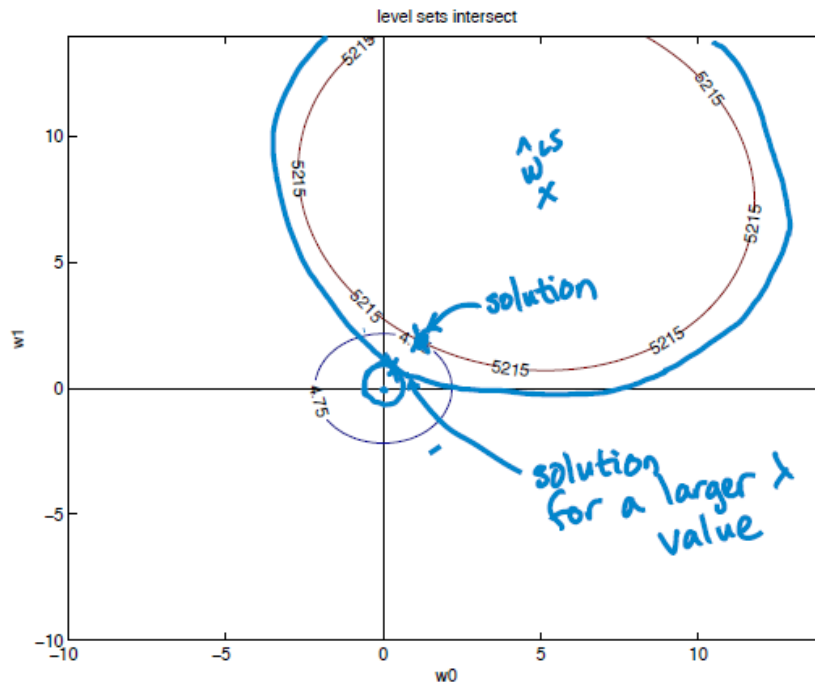
$\|w\|_2^2 = \text{const}_1$
 $\|w\|_2^2 = \text{const}_2 > \text{const}_1$
 \vdots

$w_0^2 + w_1^2 = \text{constant}$
circle

$$\text{RSS}(\mathbf{w}) + \lambda \|\mathbf{w}\|_2^2 = \sum_{i=1}^N (y_i - w_0 h_0(\mathbf{x}_i) - w_1 h_1(\mathbf{x}_i))^2 + \lambda (w_0^2 + w_1^2)$$

Visualising ridge cost in 2D

170

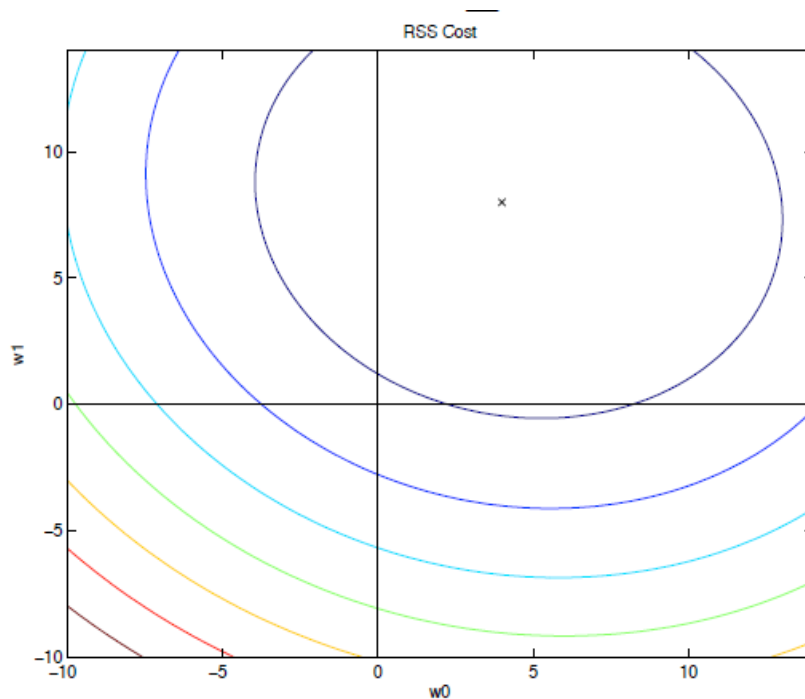


For a specific λ value,
some balance between
RSS and $\|w\|_2^2$

$$\text{RSS}(\mathbf{w}) + \lambda \|\mathbf{w}\|_2^2 = \sum_{i=1}^N (y_i - w_0 h_0(\mathbf{x}_i) - w_1 h_1(\mathbf{x}_i))^2 + \lambda (w_0^2 + w_1^2)$$

Visualising lasso cost in 2D

171

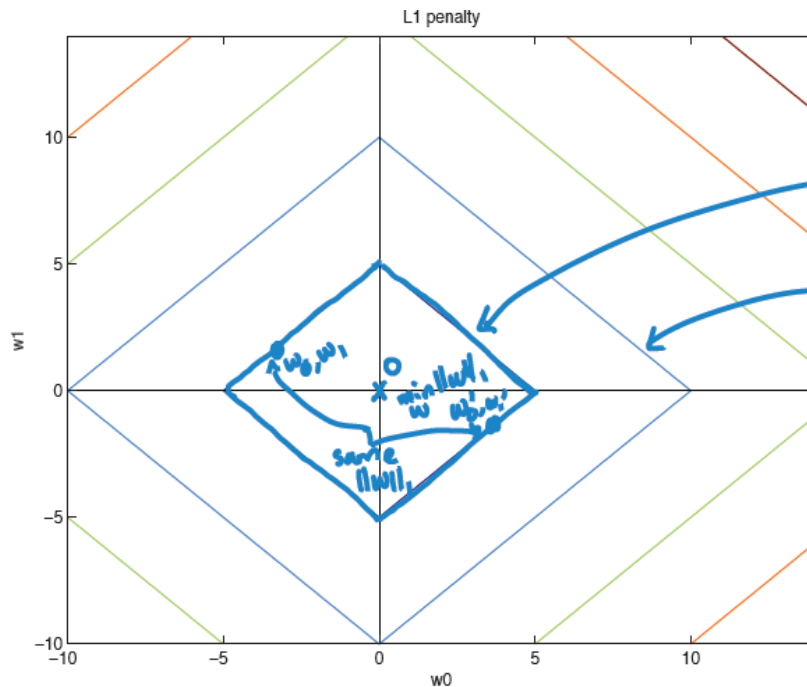


RSS contours for lasso are exactly the same as those for ridge!

$$\text{RSS}(\mathbf{w}) + \lambda \|\mathbf{w}\|_1 = \sum_{i=1}^N (y_i - w_0 h_0(\mathbf{x}_i) - w_1 h_1(\mathbf{x}_i))^2 + \lambda (|w_0| + |w_1|)$$

Visualising lasso cost in 2D

172



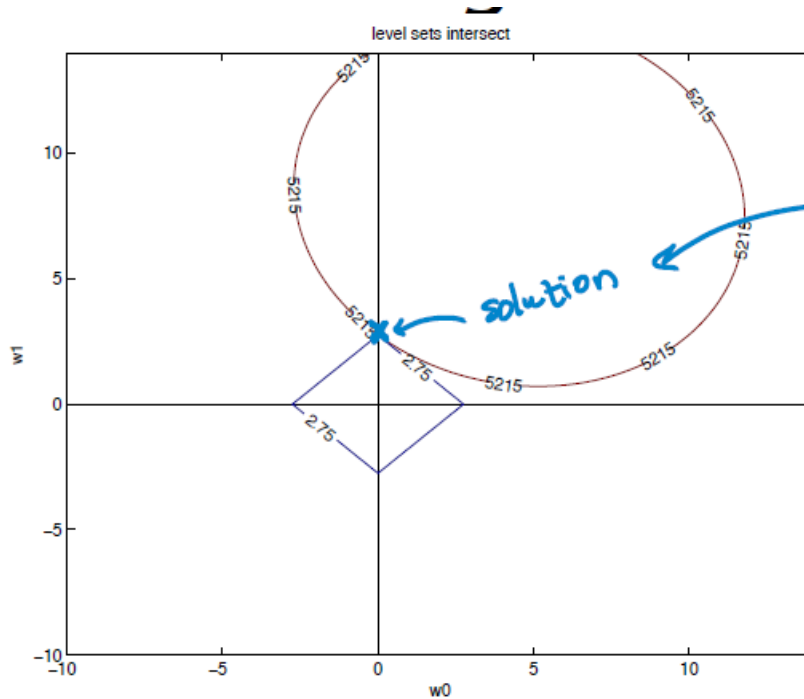
$\|w\|_1 = \text{const}_1$
 $\|w\|_1 = \text{const}_2 > \text{const}_1$
 \vdots

$|w_0| + |w_1| = \text{constant}$
diamond

$$\text{RSS}(w) + \lambda \|w\|_1 = \sum_{i=1}^N (y_i - w_0 h_0(\mathbf{x}_i) - w_1 h_1(\mathbf{x}_i))^2 + \lambda (|w_0| + |w_1|)$$

Visualising lasso cost in 2D

173



For a specific value of λ ,

We are getting sparse solution,
the $w_0 = 0$

$$\text{RSS}(\mathbf{w}) + \lambda \|\mathbf{w}\|_1 = \sum_{i=1}^N (y_i - w_0 h_0(\mathbf{x}_i) - w_1 h_1(\mathbf{x}_i))^2 + \lambda (|w_0| + |w_1|)$$

How we optimise for objective

174

To solve for $\hat{\mathbf{w}}$, previously took gradient of total cost objective and either:

- 1) Derived closed-form solution
- 2) Used in gradient descent algorithm

Optimise for lasso objective

175

Lasso total cost: $RSS(\mathbf{w}) + \lambda \|\mathbf{w}\|_1$

$\sum_{j=0}^p |w_j|$

Issues:

- 1) What's the derivative of $|w_j|$?



gradients \rightarrow subgradients

- 2) Even if we could compute derivative, no closed-form solution

can use subgradient descent

Coordinate descent

176

Goal: Minimize some function g

$$\min_w g(w)$$

$$g(w) = g(w_0, w_1, \dots, w_D)$$

when keeping others fixed

Often, hard to find minimum for all coordinates, but **easy for each coordinate**

Coordinate descent:

Initialize $\hat{w} = 0$ (or smartly...)

while not converged

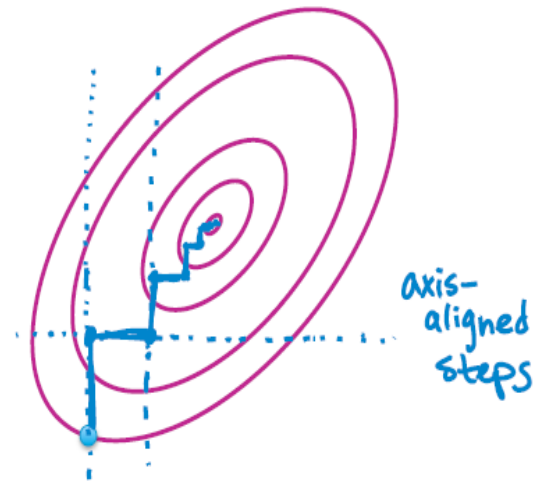
pick a coordinate j

$$\hat{w}_j \leftarrow$$

values from previous iterations

$$\min_w g(\hat{w}_0, \dots, \hat{w}_{j-1}, w, \hat{w}_{j+1}, \dots, \hat{w}_D)$$

just min over j th coordinate



Comments on coordinate descent

177

How do we pick next coordinate?

