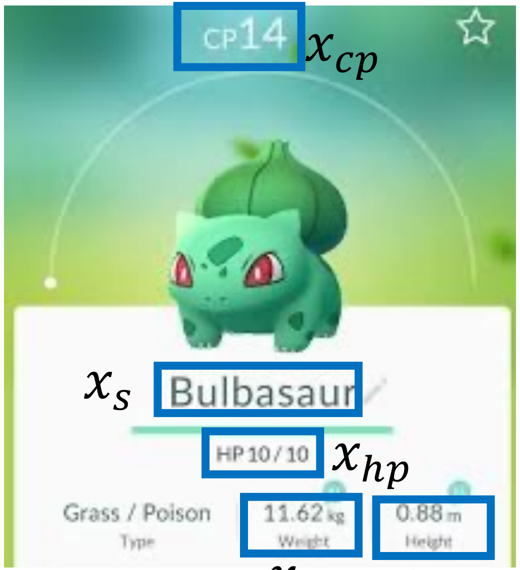
Underfitting and Overfitting

Example Application

Estimating the Combat Power (CP) of a Pokémon after evolution.

$f($



x_{cp} x_s x_{hp} x_w x_h

$) =$ CP after evolution

x y

Step One: Model

w and b are parameters (can be any value)

$y = b + w \cdot x_{cp}$

A set of function

Model

$f_1, f_2 \cdots$

f_1: $y = 10.0 + 9.0 \cdot x_{cp}$

f_2: $y = 9.8 + 9.2 \cdot x_{cp}$

f_3: $y = -0.8 - 1.2 \cdot x_{cp}$

…… infinite

$f($ 

$x) =$ CP after evolution y

Linear model: $y = b + \sum w_i x_i$

x_i: x_{cp}, x_{hp}, x_w, x_h …

feature

w_i: weight, b: bias

Step Two: Goodness of Function: Part I

$y = b + w \cdot x_{cp}$

A set of function

Model

$f_1, f_2 \cdots$

Training Data

function input:

function output (scalar):

Step Two: Goodness of Function: Part II

Training Data:
10 Pokémon's

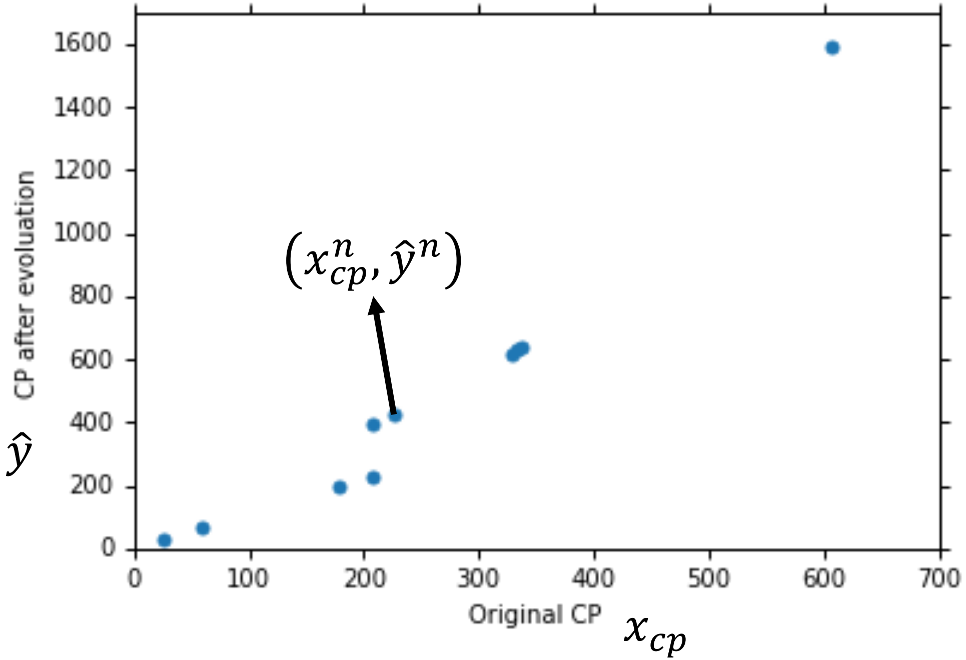
$(x^1, \hat{y}^1)$

$(x^2, \hat{y}^2)$

$\vdots$

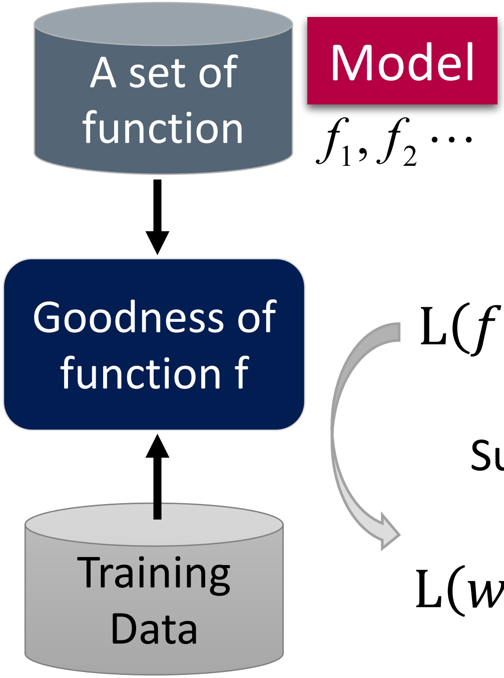
$(x^{10}, \hat{y}^{10})$

This is real data.



Step Two: Goodness of Function: Part III

$y = b + w \cdot x_{cp}$



Loss function L:

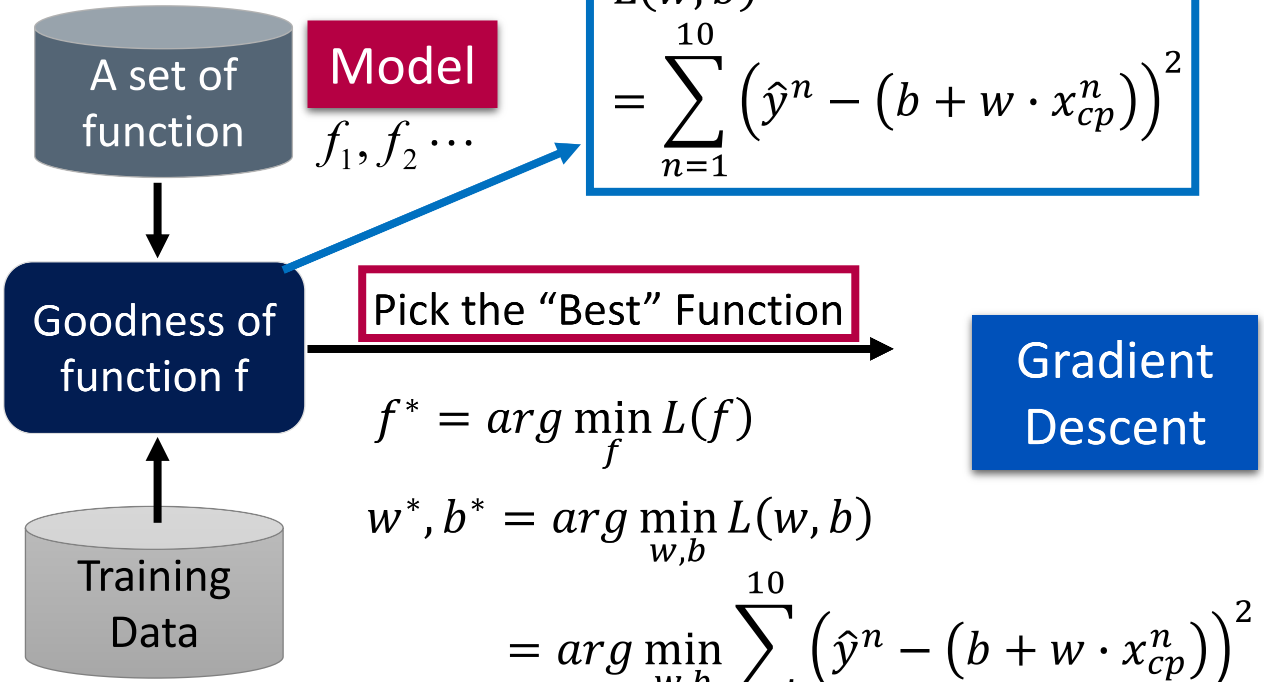
Input: a function, output: how bad it is

