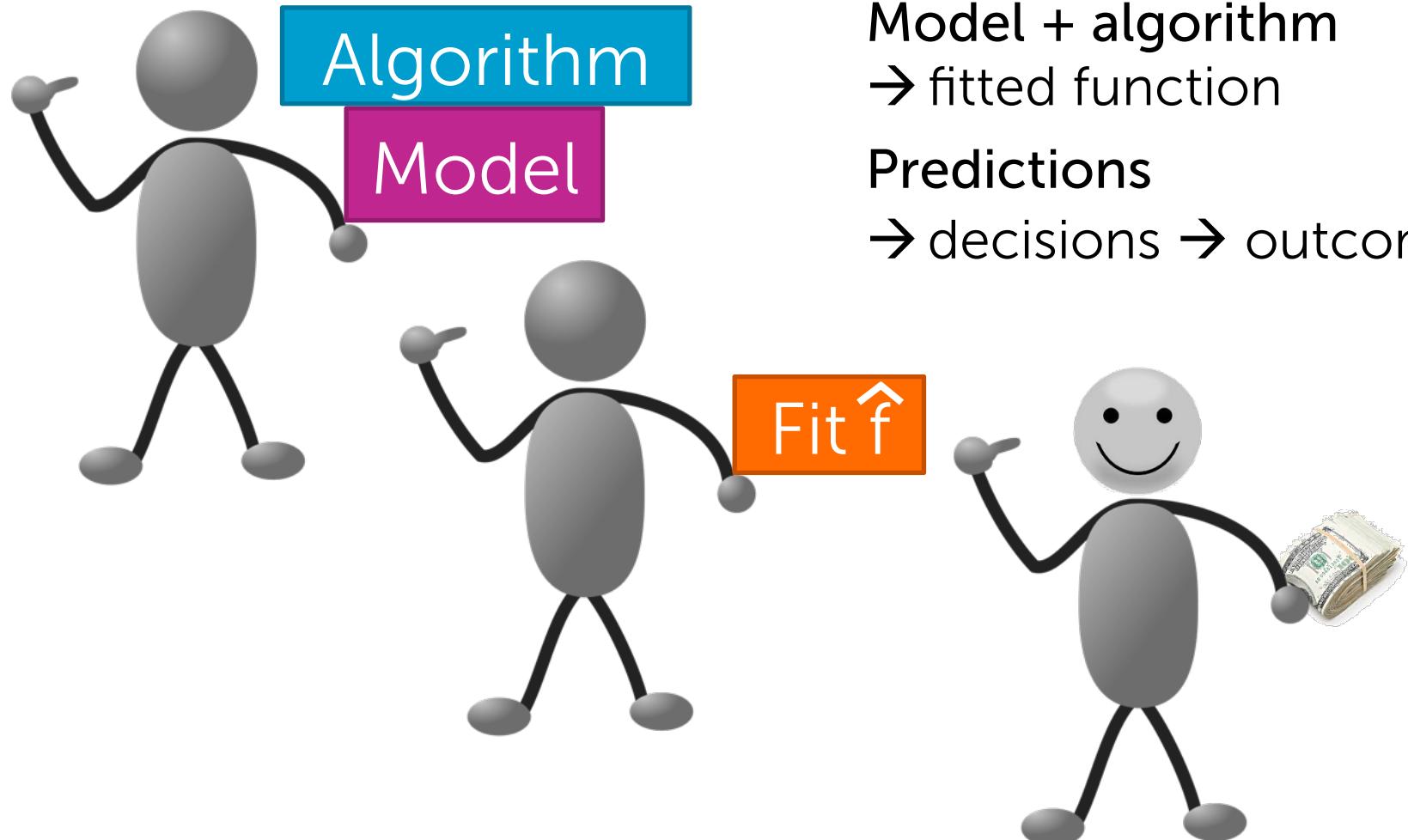


# Assessing Performance

Emily Fox & Carlos Guestrin  
Machine Learning Specialization  
University of Washington

# Make predictions, get \$, right??



Model + algorithm  
→ fitted function

Predictions  
→ decisions → outcome

# 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

Loss function:

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

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

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

Examples:

(assuming loss for underpredicting = overpredicting)

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

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

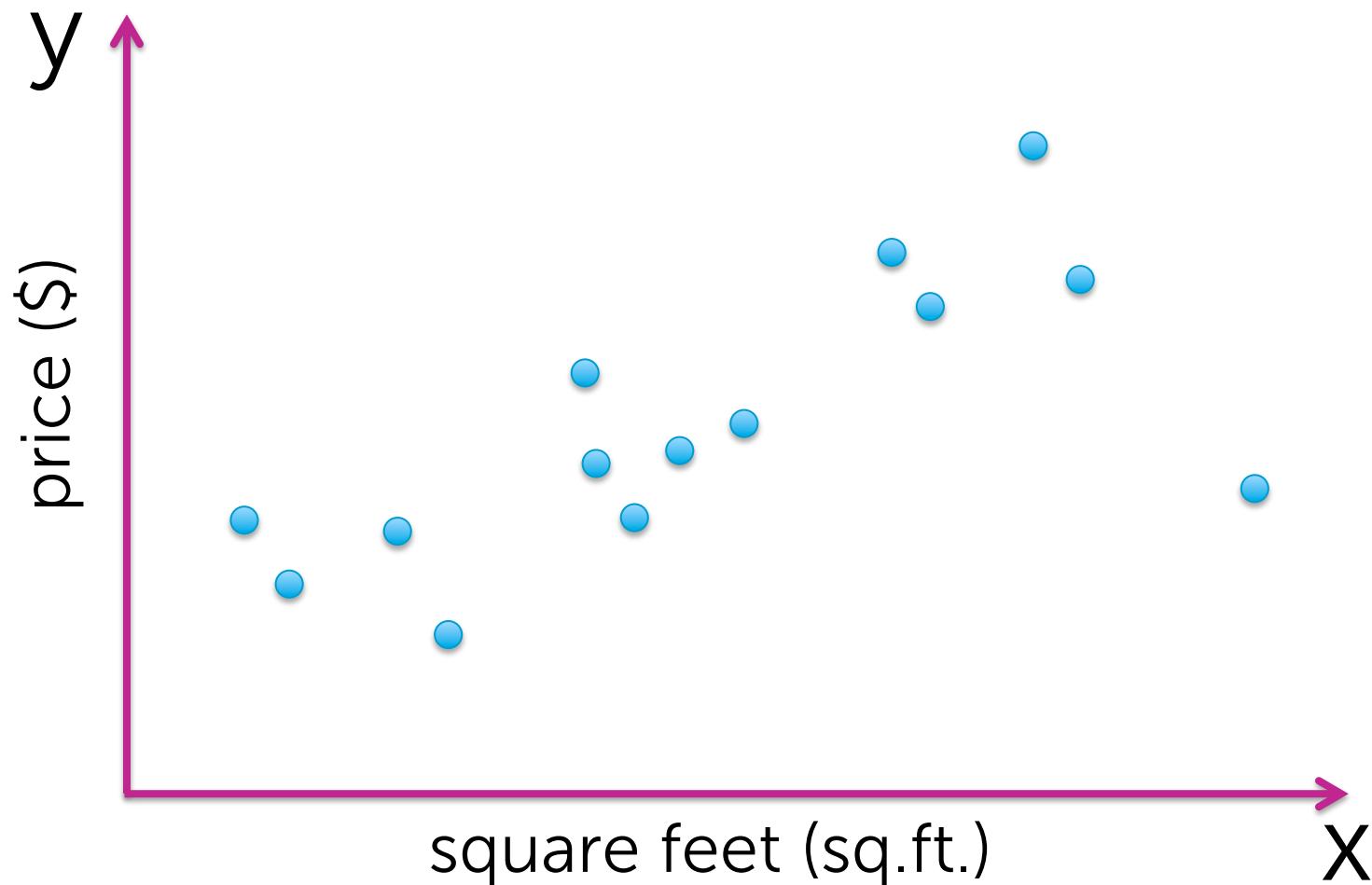
“Remember that all models are wrong; the practical question is how wrong do they have to be to not be useful.” George Box, 1987.