- At random ("random" or "stochastic" coordinate descent), round robin, ...

No stepsize to choose!

Super useful approach for *many* problems

- Converges to optimum in some cases (e.g., "strongly convex")
- Converges for lasso objective

Normalizing features

178

Normalizing features

Scale training columns (not rows!) as:

$$\underline{h}_j(\mathbf{x}_k) = \frac{h_j(\mathbf{x}_k)}{\sqrt{\sum_{i=1}^N h_j(\mathbf{x}_i)^2}}$$

Normalizer: z_j

Apply same training scale factors to test data:

$$\underline{h}_j(\mathbf{x}_k) = \frac{h_j(\mathbf{x}_k)}{\sqrt{\sum_{i=1}^N h_j(\mathbf{x}_i)^2}}$$

apply to test point

Normalizer: z_j

summing over training points



Optimising least squares objective

179

One coordinate at a time

$$\text{RSS}(\mathbf{w}) = \sum_{i=1}^N \left(y_i - \sum_{j=0}^D w_j \underline{h}_j(\mathbf{x}_i) \right)^2$$

normalized features

Fix all coordinates w_{-j} and take partial w.r.t. w_j

all w_k for $k \neq j$

1d optimization coordinate by coordinate

$$\frac{\partial}{\partial w_j} \text{RSS}(\mathbf{w}) = -2 \sum_{i=1}^N \underline{h}_j(\mathbf{x}_i) \left(y_i - \sum_{j=0}^D w_j \underline{h}_j(\mathbf{x}_i) \right)$$

$$= -2 \sum_{i=1}^N h_j(x_i) \left(y_i - \sum_{k \neq j} w_k h_k(x_i) - w_j h_j(x_i) \right)$$

$$= -2 \sum_{i=1}^N h_j(x_i) \left(y_i - \sum_{k \neq j} w_k h_k(x_i) \right) + 2 w_j \sum_{i=1}^N h_j(x_i)^2$$

by definition of normalized features, = 1

$$= -2 \rho_j + 2 w_j$$

Optimising least squares objective

180

$$\text{RSS}(\mathbf{w}) = \sum_{i=1}^N \left(y_i - \sum_{j=0}^D w_j h_j(\mathbf{x}_i) \right)^2$$

Set partial = 0 and solve

$$\frac{\partial}{\partial w_j} \text{RSS}(\mathbf{w}) = -2\rho_j + 2w_j = 0$$
$$\hat{w}_j = \rho_j$$

Coordinate descent for least squares regression

181

Initialize $\hat{\mathbf{w}} = 0$ (or smartly...)
while not converged
for $j=0,1,\dots,D$

compute:
$$\rho_j = \sum_{i=1}^N \underline{h}_j(\mathbf{x}_i) \overbrace{(y_i - \hat{y}_i(\hat{\mathbf{w}}_{-j}))}^{\text{residual without feature } j}$$

set: $\hat{\mathbf{w}}_j = \rho_j$

prediction
without feature j

Measure of the correlation between w_j
and the residual without this feature.

How to access convergence

182

Initialize $\hat{\mathbf{w}} = 0$ (or smartly...)

while not converged

for $j=0,1,\dots,D$

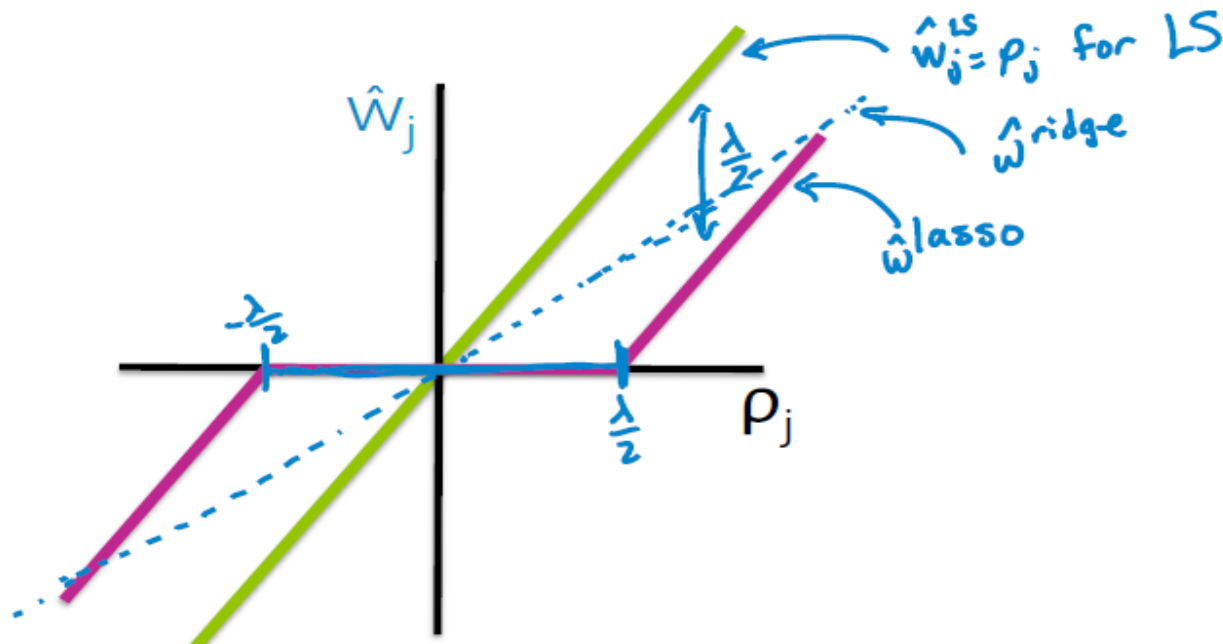
compute: $\rho_j = \sum_{i=1}^N \underline{h}_j(\mathbf{x}_i)(y_i - \hat{y}_i(\hat{\mathbf{w}}_{-j}))$

set: $\hat{w}_j = \begin{cases} \rho_j + \lambda/2 & \text{if } \rho_j < -\lambda/2 \\ 0 & \text{if } \rho_j \text{ in } [-\lambda/2, \lambda/2] \\ \rho_j - \lambda/2 & \text{if } \rho_j > \lambda/2 \end{cases}$

Soft thresholding

183

$$\hat{w}_j = \begin{cases} \rho_j + \lambda/2 & \text{if } \rho_j < -\lambda/2 \\ 0 & \text{if } \rho_j \text{ in } [-\lambda/2, \lambda/2] \\ \rho_j - \lambda/2 & \text{if } \rho_j > \lambda/2 \end{cases}$$



Convergence criteria

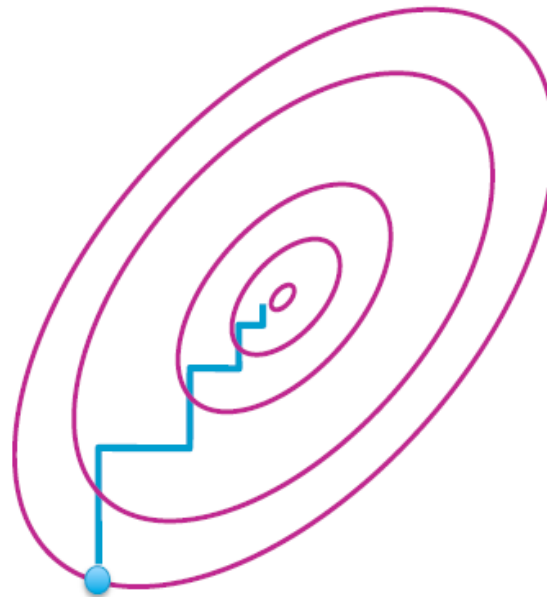
184

When to stop?

For convex problems, will start to take **smaller and smaller steps**

Measure size of steps taken in a full loop over all features

– stop when **max step** $< \epsilon$



Other lasso solvers

185

Classically: Least angle regression (LARS) [Efron et al. '04]

Then: Coordinate descent algorithm

[Fu '98, Friedman, Hastie, & Tibshirani '08]

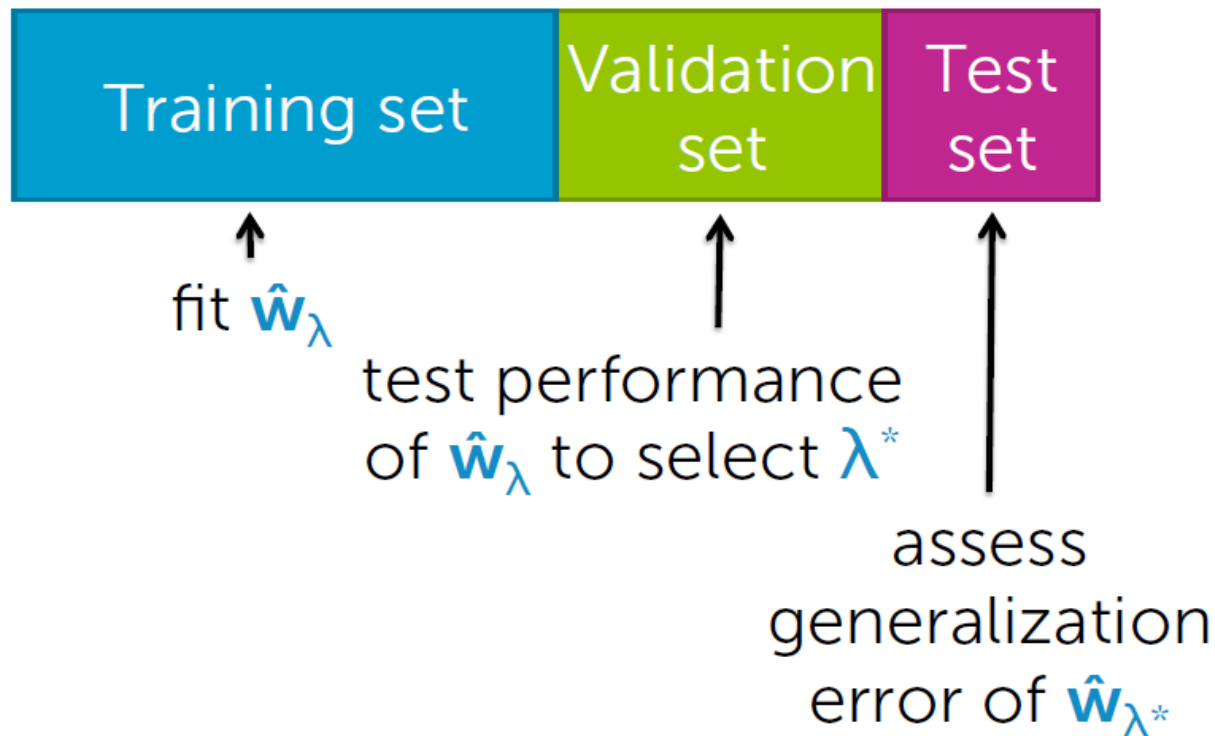
Now:

- Parallel CD (e.g., Shotgun, [Bradley et al. '11])
- Other parallel learning approaches for linear models
 - Parallel stochastic gradient descent (SGD) (e.g., Hogwild! [Niu et al. '11])
 - Parallel independent solutions then averaging [Zhang et al. '12]
- Alternating directions method of multipliers (ADMM) [Boyd et al. '11]

How do we choose λ

186

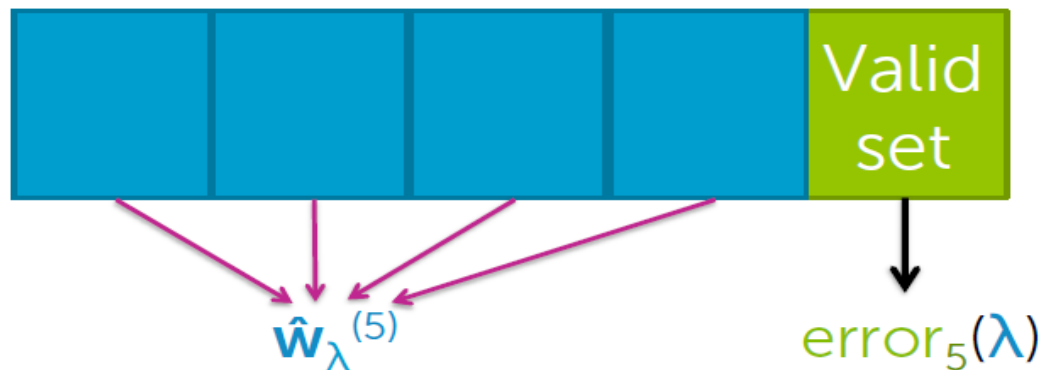
If sufficient amount of data...