Estimation error

$$L(f) = \underbrace{\sum_{n=1}^{10}}_{\text{Sum over examples}} \boxed{\left(\hat{y}^n - \underbrace{f\left(x_{cp}^n\right)}_{\text{Estimated y based on input function}}\right)^2}$$

$$L(w, b) = \sum_{n=1}^{10} \left(\hat{y}^n - \left(b + w \cdot x_{cp}^n\right)\right)^2$$

Step Three: Best Function



$$\mathrm{L}(w, b) = \sum_{n=1}^{10} \left(\hat{y}^n - \left(b + w \cdot x_{cp}^n\right)\right)^2$$

$$f^* = arg \min_f L(f)$$

$$w^*, b^* = arg \min_{w,b} L(w, b)$$

$$= arg \min_{w,b} \sum_{n=1}^{10} \left(\hat{y}^n - \left(b + w \cdot x_{cp}^n\right)\right)^2$$

How are the Results?

$y = b + w \cdot x_{cp}$

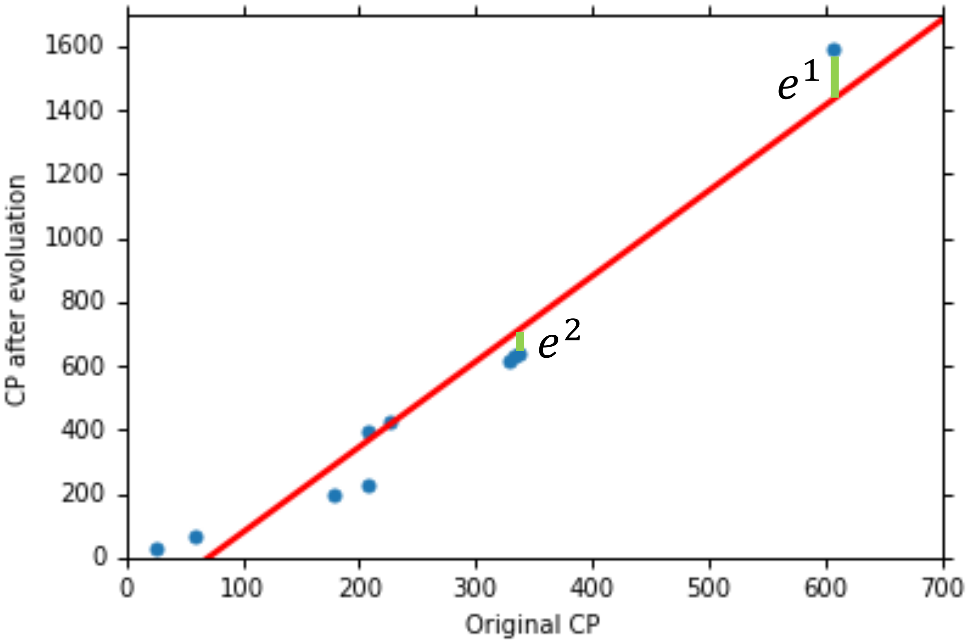
b = -188.4

w = 2.7

Average Error on Training Data

$$= \frac{1}{10}\sum_{n=1}^{10} e^n \quad = 31.9$$

Training Data



How are the Results? - Generalization

What we really care about is the error on new data (testing data)

$y = b + w \cdot x_{cp}$

$b = -188.4$

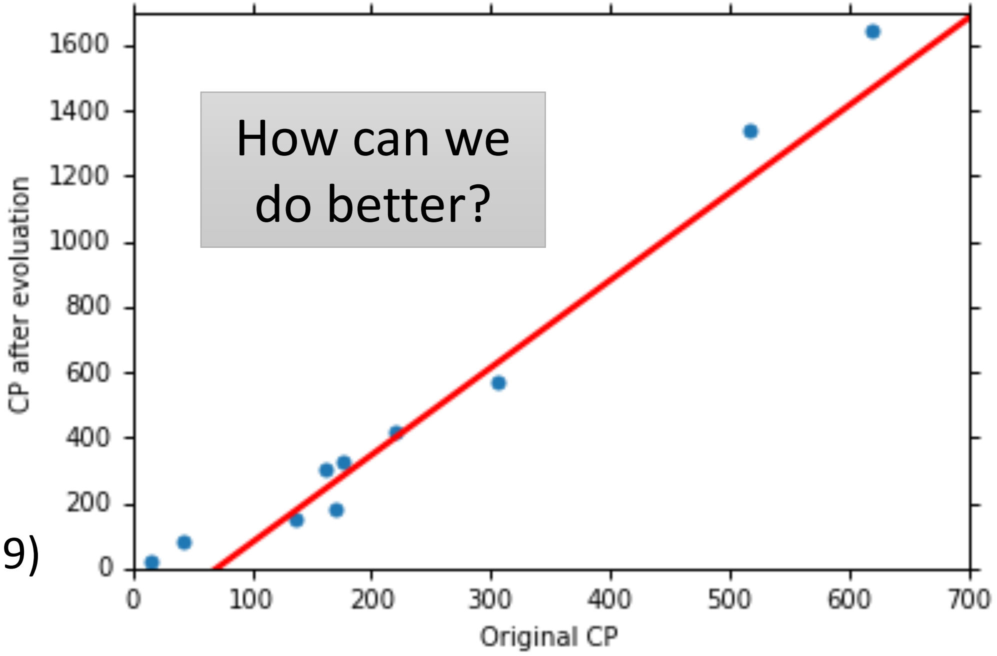
$w = 2.7$

Average Error on Testing Data

$$= \frac{1}{10}\sum_{n=1}^{10} e^n \quad = 35.0$$

> Average Error on Training Data (31.9)