# Assessing the loss

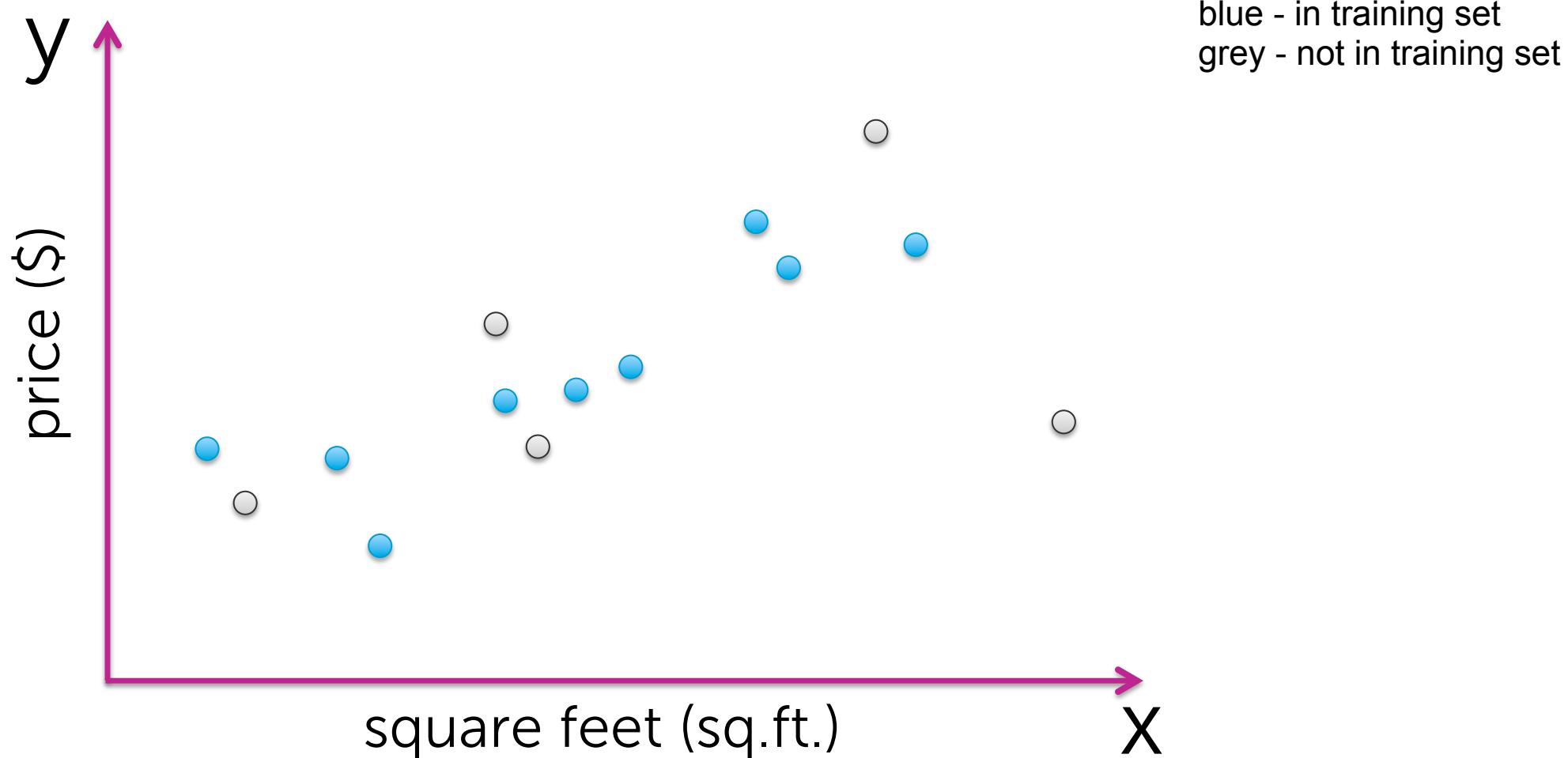
# Assessing the loss

## Part 1: Training error

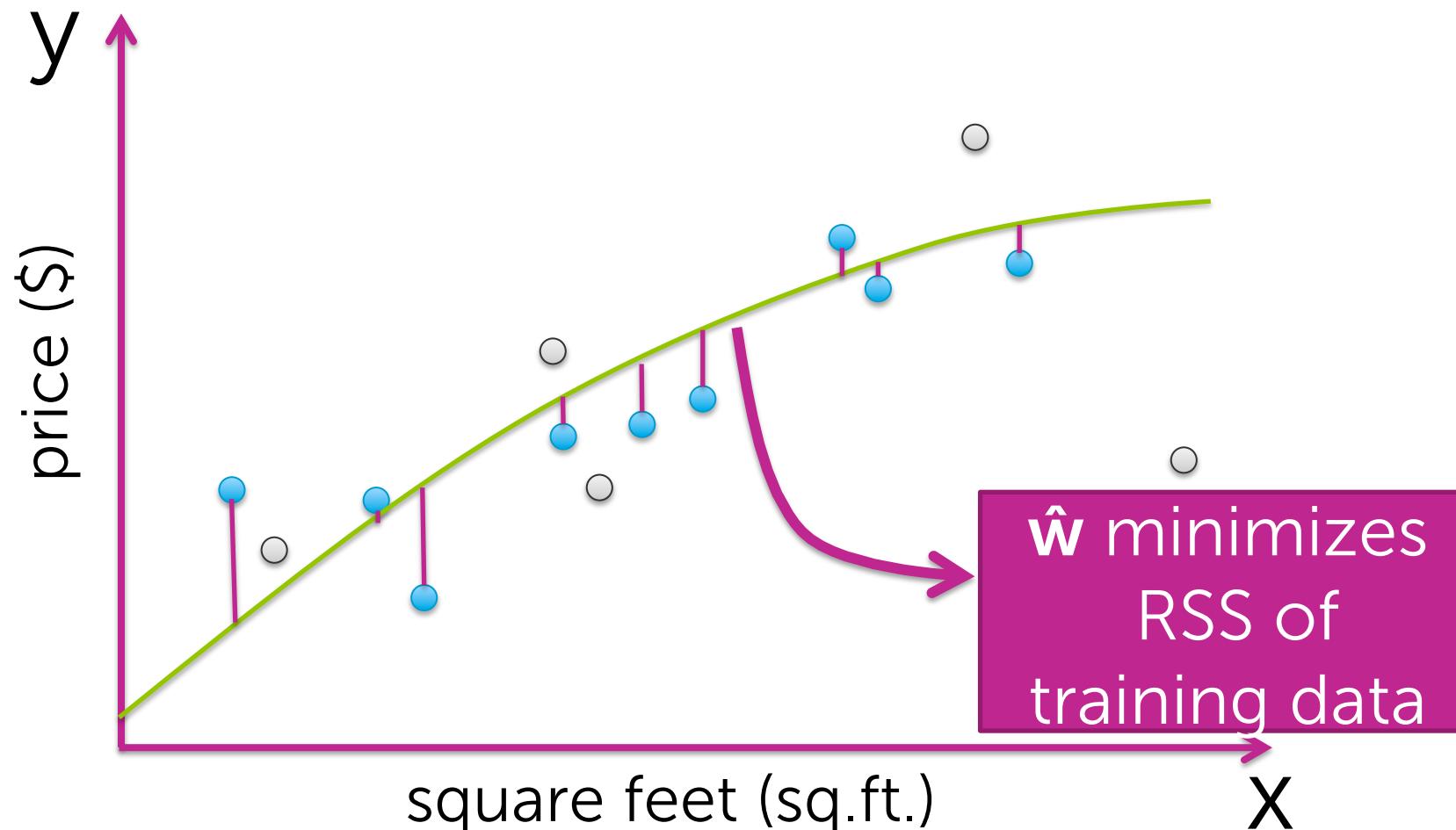
# Define training data



# Define training data



# Example: Fit quadratic to minimize RSS

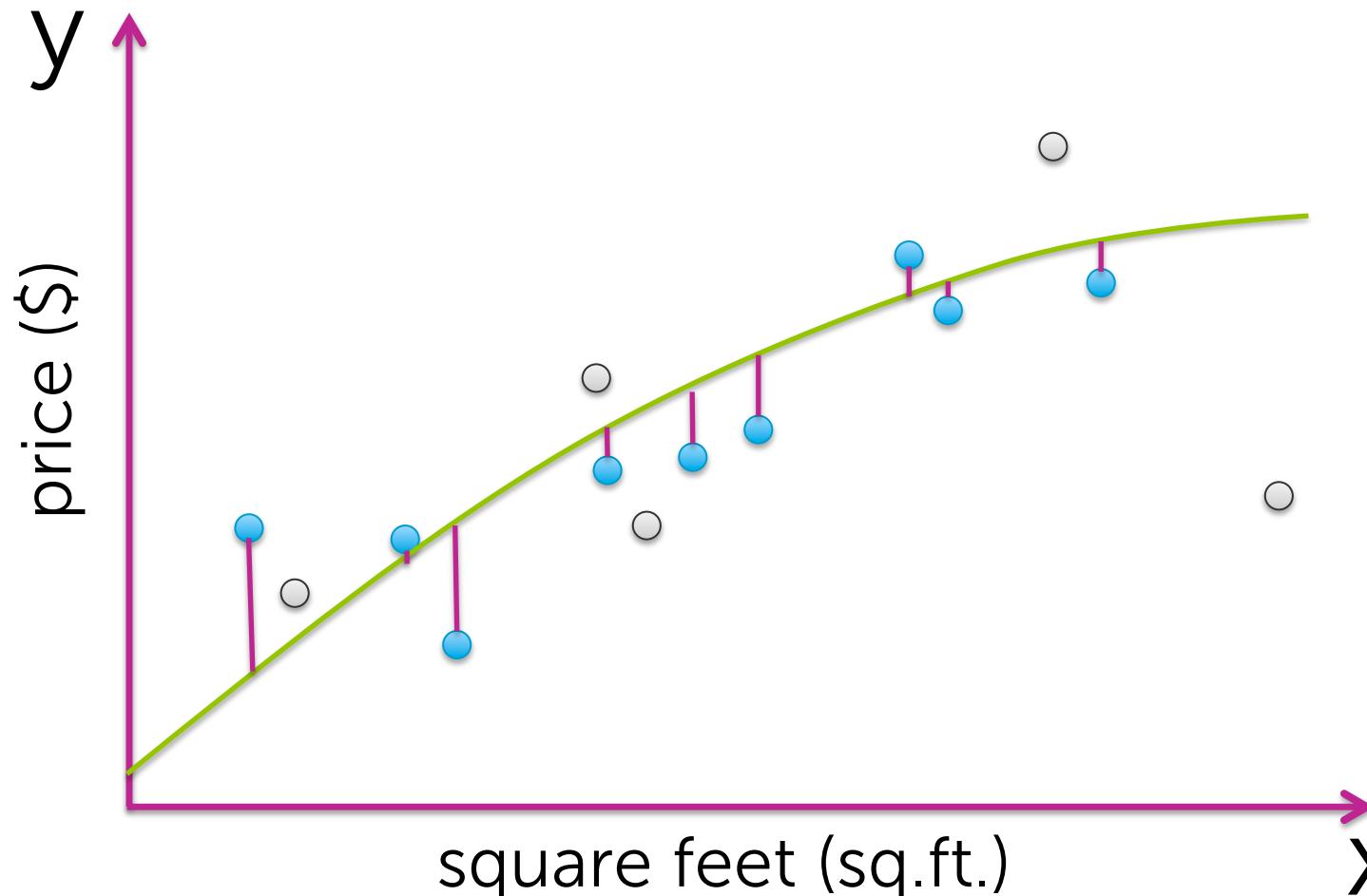


# Compute training error

1. Define a loss function  $L(y, f_{\hat{w}}(\mathbf{x}))$ 
  - E.g., squared error, absolute error,...
2. Training error
  - = avg. loss on houses in **training set**
  - =  $\frac{1}{N} \sum_{i=1}^N L(y_i, f_{\hat{w}}(\mathbf{x}_i))$ 