How do we choose λ

187

K-fold cross validation



For $k=1, \dots, K$

1. Estimate $\hat{w}_\lambda^{(k)}$ on the training blocks
2. Compute error on validation block: $error_k(\lambda)$

Compute average error: $CV(\lambda) = \frac{1}{K} \sum_{k=1}^K error_k(\lambda)$

How do we chose λ

188

Choosing λ via cross validation

Cross validation is choosing the λ that provides best predictive accuracy

Tends to favor less sparse solutions, and thus smaller λ , than optimal choice for feature selection

c.f., "Machine Learning: A Probabilistic Perspective",
Murphy, 2012 for further discussion

Impact of feature selection and lasso

189

Lasso has changed machine learning, statistics, & electrical engineering

But, for feature selection in general, be careful about interpreting selected features

- selection only considers features included
- sensitive to correlations between features
- result depends on algorithm used
- there are theoretical guarantees for lasso under certain conditions

What you can do now

190

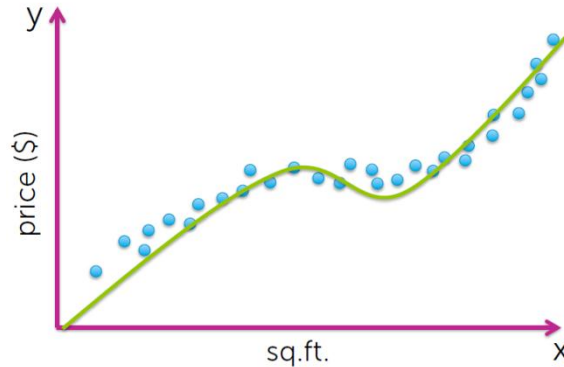
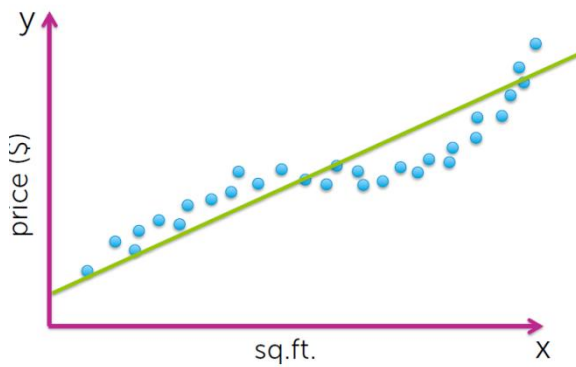
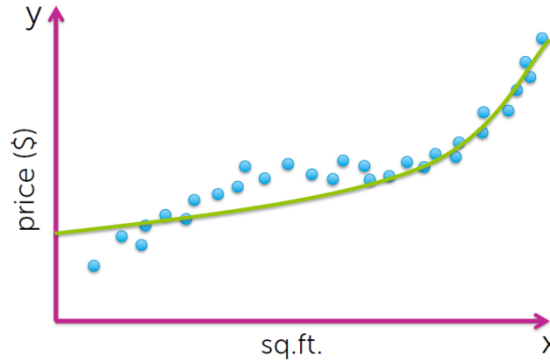
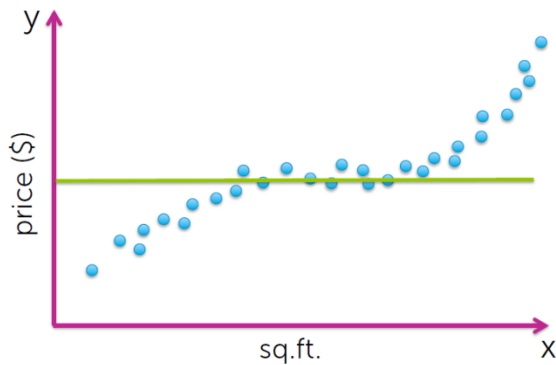
- Perform feature selection using “all subsets” and “forward stepwise” algorithms
- Analyze computational costs of these algorithms
- Contrast greedy and optimal algorithms
- Formulate lasso objective
- Describe what happens to estimated lasso coefficients as tuning parameter λ is varied
- Interpret lasso coefficient path plot
- Contrast ridge and lasso regression
- Describe geometrically why L1 penalty leads to sparsity
- Estimate lasso regression parameters using an iterative coordinate descent algorithm
- Implement K-fold cross validation to select lasso tuning parameter λ

NONPARAMETRIC REGRESSION

Fit globaly vs fit locally

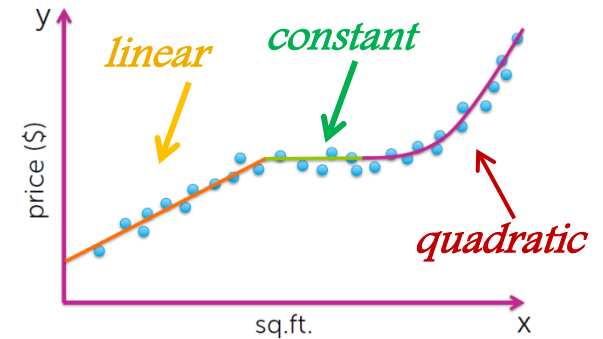
192

Parametric models



Below ...

$f(x)$ is not really a polynomial function



What alternative do we have?

193

If we:

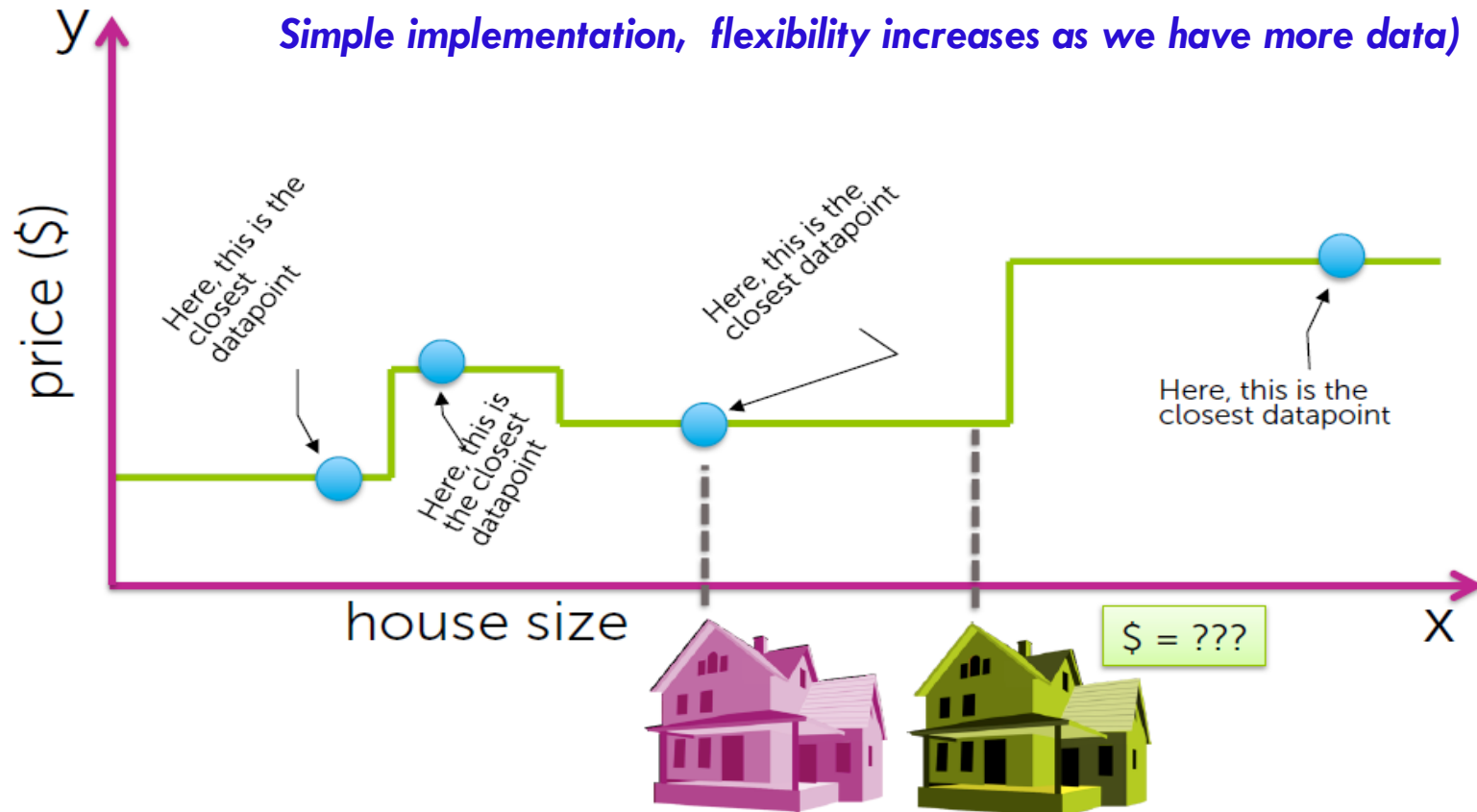
- Want to allow flexibility in $f(\mathbf{x})$ having **local structure**
- Don't want to infer "structural breaks"

What's a simple option we have?