Another 10 Pokémon's as testing data



Selecting Another Model: Part I

$y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2$

Best Function

$b = -10.3$

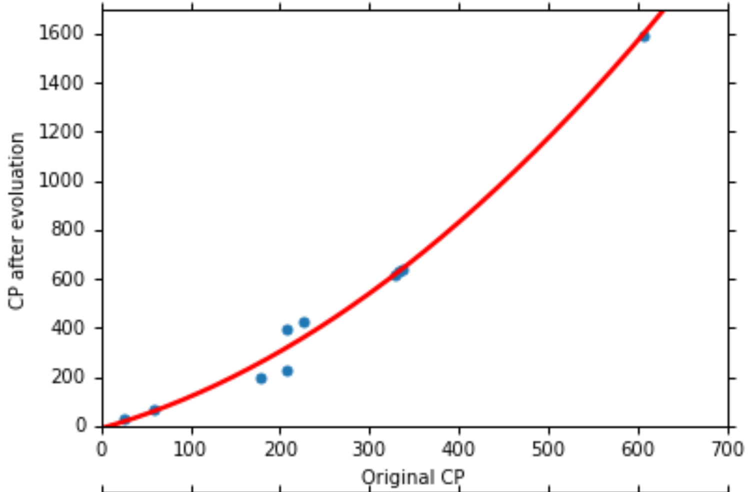
$w_1 = 1.0$, $w_2 = 2.7 \times 10^{-3}$

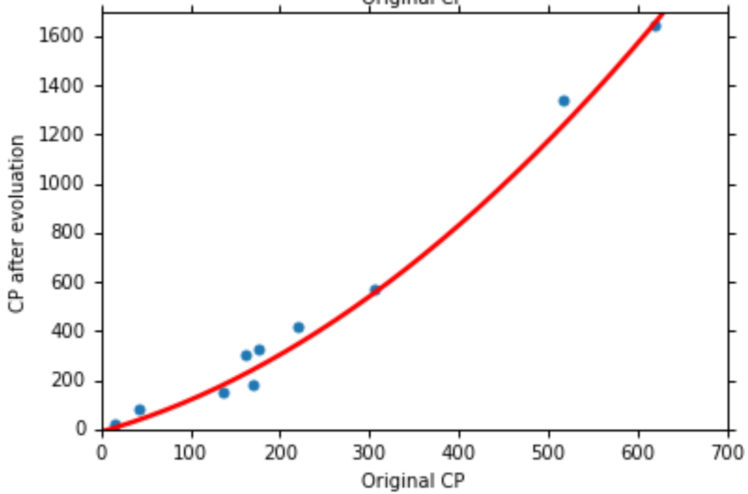
Average Error = 15.4

Testing:

Average Error = 18.4

Better! Could it be even better?





Selecting Another Model: Part II

$y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2$
$+ w_3 \cdot (x_{cp})^3$

Best Function

$b = 6.4,\ w_1 = 0.66$

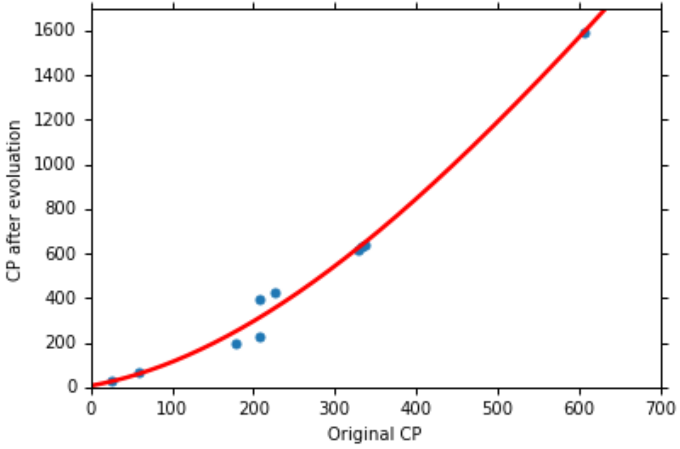
$w_2 = 4.3 \times 10^{-3}$
$w_3 = -1.8 \times 10^{-6}$

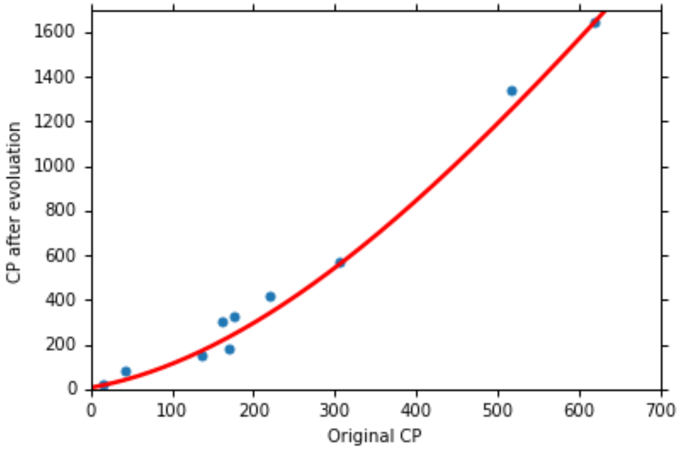
Average Error = 15.3

Testing:

Average Error = 18.1

Slightly better. How about more complex model?





Selecting Another Model: Part III

$$y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2 + w_3 \cdot (x_{cp})^3 + w_4 \cdot (x_{cp})^4$$

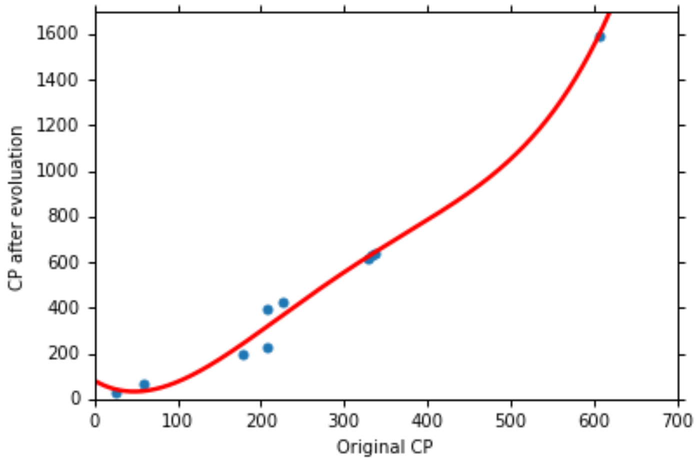
Best Function

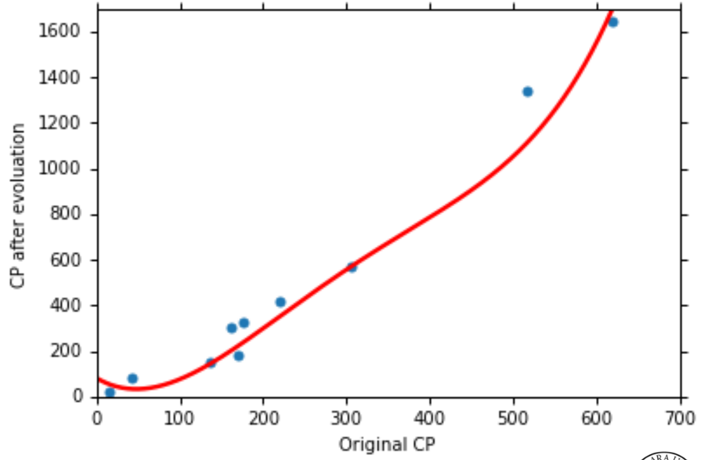
Average Error = 14.9

Testing:

Average Error = 28.8

The results become worse.