fit using training data

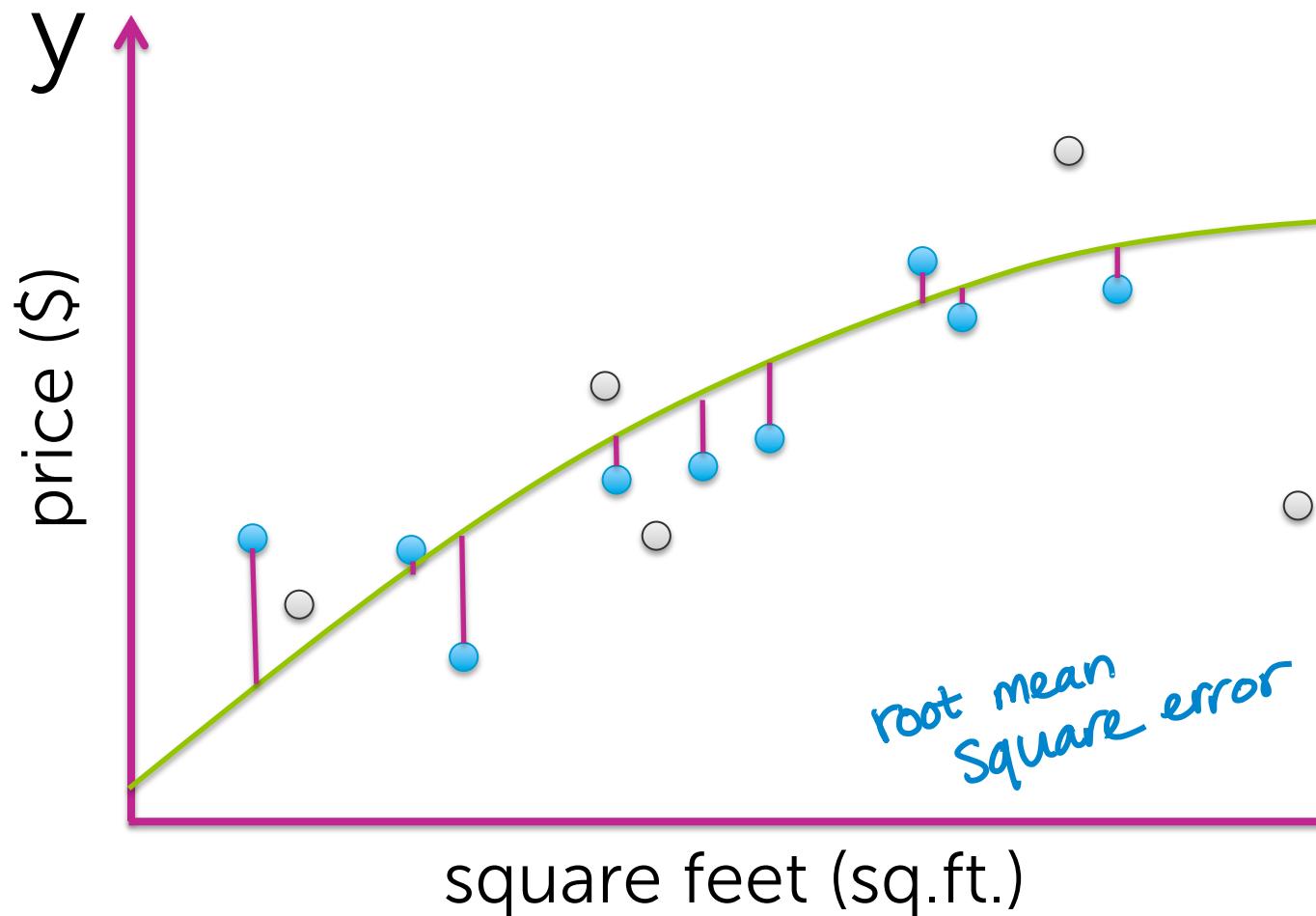
# Example: Use squared error loss $(y - f_{\hat{w}}(x))^2$



Using squared error,  
training error =  $1/N * \text{RSS}$

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

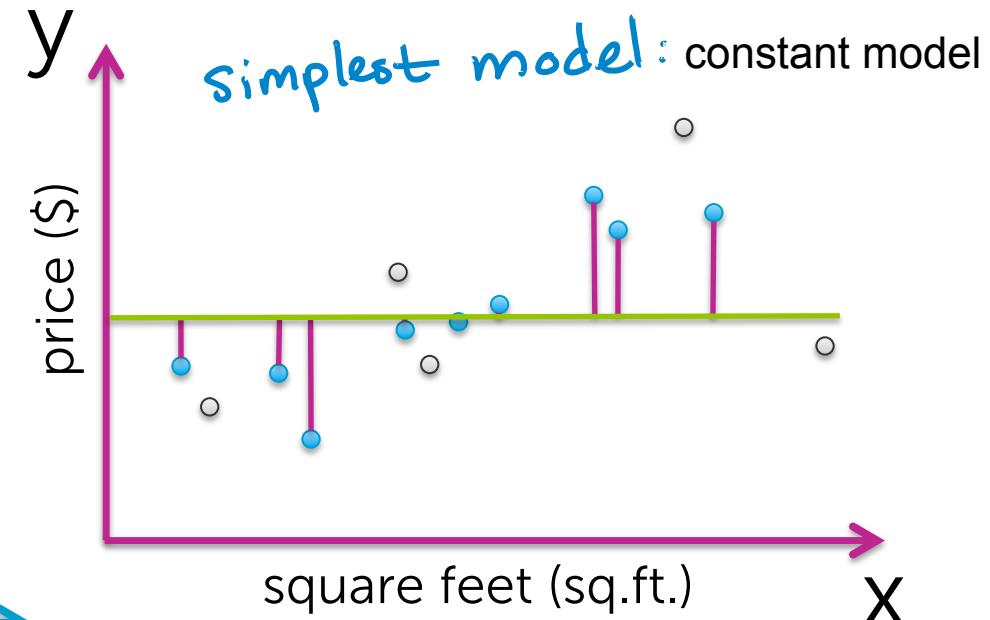
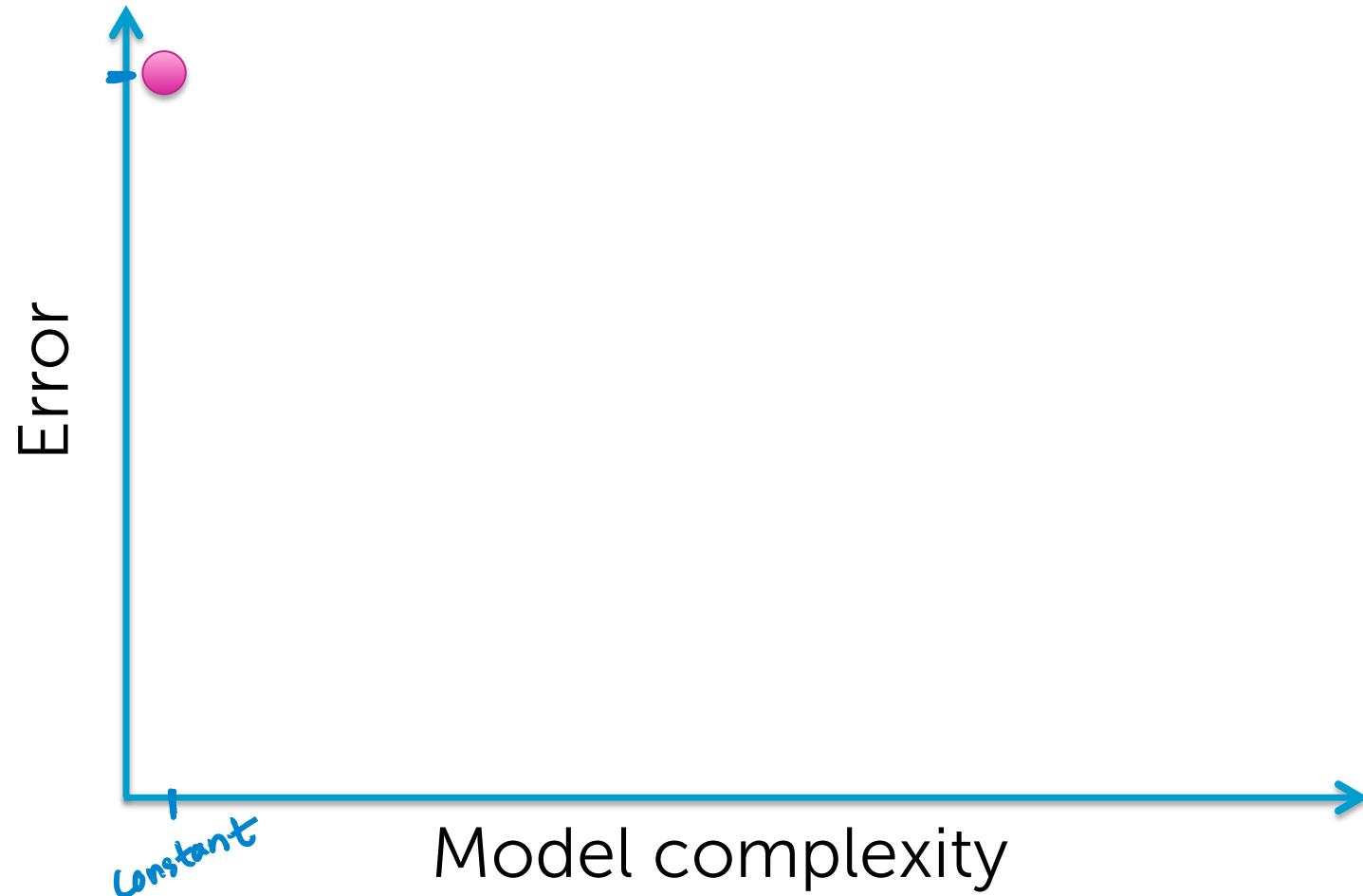
# Example: Use squared error loss $(y - f_{\hat{w}}(x))^2$