- Assuming we have **plenty of data...**

Nearest Neighbor & Kernel Regression (nonparametric approach)

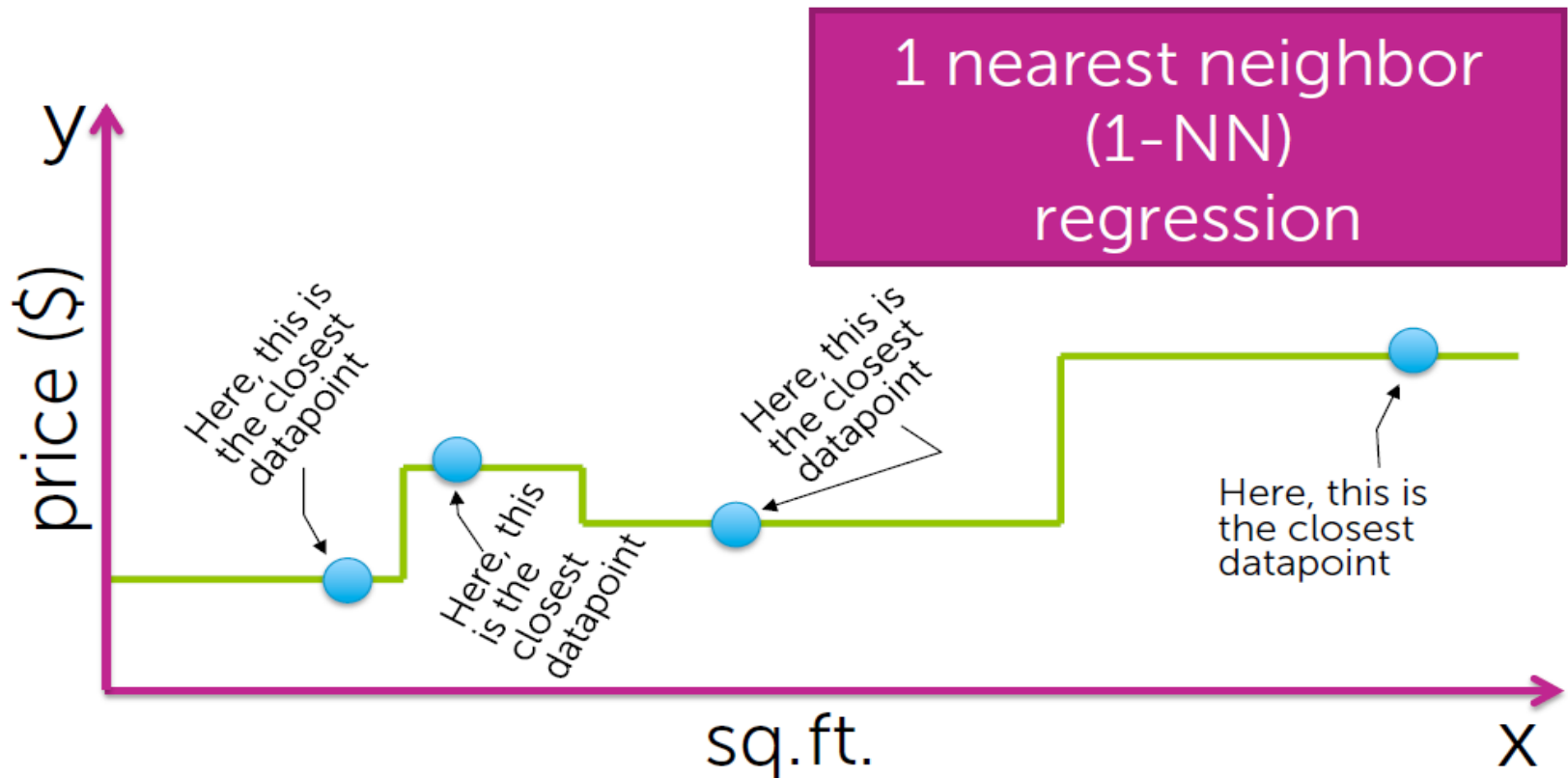
194



Fit locally to each data point

195

Predicted value = "closest" y_i



What people do naturally...

196

Real estate agent assesses value by finding sale of most similar house



\$ = ???



\$ = 850k

1-NN regression more formally

197

Dataset of (🏠, \$) pairs: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)$

Query point: \mathbf{x}_q ← 🏠 \$?
big lime green house

1. Find "closest" \mathbf{x}_i in dataset

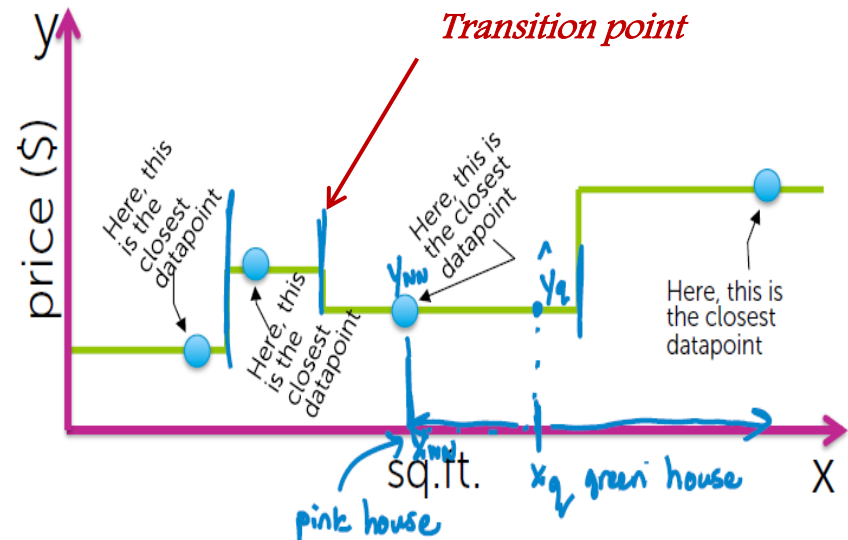
$$x_{NN} \leftarrow \min_i \text{distance}(x_i, x_q)$$

big pink house

2. Predict

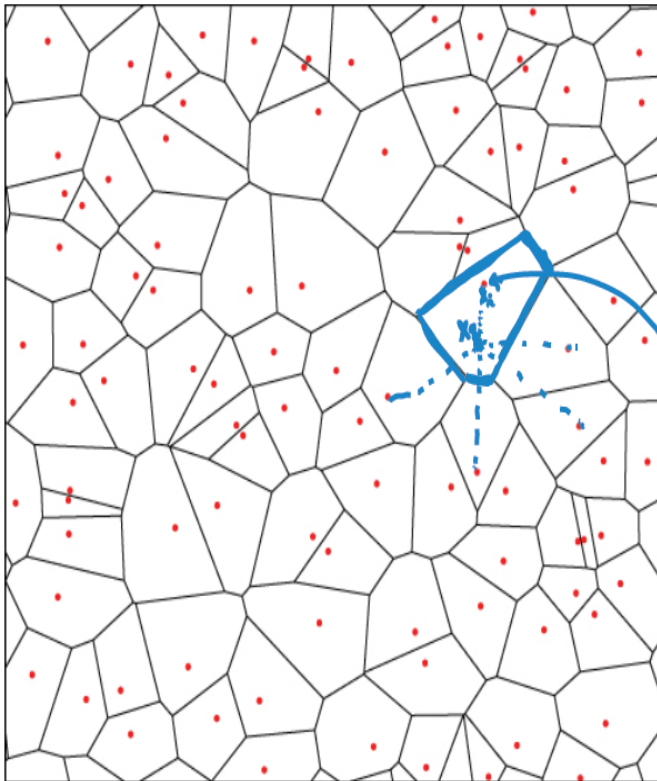
$$\hat{y}_q = y_{NN}$$

sales price of big pink house



Visualizing 1-NN in multiple dimensions

198



Voronoi tessellation (or diagram):

- Divide space into N regions, each containing 1 datapoint
- Defined such that any x in region is "closest" to region's datapoint

x_j closer to x_i than any other x_j for $j \neq i$.

Don't explicitly form!

Distance metrics: Notion of „closest”

199

In 1D, just Euclidean distance:

$$\text{distance}(x_j, x_q) = |x_j - x_q|$$

In multiple dimensions:

- can define many interesting distance functions
- most straightforwardly, might want to weight different dimensions differently

Weighting housing inputs

200

Some inputs are more relevant than others



bedrooms
bathrooms
sq.ft. living
sq.ft. lot
floors
year built
year renovated
waterfront



15/10, 22/10, 29/10 2019

Scaled Euclidean distance

201

Formally, this is achieved via

distance($\mathbf{x}_j, \mathbf{x}_q$) =

$$\sqrt{a_1(\mathbf{x}_j[1] - \mathbf{x}_q[1])^2 + \dots + a_d(\mathbf{x}_j[d] - \mathbf{x}_q[d])^2}$$

weight on each input
(defining relative importance)

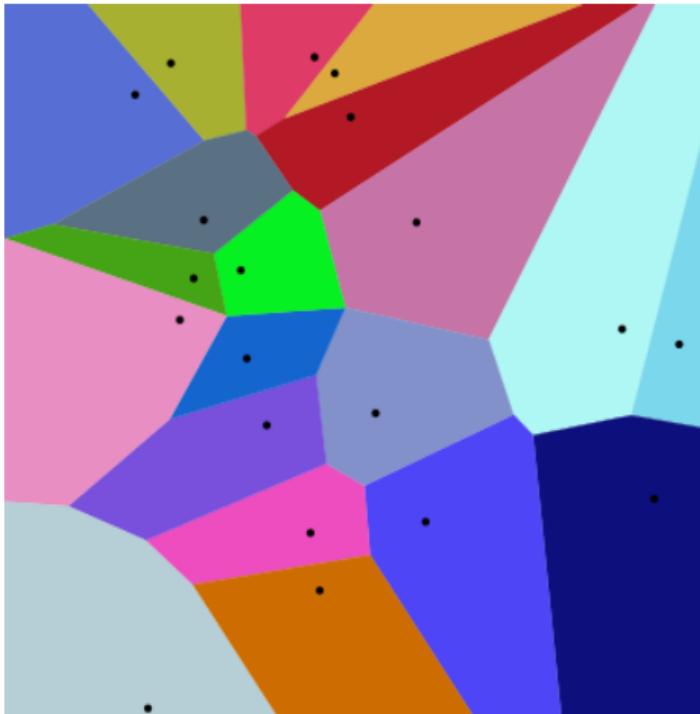
Other example distance metrics:

- Mahalanobis, rank-based, correlation-based, cosine similarity, Manhattan, Hamming, ...