Selecting Another Model: Part IV

$$y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2 + w_3 \cdot (x_{cp})^3 + w_4 \cdot (x_{cp})^4 + w_5 \cdot (x_{cp})^5$$

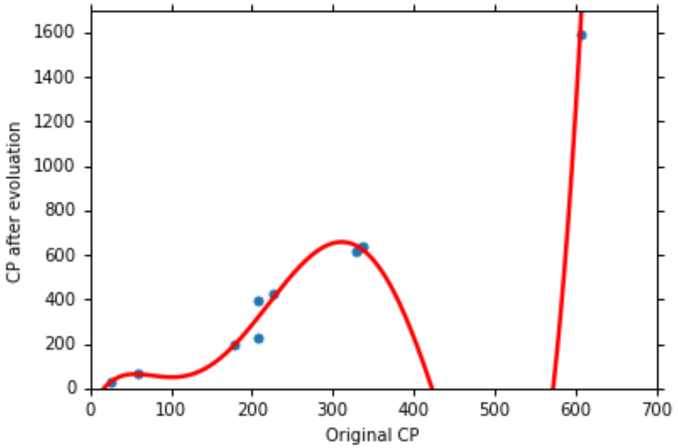
Best Function

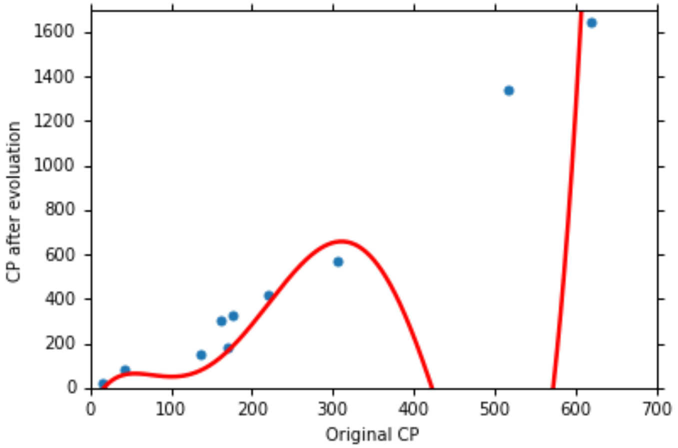
Average Error = 12.8

Testing:

Average Error = 232.1

The results are so bad.

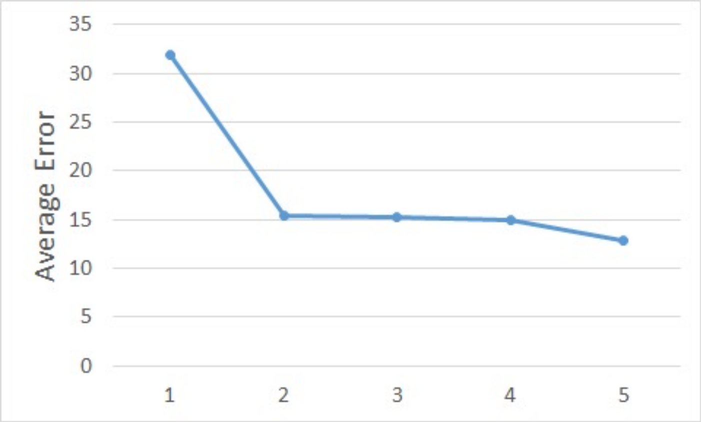


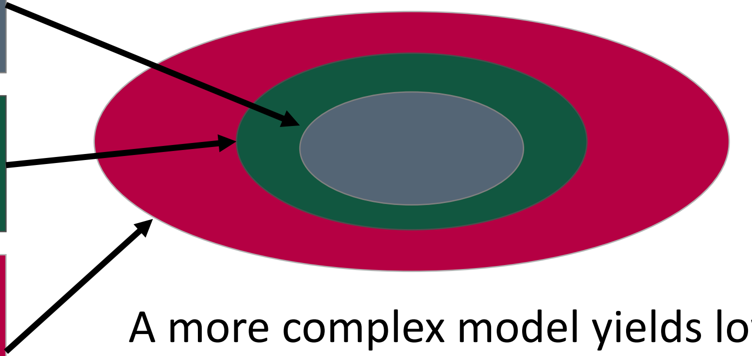


Model Selection

1. $y = b + w \cdot x_{cp}$
2. $y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2$
3. $y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2 + w_3 \cdot (x_{cp})^3$
4. $y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2 + w_3 \cdot (x_{cp})^3 + w_4 \cdot (x_{cp})^4$
5. $y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2 + w_3 \cdot (x_{cp})^3 + w_4 \cdot (x_{cp})^4 + w_5 \cdot (x_{cp})^5$

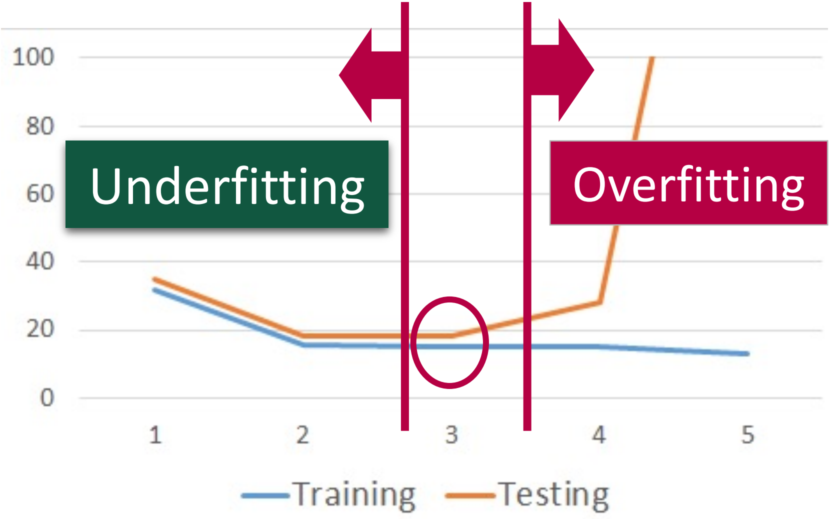
Training Data





A more complex model yields lower error on training data.
If we can truly find the best function.