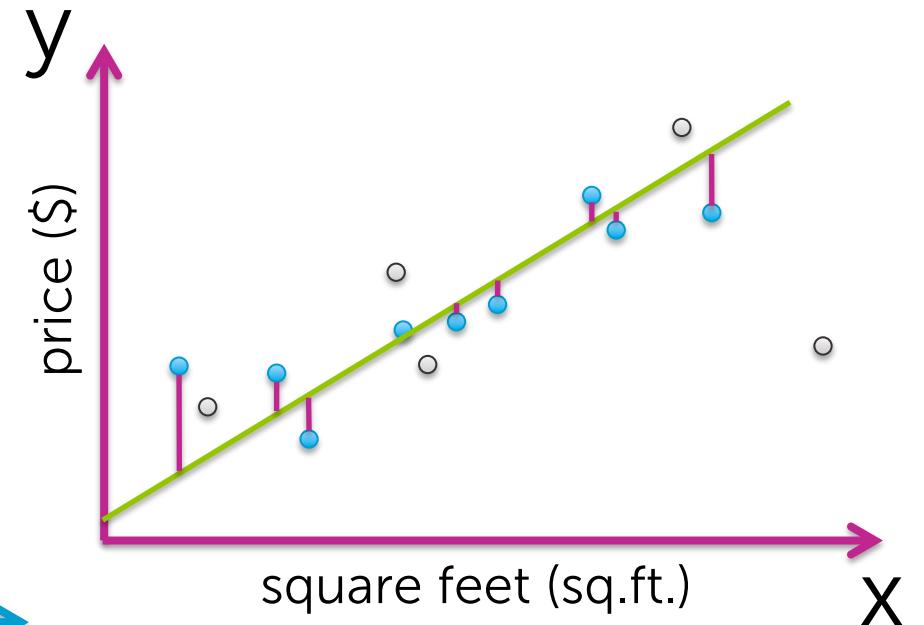
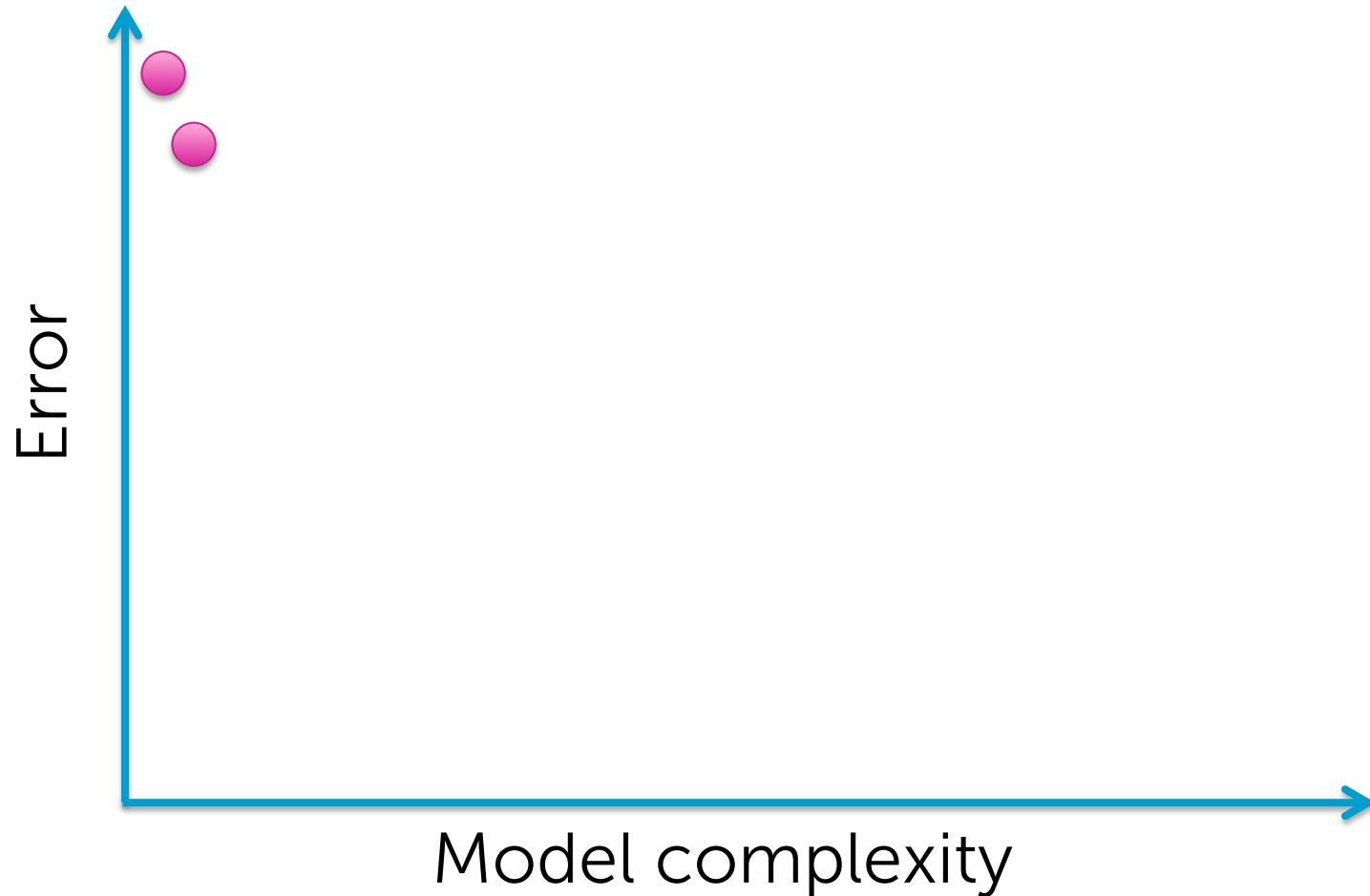
$$\text{Training error } (\hat{w}) = \frac{1}{N} \sum_{i=1}^N (y_i - f_{\hat{w}}(x_i))^2$$