Different distance metrics

202

Euclidean distance





Manhattan distance



Performing 1-NN search


203

- Query house: 
- Dataset: 
- **Specify:** Distance metric
- **Output:** Most similar house




1-NN algorithm



204

Initialize **Dist2NN** = ∞ ,  = \emptyset ← closest house



For $i=1,2,\dots,N$

 Compute: $\delta = \text{distance}(\text{house}_i, \text{house}_q)$ ← query house 

 If $\delta < \text{Dist2NN}$

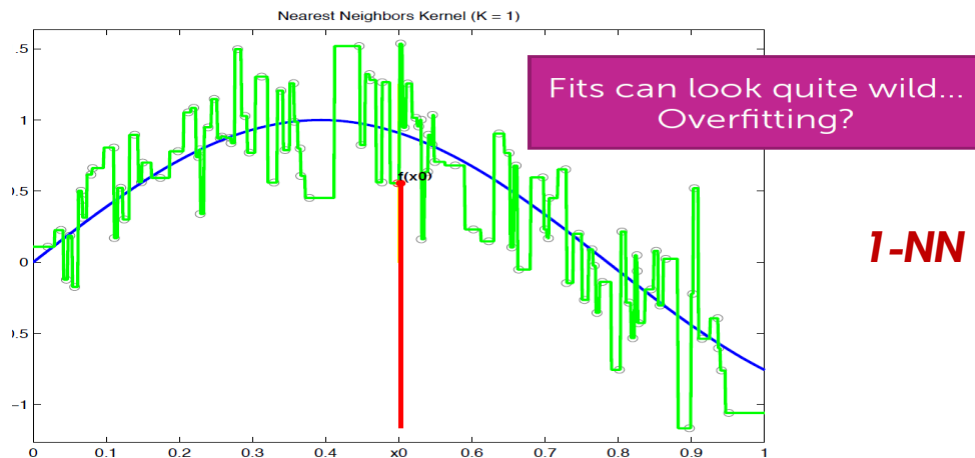
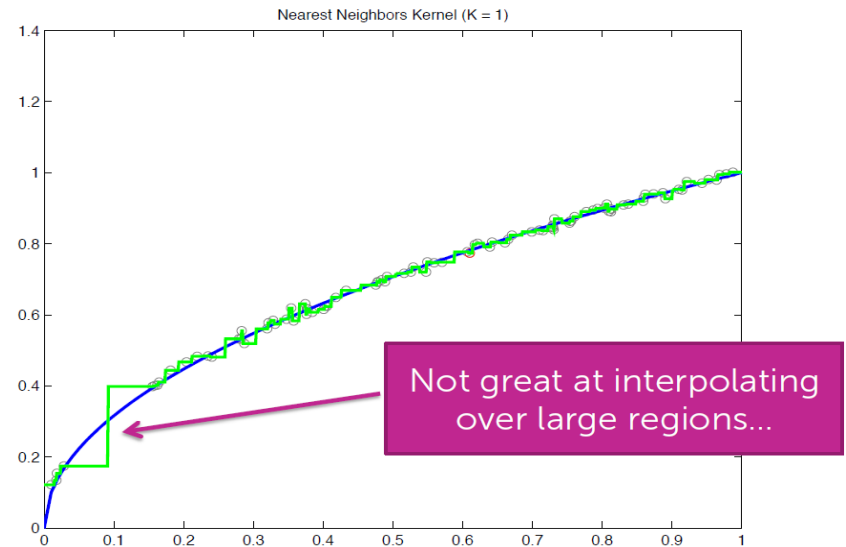
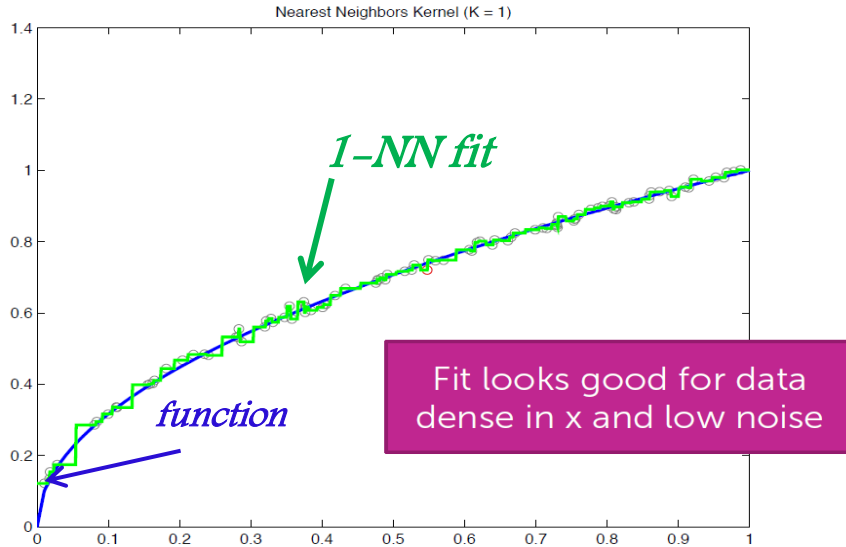
 set   _{i}

 set **Dist2NN** = δ

Return **most similar house**  ← closest house to query house 

1-NN in practice

205

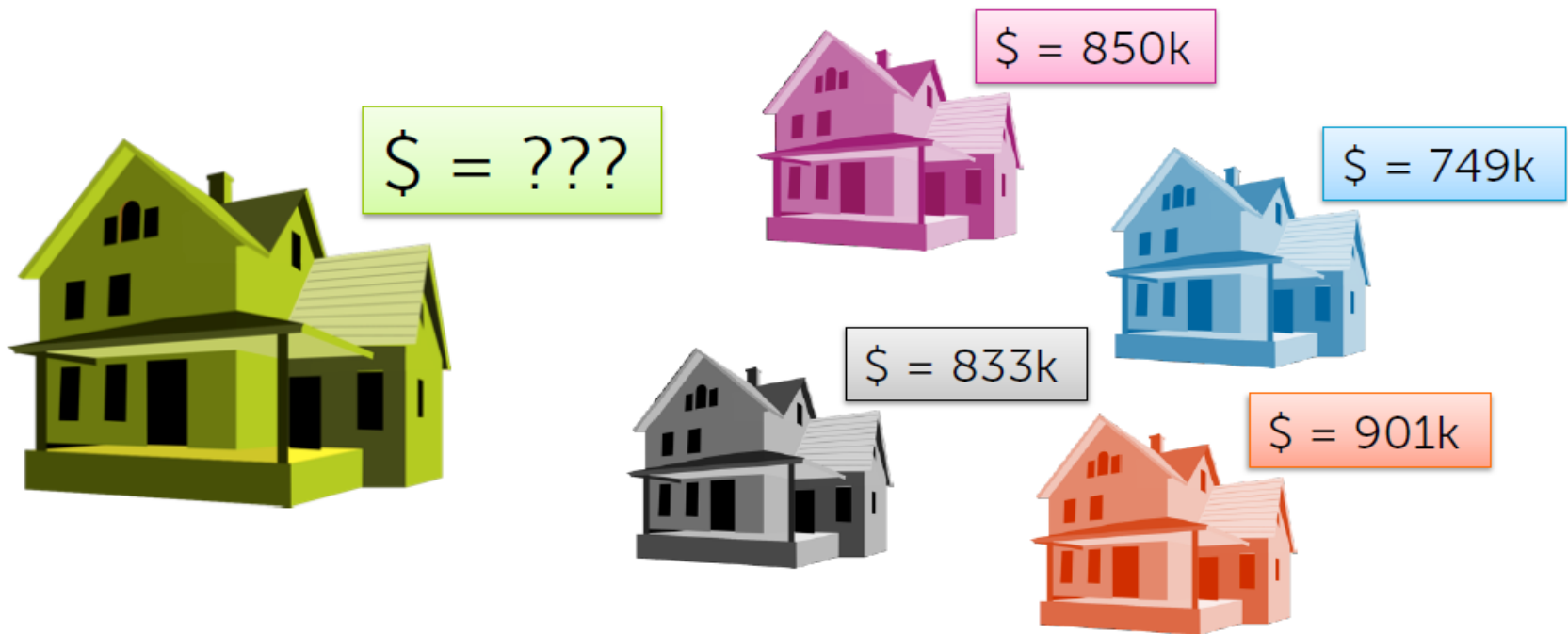


1-NN sensitive to noise in the data

Get more „comps”

206

More reliable estimate if you base estimate off of a larger set of comparable homes



K-NN regression more formally

207

Dataset of (🏠, \$) pairs: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)$

Query point: \mathbf{x}_q

1. Find k closest \mathbf{x}_i in dataset



$(x_{NN1}, x_{NN2}, \dots, x_{NNk})$ such that for any x_i not in nearest neighbor set,
 $\text{distance}(x_i, x_q) \geq \text{distance}(x_{NNk}, x_q)$

2. Predict

$$\begin{aligned}\hat{y}_q &= \frac{1}{k} (y_{NN1} + y_{NN2} + \dots + y_{NNk}) \\ &= \frac{1}{k} \sum_{j=1}^k y_{NNj}\end{aligned}$$

K-NN more formally

208

- Query house: 
- Dataset: 
- **Specify:** Distance metric
- **Output:** Most similar houses



K-NN algorithm

209

Initialize **Dist2kNN** = $\text{sort}(\delta_1, \dots, \delta_k)$ ← list of sorted distances
= $\text{sort}(\text{house}_1, \dots, \text{house}_k)$ ← list of sorted houses

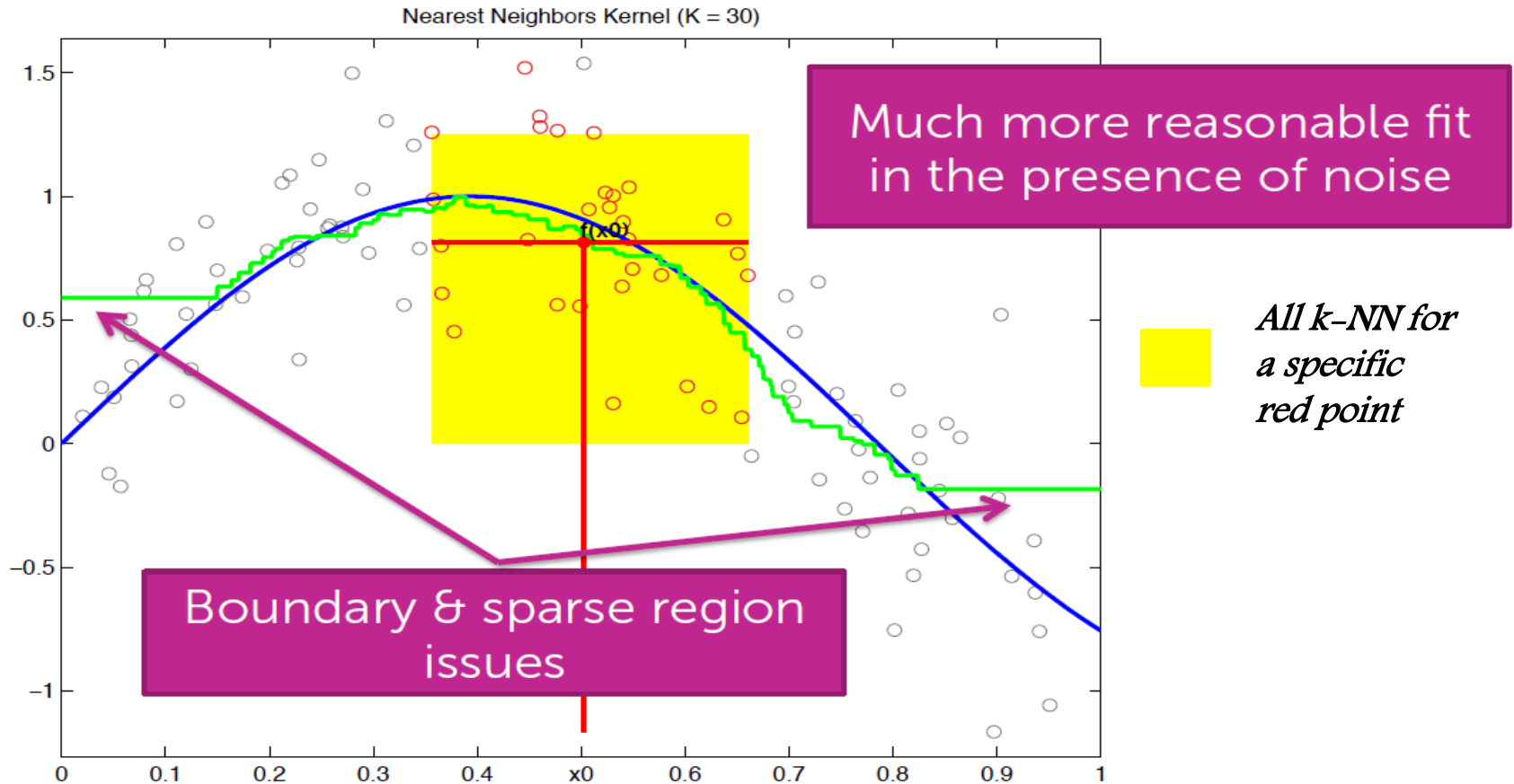
For $i=k+1, \dots, N$
Compute: $\delta = \text{distance}(\text{house}_i, \text{house}_q)$ ← query house