Model Selection, Cont'd

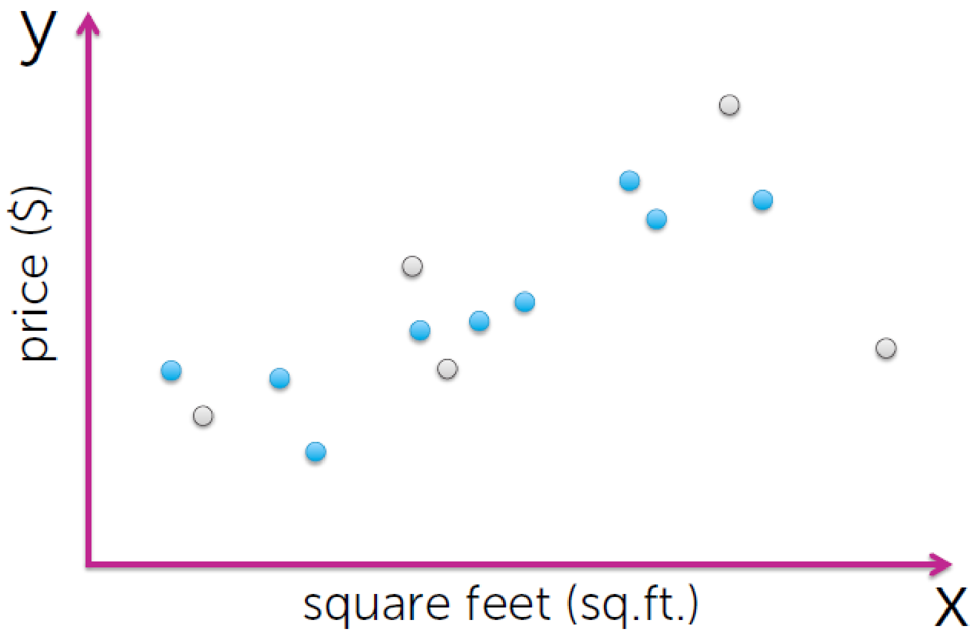


	Training	Testing
1	31.9	35.0
2	15.4	18.4
3	15.3	18.1
4	14.9	28.2
5	12.8	232.1

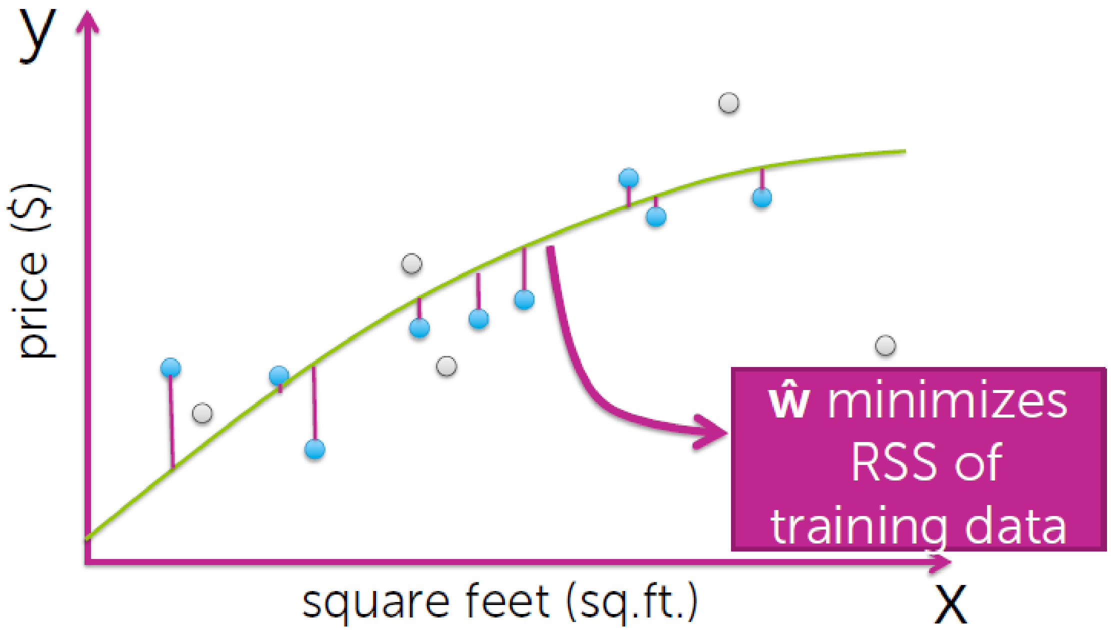
A more complex model does not always lead to better performance on **testing data**.