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

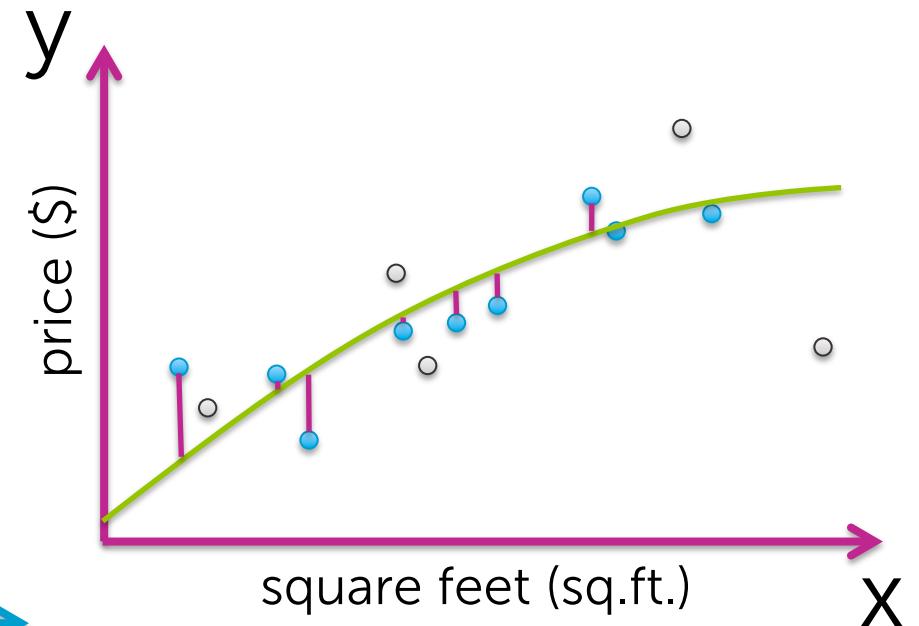
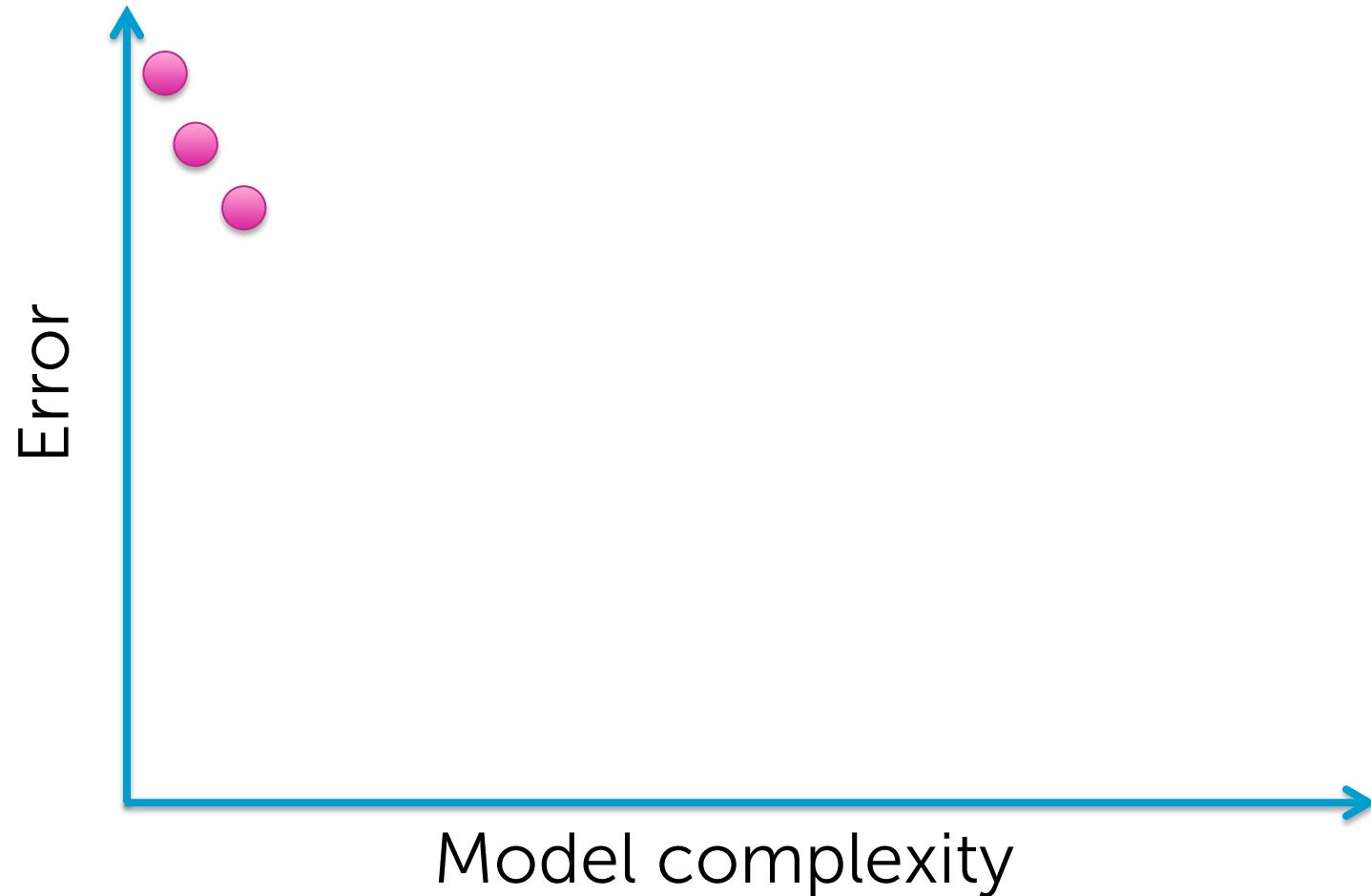
# Training error vs. model complexity