If $\delta < \text{Dist2kNN}[k]$
find j such that $\delta > \text{Dist2kNN}[j-1]$ but $\delta < \text{Dist2kNN}[j]$
remove furthest house and shift queue:
 $[j+1:k] = [j:k]$
Dist2kNN $[j+1:k] = \text{Dist2kNN}[j:k-1]$
set **Dist2kNN** $[j] = \delta$ and $\text{house}_j = \text{house}_i$

Return **k most similar houses** ← closest houses to query house

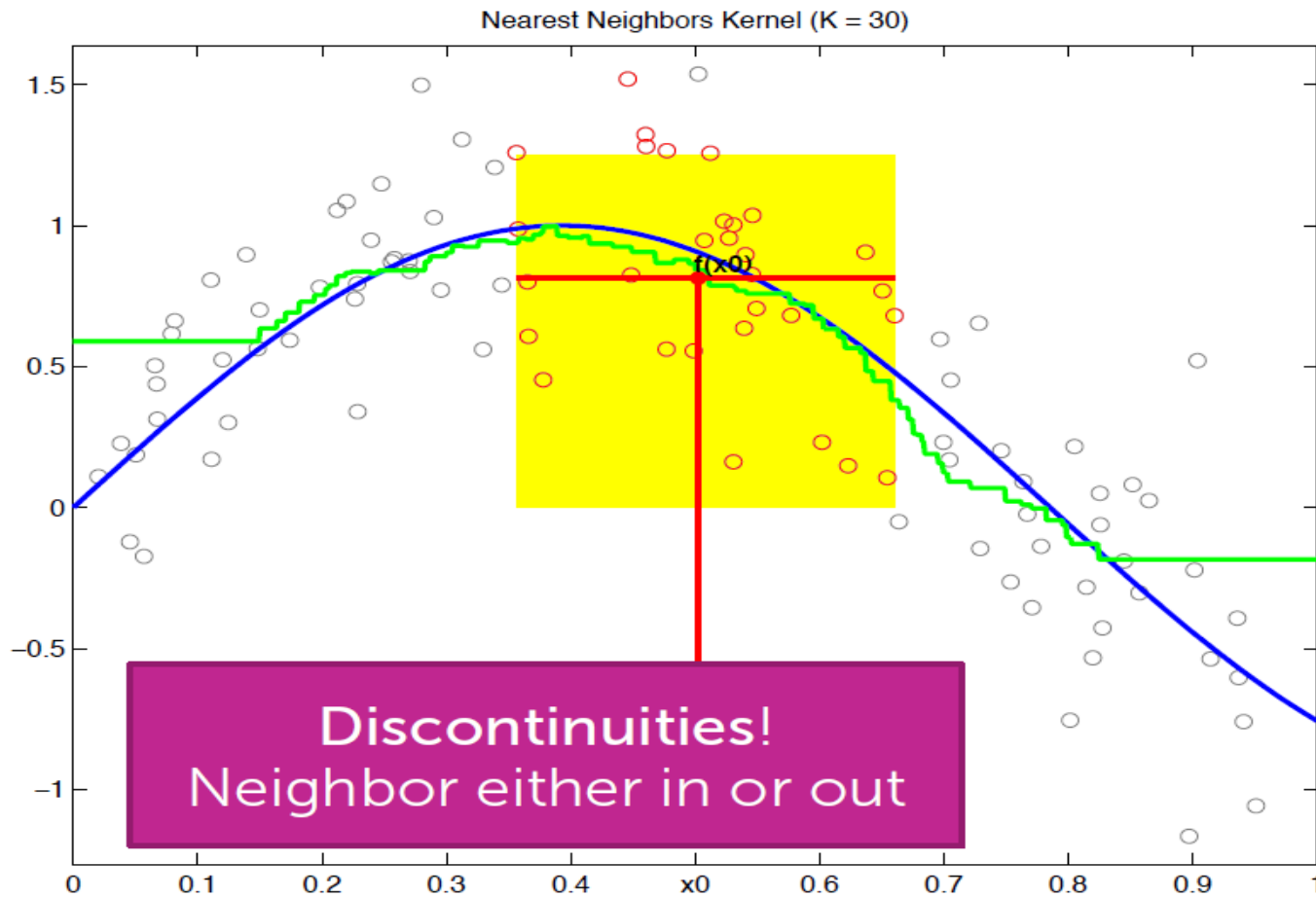
K-NN in practice

210



K-NN in practice

211



Issues with discontinuities

212

Overall predictive accuracy might be okay, but...

For example, in housing application:

- If you are a buyer or seller, this matters
- Can be a jump in estimated value of house going just from 2640 sq.ft. to 2641 sq.ft.
- Don't really believe this type of fit

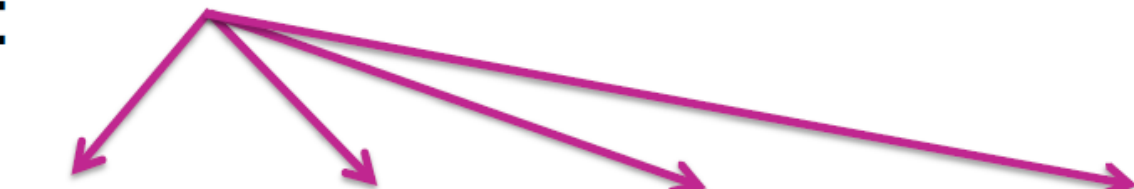
Weighted k-NN

213

Weigh more similar houses more than those less similar in list of k-NN

Predict:

weights on NN


$$\hat{y}_q = \frac{C_{qNN1}y_{NN1} + C_{qNN2}y_{NN2} + C_{qNN3}y_{NN3} + \dots + C_{qNNk}y_{NNk}}{\sum_{j=1}^k C_{qNNj}}$$

How to define weights

214

Want weight c_{qNNj} to be small when
distance($\mathbf{x}_{NNj}, \mathbf{x}_q$) large

and c_{qNNj} to be large when
distance($\mathbf{x}_{NNj}, \mathbf{x}_q$) small

Simple method:

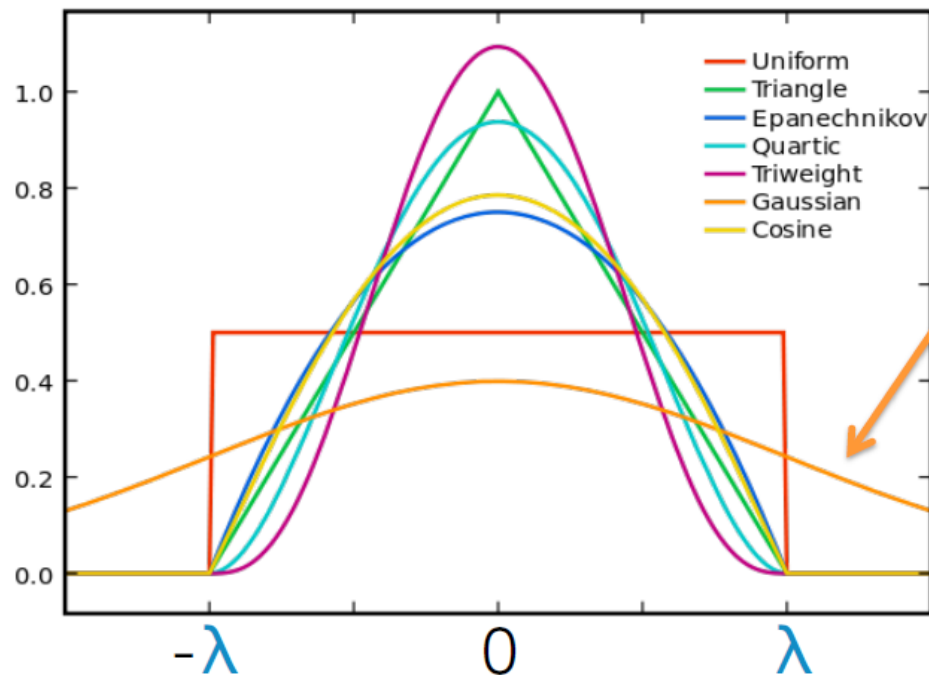
$$c_{qNNj} = \frac{1}{\text{distance}(x_j, x_q)}$$

Kernel weights for $d=1$

215

Define: $c_{qNNj} = \text{Kernel}_\lambda(|x_{NNj} - x_q|)$

simple isotropic case



Gaussian kernel:
 $\text{Kernel}_\lambda(|x_i - x_q|) = \exp(-(x_i - x_q)^2 / \lambda)$

Note: never exactly 0!

Kernel drives how the weights will decay, if at all, as a function of the distance.

Kernel regression

Nadaraya-Watson
kernel weighted average

216

Instead of just weighting NN, weight *all* points

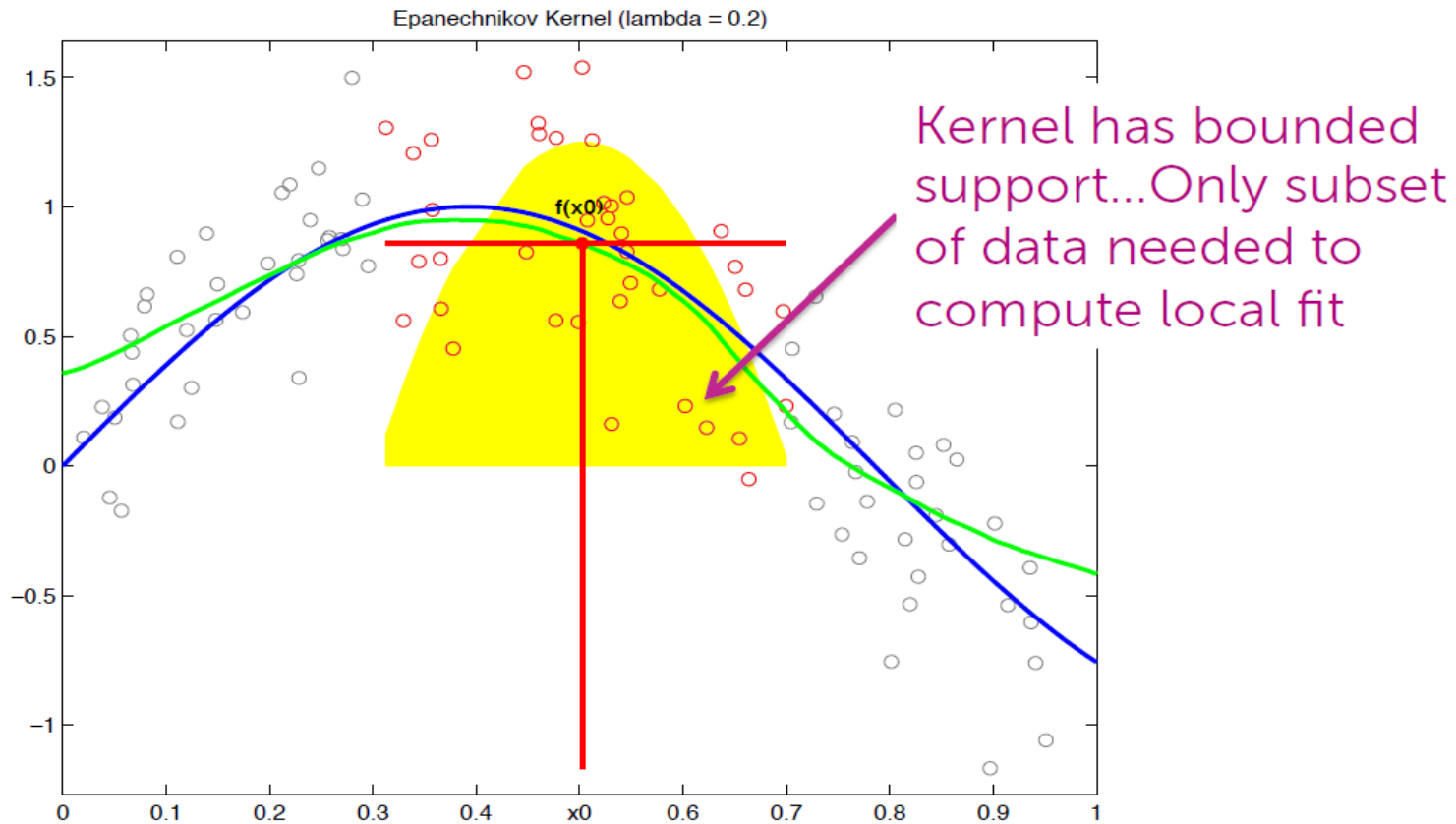
Predict:

weight on each datapoint

$$\hat{y}_q = \frac{\sum_{i=1}^N c_{qi} y_i}{\sum_{i=1}^N c_{qi}} = \frac{\sum_{i=1}^N \text{Kernel}_\lambda(\text{distance}(\mathbf{x}_i, \mathbf{x}_q)) * y_i}{\sum_{i=1}^N \text{Kernel}_\lambda(\text{distance}(\mathbf{x}_i, \mathbf{x}_q))}$$

Kernel regression in practice

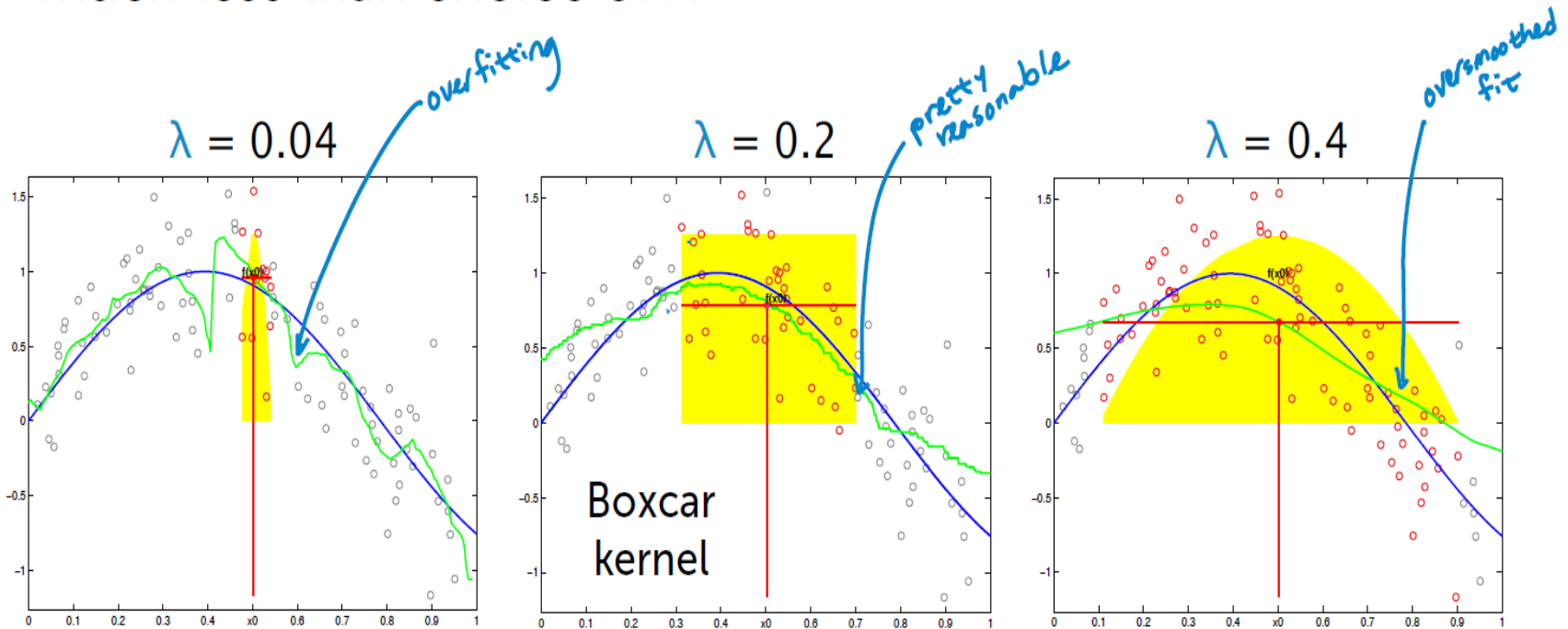
217



Choice of bandwidth λ

218

Often, choice of kernel matters much less than choice of λ



Choosing λ (or k on k -NN)

219

How to choose? Same story as always...

Cross Validation

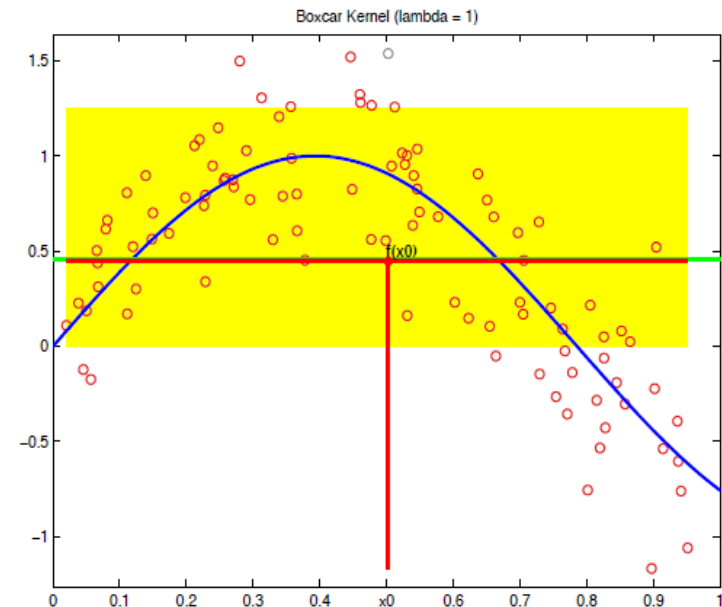
Contrasting with global average

220

A globally constant fit weights all points equally

$$\hat{y}_q = \frac{1}{N} \sum_{i=1}^N y_i = \frac{\sum_{i=1}^N c y_i}{\sum_{i=1}^N c}$$

equal weight on each datapoint



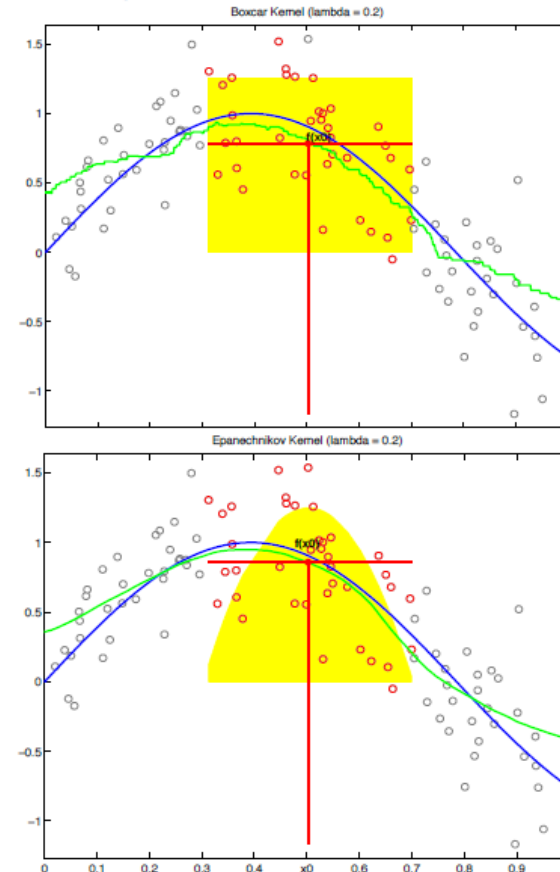
Contrasting with global average

221

Kernel regression leads to **locally constant fit**

- slowly add in some points and
and let others gradually die off

$$\hat{y}_q = \frac{\sum_{i=1}^N \text{Kernel}_\lambda(\text{distance}(\mathbf{x}_i, \mathbf{x}_q)) * y_i}{\sum_{i=1}^N \text{Kernel}_\lambda(\text{distance}(\mathbf{x}_i, \mathbf{x}_q))}$$



Local linear regression

222

So far, discussed fitting **constant function locally** at each point

→ “locally weighted averages”

Can instead fit a **line or polynomial locally** at each point

→ “locally weighted linear regression”

Local regression rules of thumb

223

- Local linear fit reduces bias at boundaries with minimum increase in variance
- Local quadratic fit doesn't help at boundaries and increases variance, but does help capture curvature in the interior
- With sufficient data, local polynomials of odd degree dominate those of even degree

Recommended default choice:
local linear regression

Nonparametric approaches

224

k-NN and kernel regression are examples of **nonparametric** regression

General goals of nonparametrics:

- Flexibility
- Make few assumptions about $f(\mathbf{x})$
- **Complexity can grow with the number of observations N**

Lots of other choices:

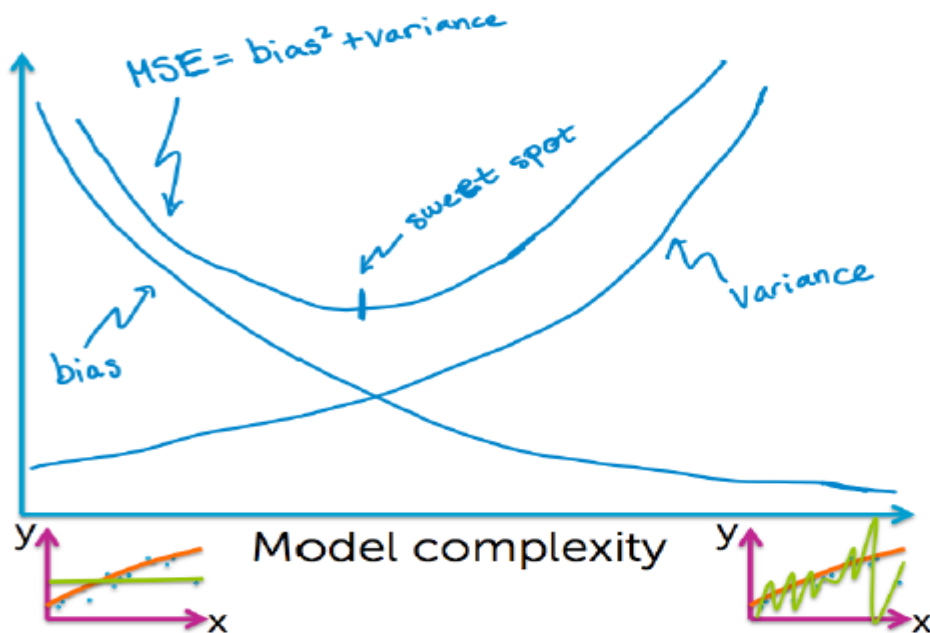
- Splines, trees, locally weighted structured regression models...