This is ***overfitting*** → Select suitable model

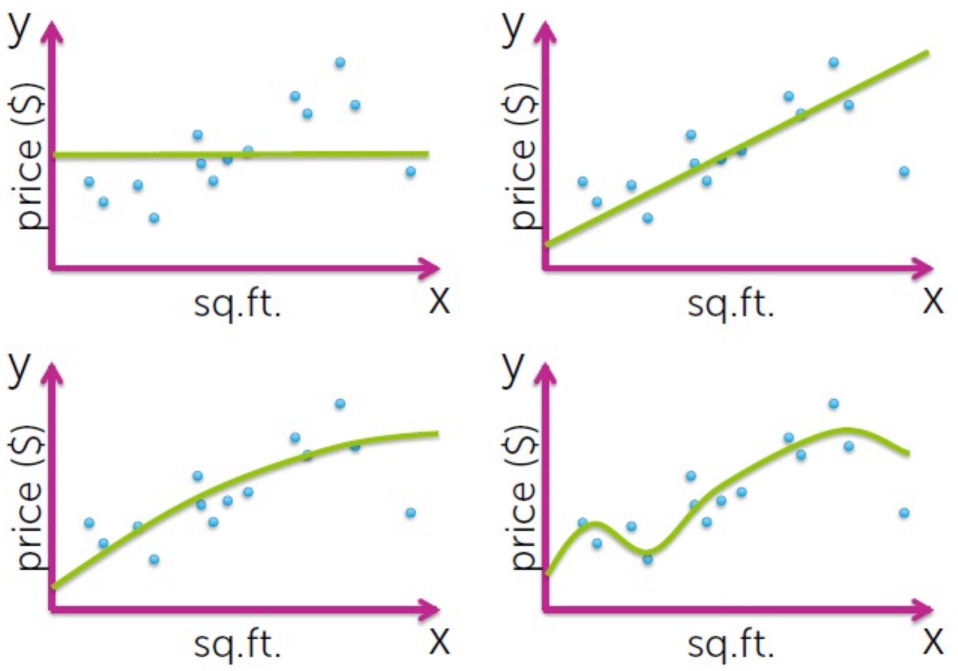
Define Training Data



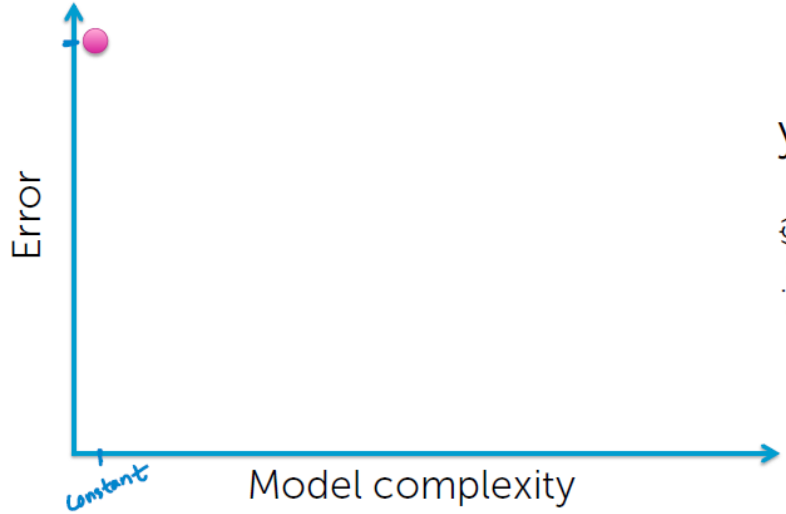
Fit Quadratic to Minimize RSS

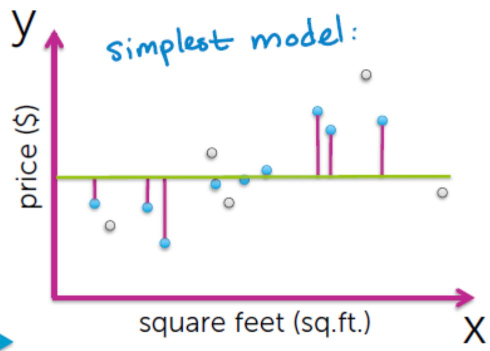


House Price Versus Square Feet

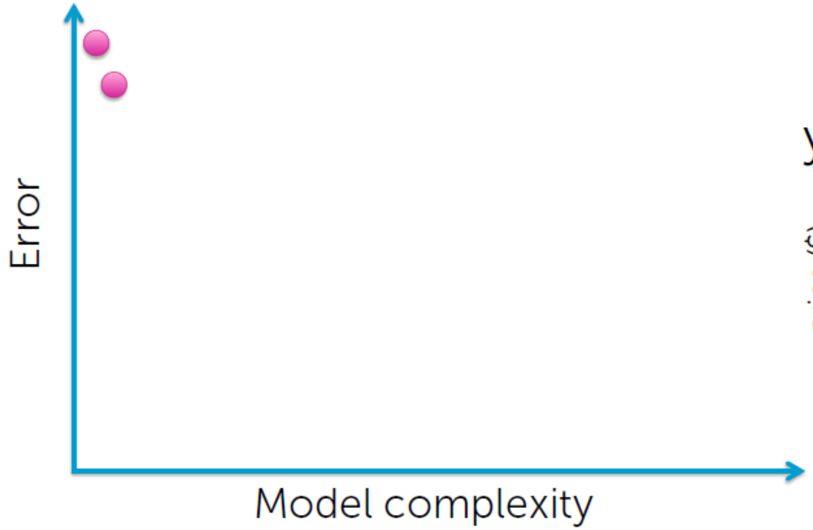


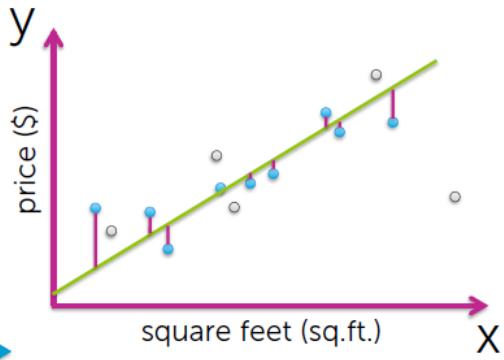
Training Error Versus Model Complexity: Part I



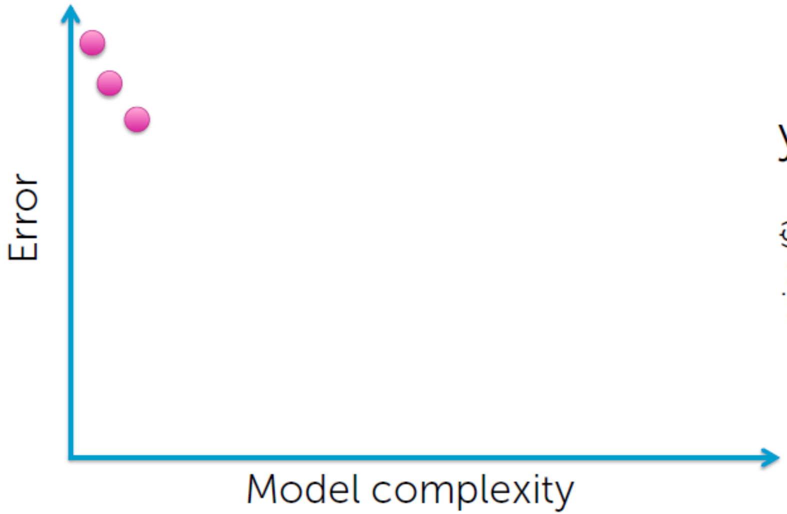


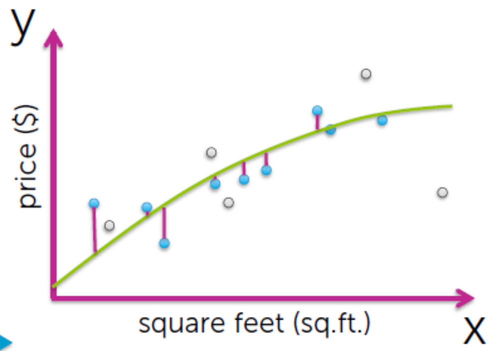
Training Error Versus Model Complexity: Part II