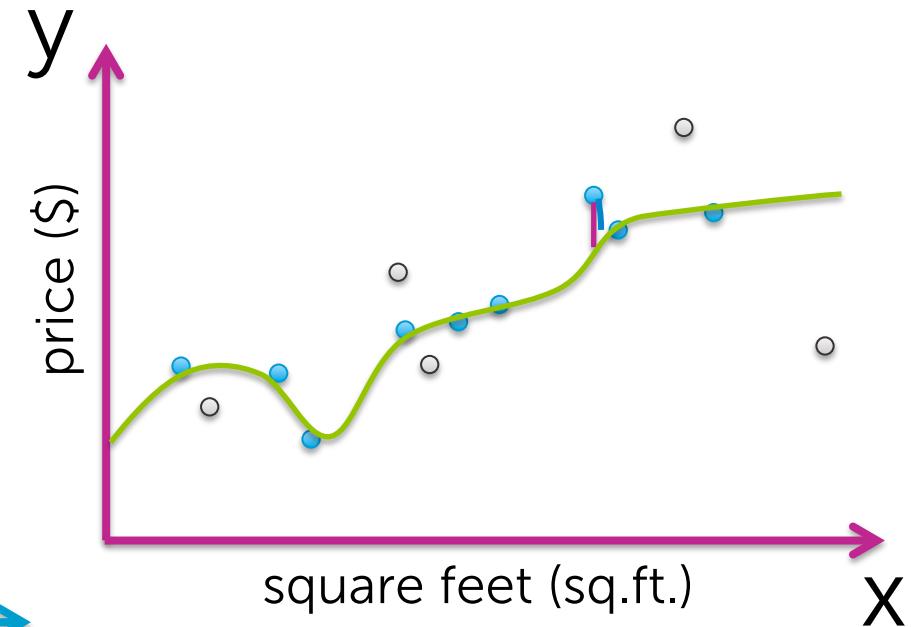
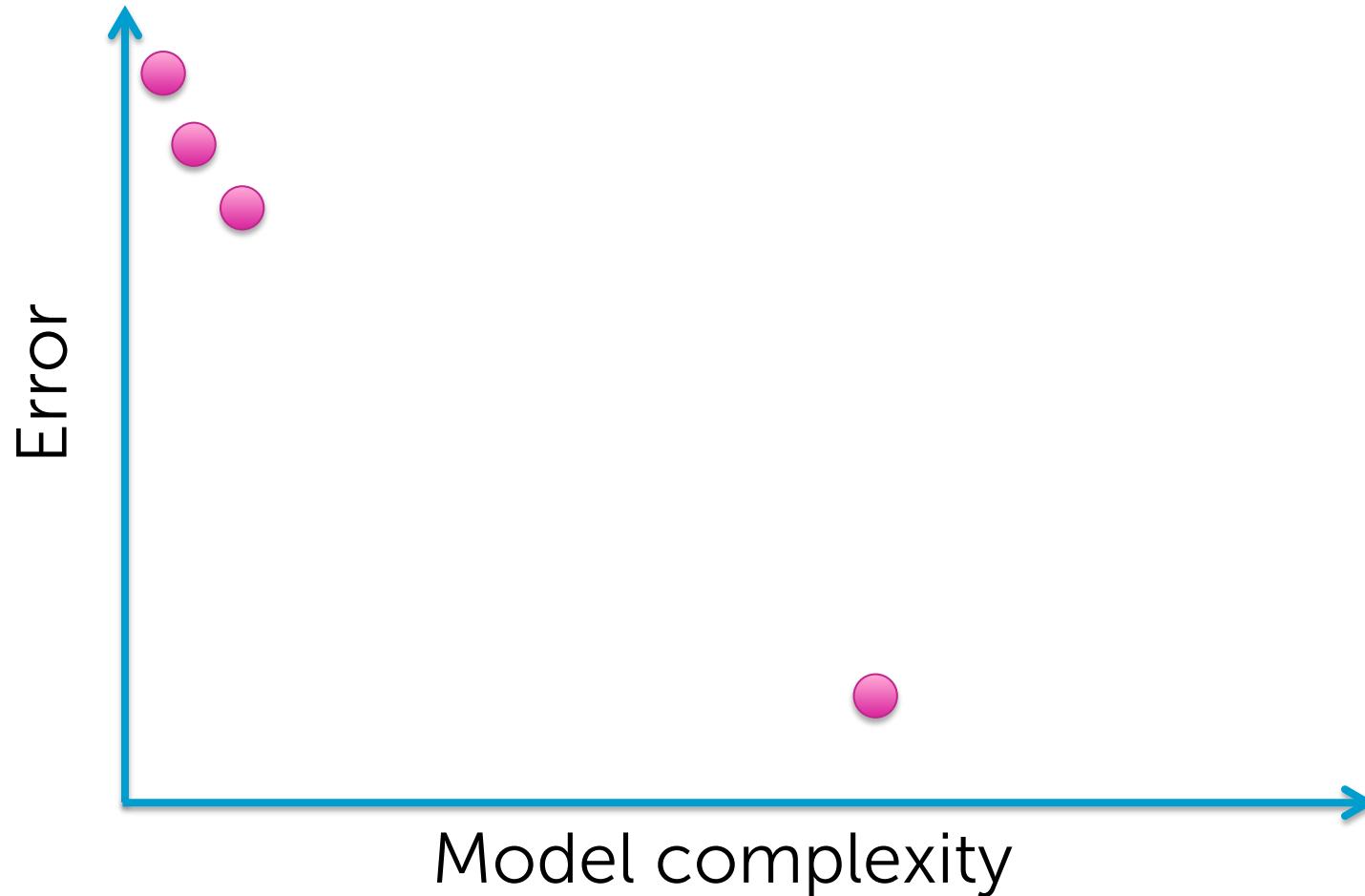
# Training error vs. model complexity