Limiting behaviour of NN

225

Noiseless setting ($\varepsilon_i=0$)

In the limit of getting an infinite amount of noiseless data, the **MSE of 1-NN fit goes to 0**

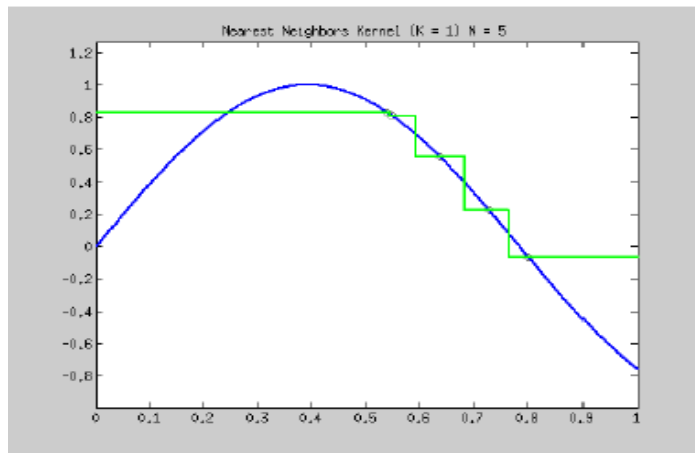


Limiting behaviour of NN

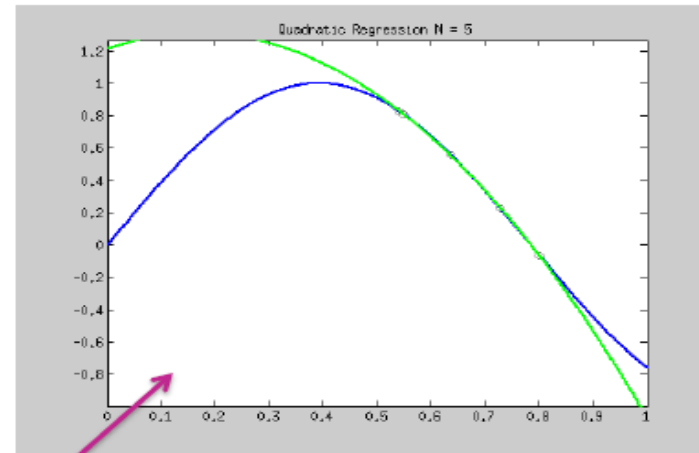
226

Noiseless setting ($\epsilon_i=0$)

In the limit of getting an infinite amount of noiseless data, the **MSE of 1-NN fit goes to 0**



1-NN fit

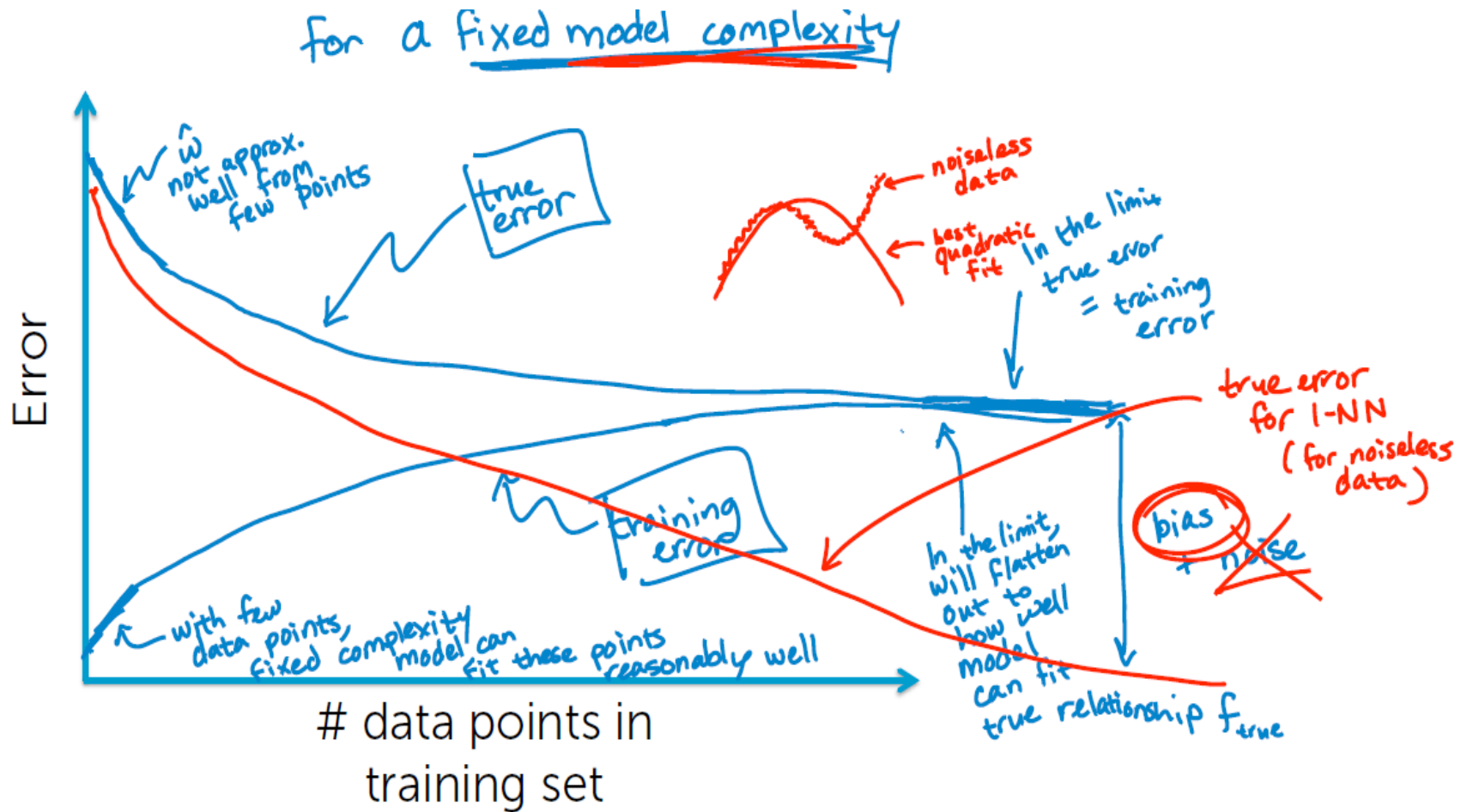


Quadratic fit

Not true for parametric models!

Error vs amount of data

227

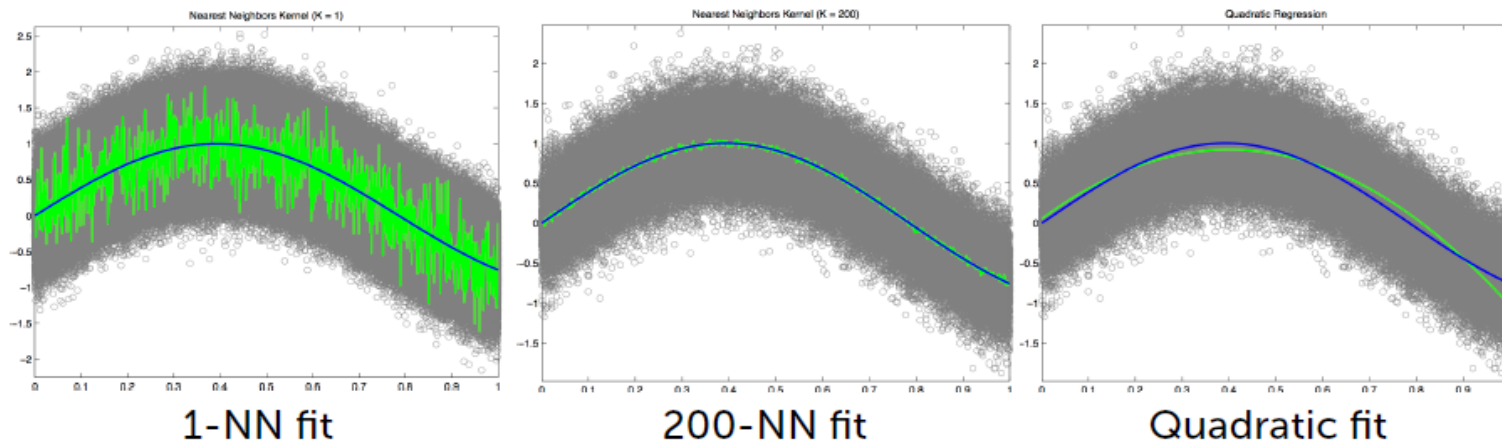


Limiting behaviour of NN

228

Noisy data setting

In the limit of getting an infinite amount of data, the **MSE of NN fit goes to 0** if **k grows, too**



Issues: NN and kernel methods

229

NN and kernel methods work well when the data cover the space, but...

- the more dimensions d you have, the more points N you need to cover the space
- need $N = O(\exp(d))$ data points for good performance

This is where parametric models become useful...

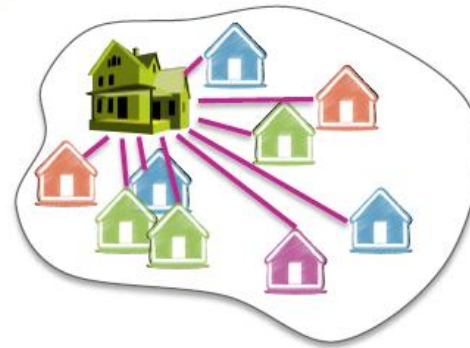
Issues: Complexity of NN search

230

Naïve approach: Brute force search

- Given a query point \mathbf{x}_q
- Scan through each point $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$
- $O(N)$ distance computations per 1-NN query!
- $O(N \log k)$ per k-NN query!

What if N is huge???
(and many queries)



Will talk more about efficient methods in
Clustering & Retrieval course

We discussed how to

231

- Motivate the use of nearest neighbor (NN) regression
- Define distance metrics in 1D and multiple dimensions
- Perform NN and k-NN regression
- Analyze computational costs of these algorithms
- Discuss sensitivity of NN to lack of data, dimensionality, and noise
- Perform weighted k-NN and define weights using a kernel
- Define and implement kernel regression
- Describe the effect of varying the kernel bandwidth λ or # of nearest neighbors k
- Select λ or k using cross validation
- Compare and contrast kernel regression with a global average fit
- Define what makes an approach nonparametric and why NN and kernel regression are considered nonparametric methods
- Analyze the limiting behavior of NN regression

Summarising

232

Models

- Linear regression
- Regularization: Ridge (L2), Lasso (L1)
- Nearest neighbor and kernel regression

Algorithms

- Gradient descent
- Coordinate descent

Concepts

- Loss functions, bias-variance tradeoff, cross-validation, sparsity, overfitting, model selection, feature selection