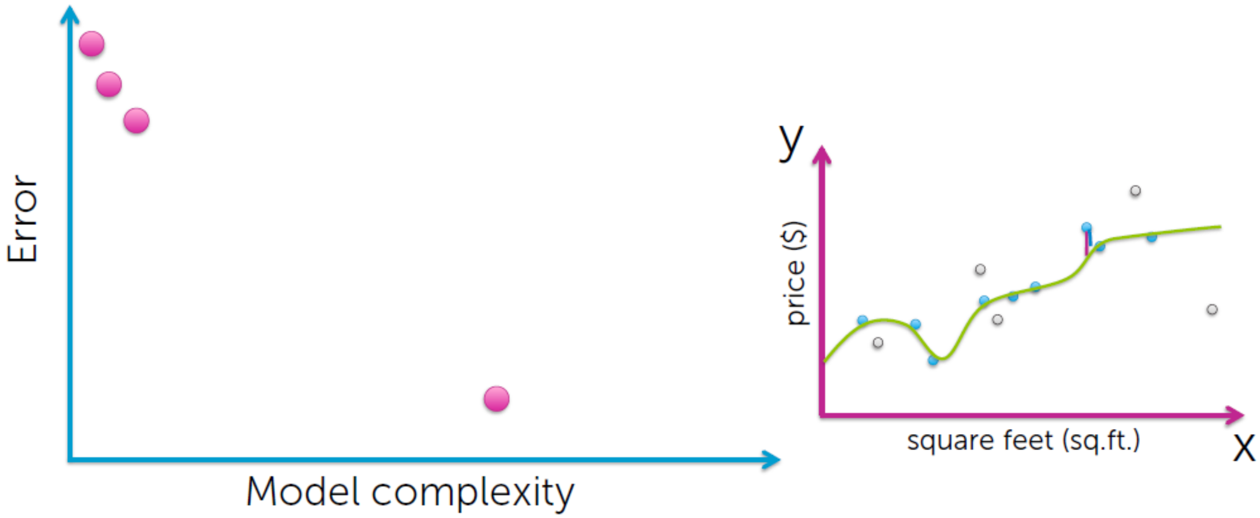


Training Error Versus Model Complexity: Part III

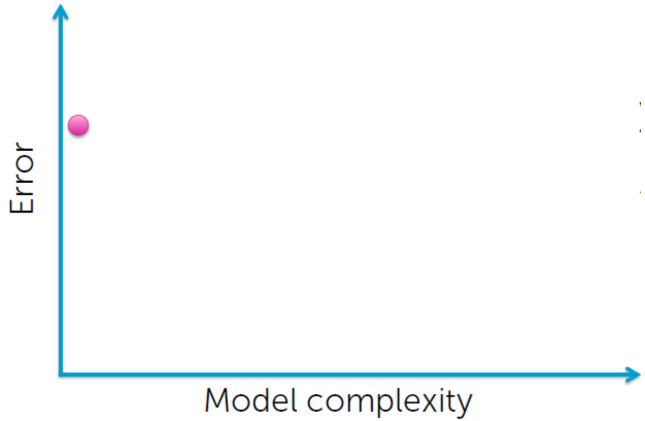


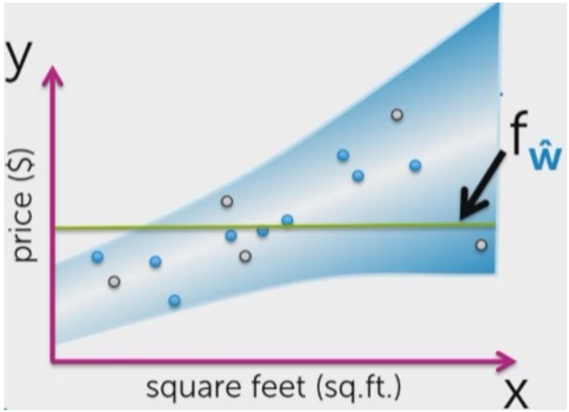


Training Error Versus Model Complexity: Part IV

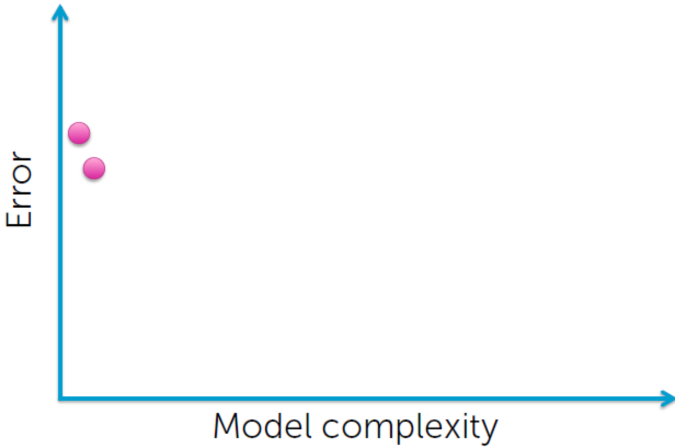


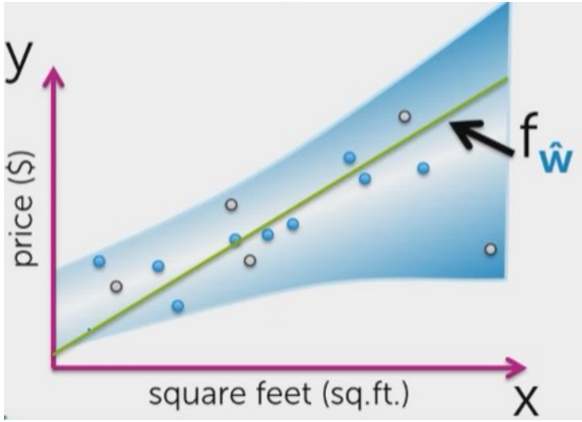
Generalization Error Versus Model Complexity: Part I



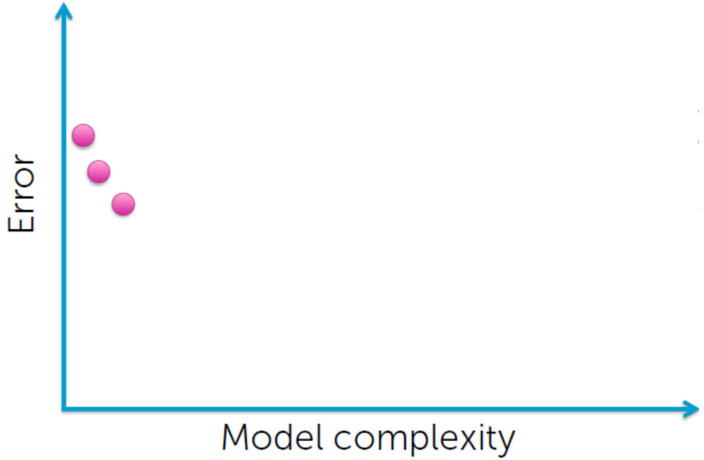


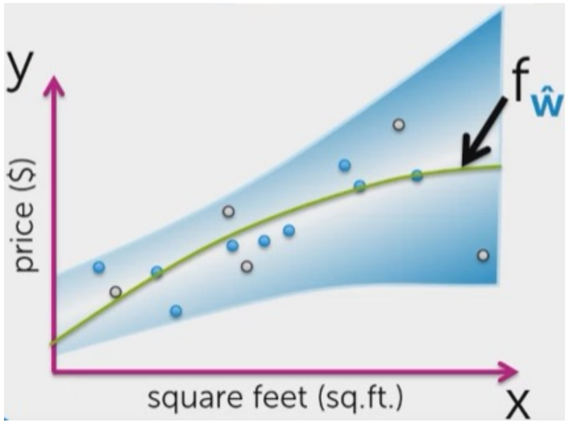
Generalization Error Versus Model Complexity: Part II



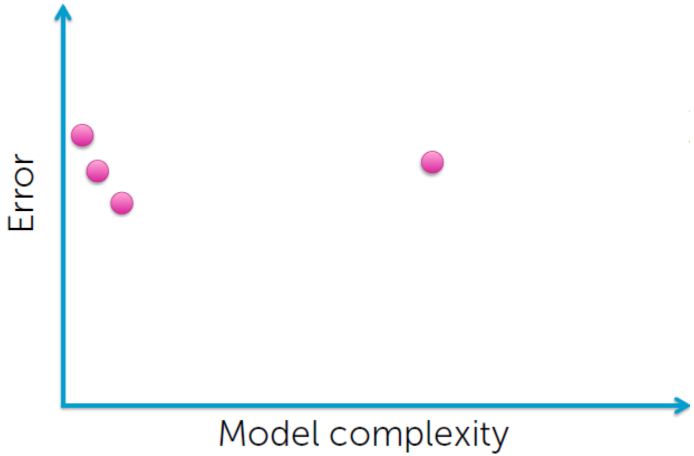


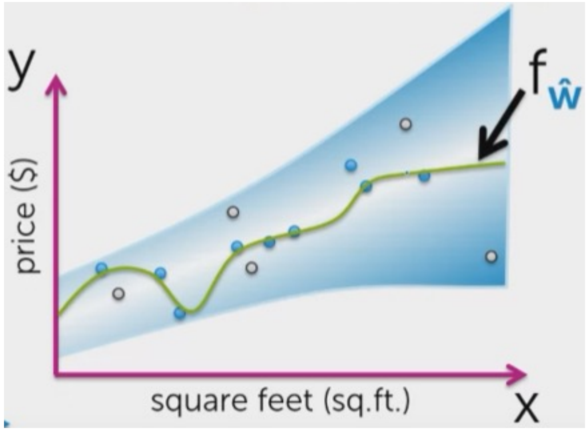
Generalization Error Versus Model Complexity: Part III



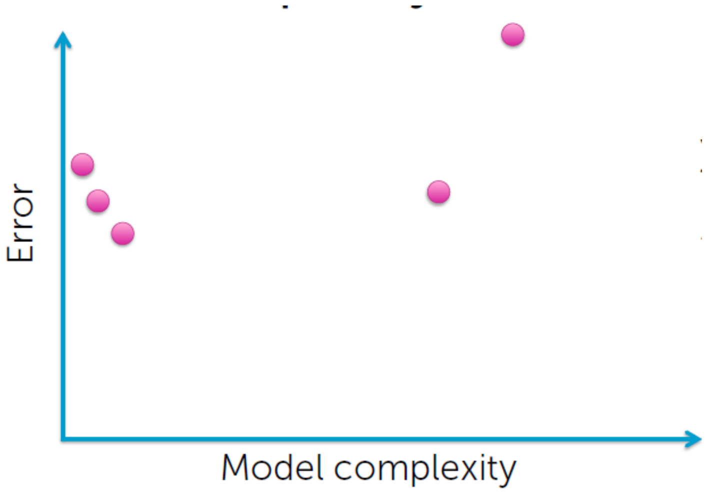


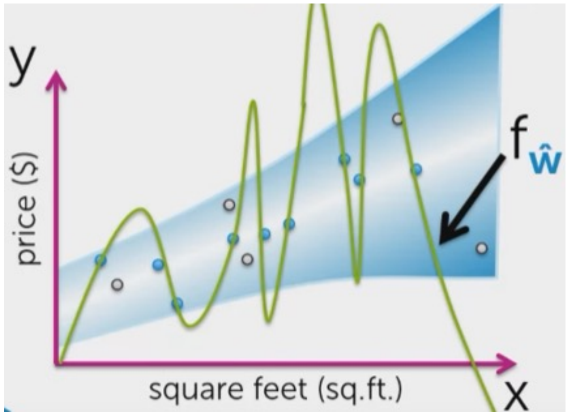
Generalization Error Versus Model Complexity: Part IV



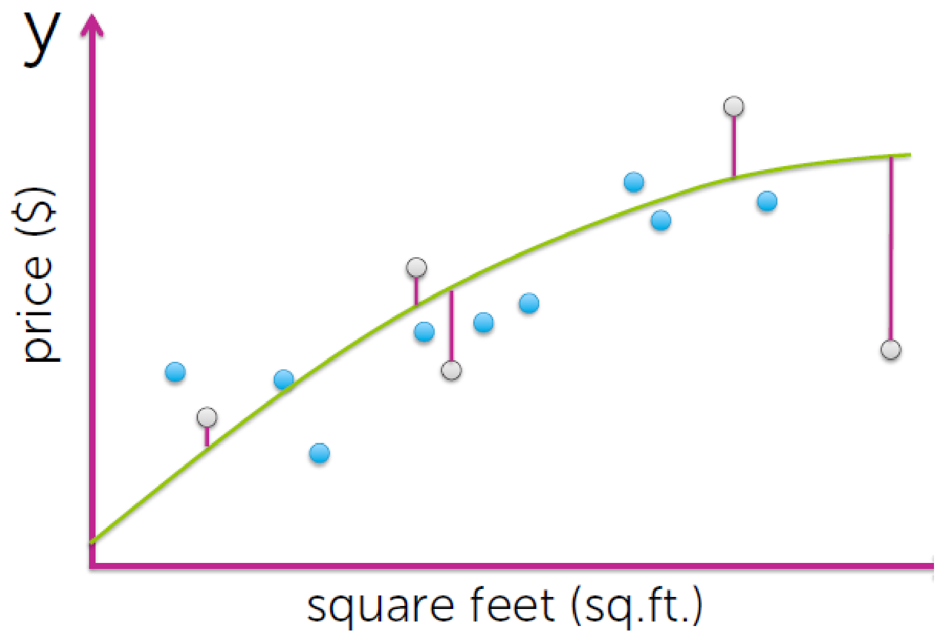


Generalization Error Versus Model Complexity: Part V

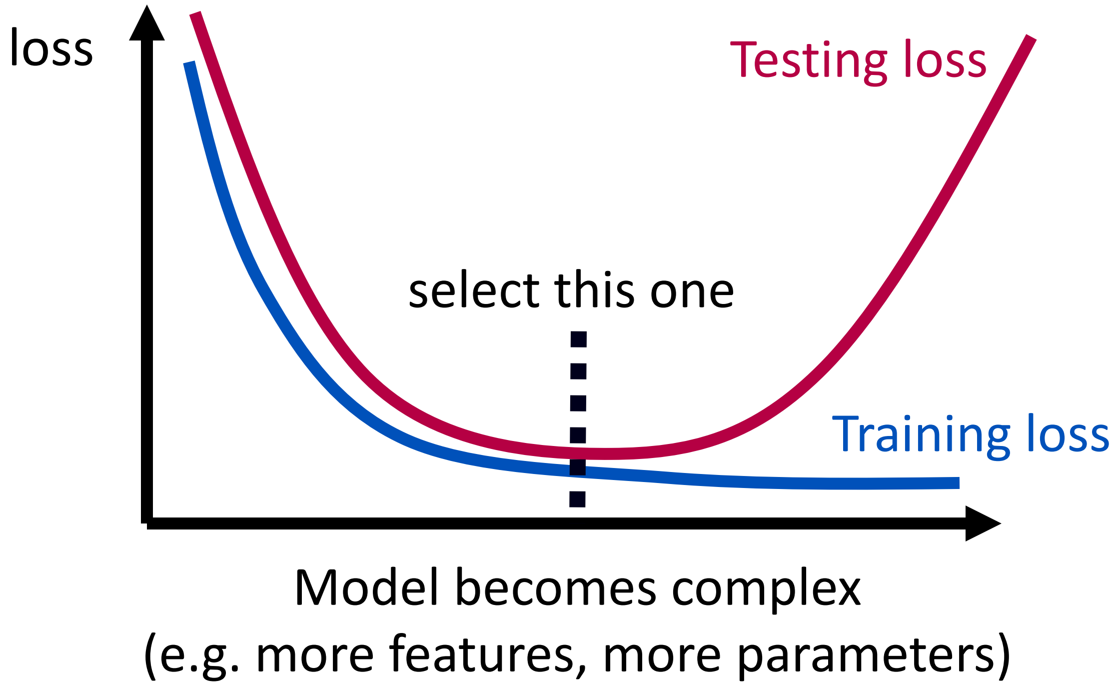




Test Error



Bias-Complexity Trade-off



References

- OpenIntro
    - Pokémon data