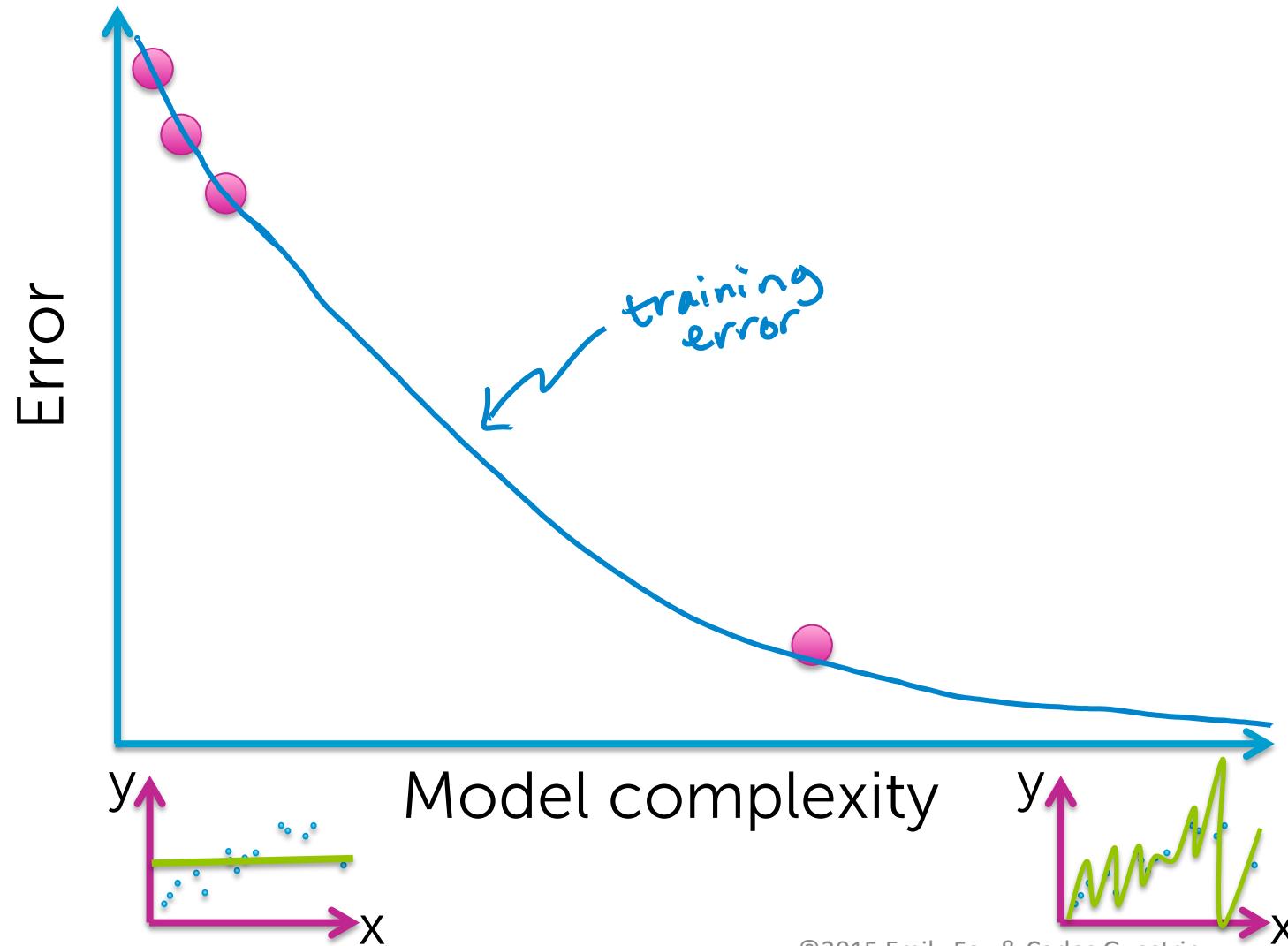
# Training error vs. model complexity



# Training error vs. model complexity

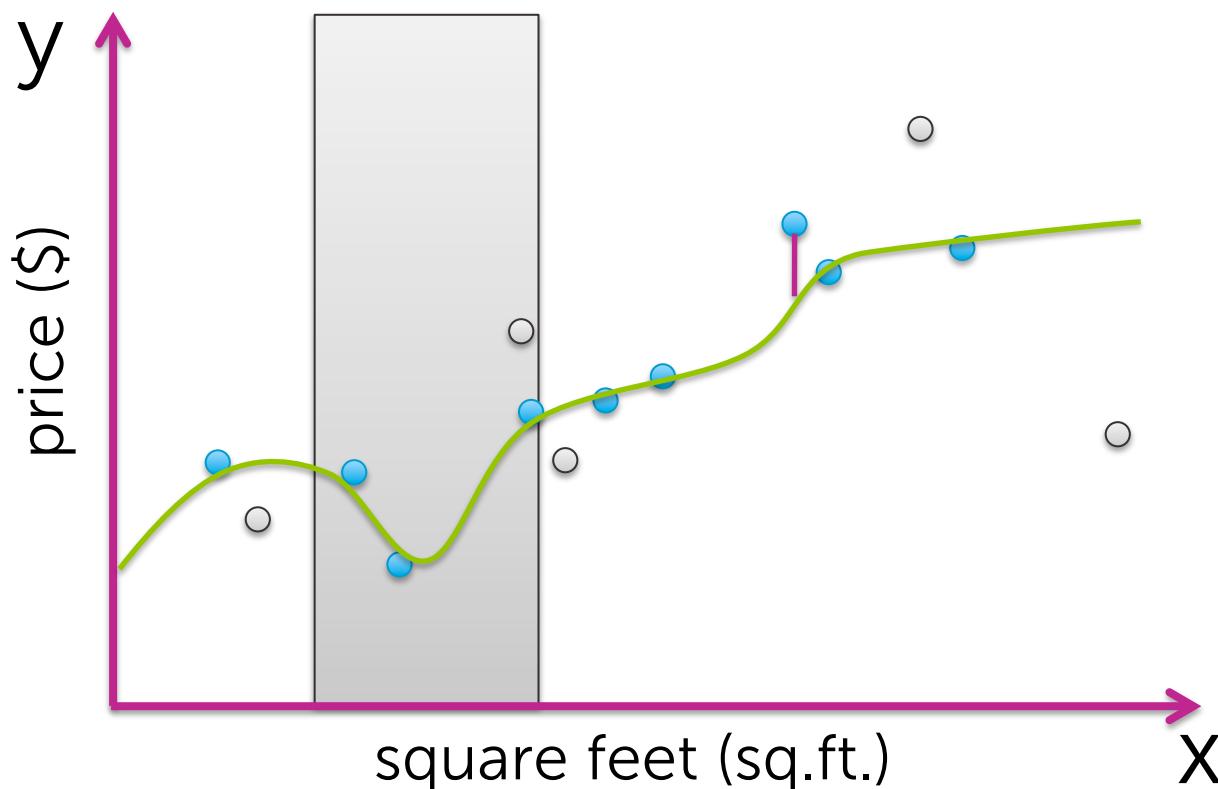


# Training error vs. model complexity



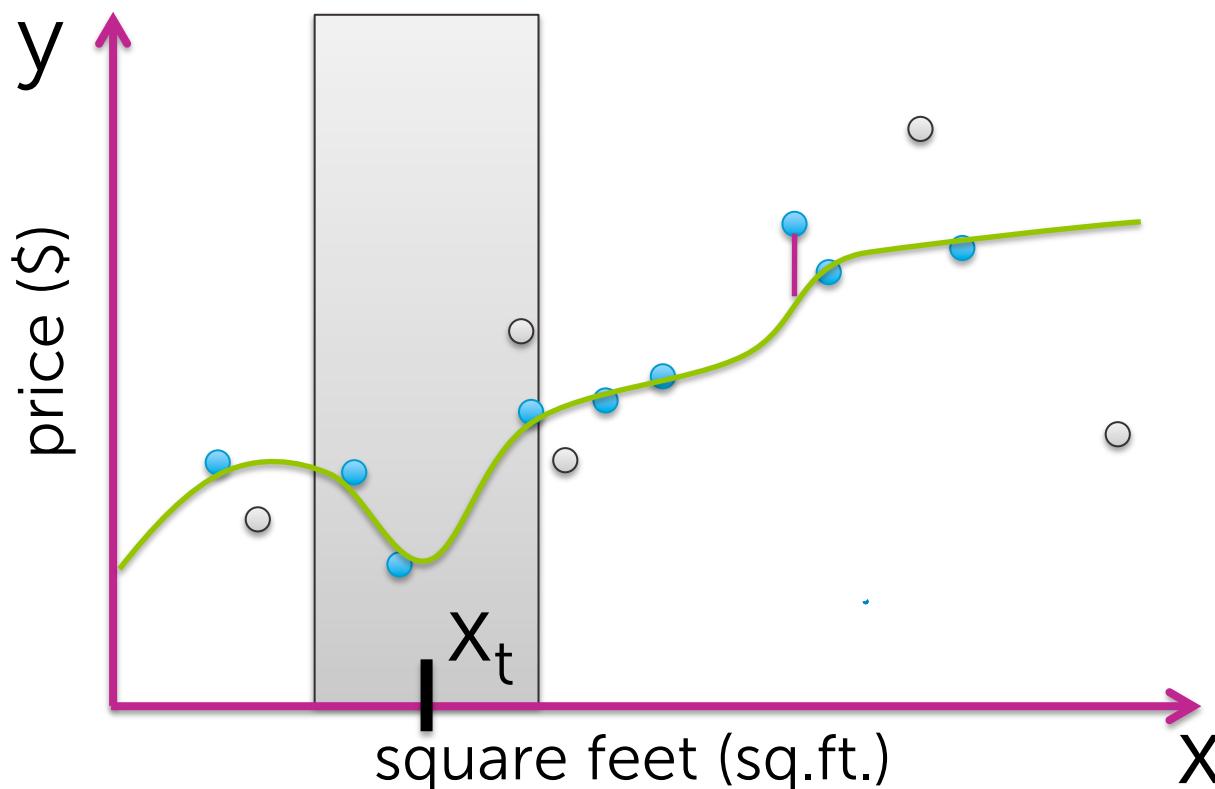
# Is training error a good measure of predictive performance?

How do we expect to perform on a new house?



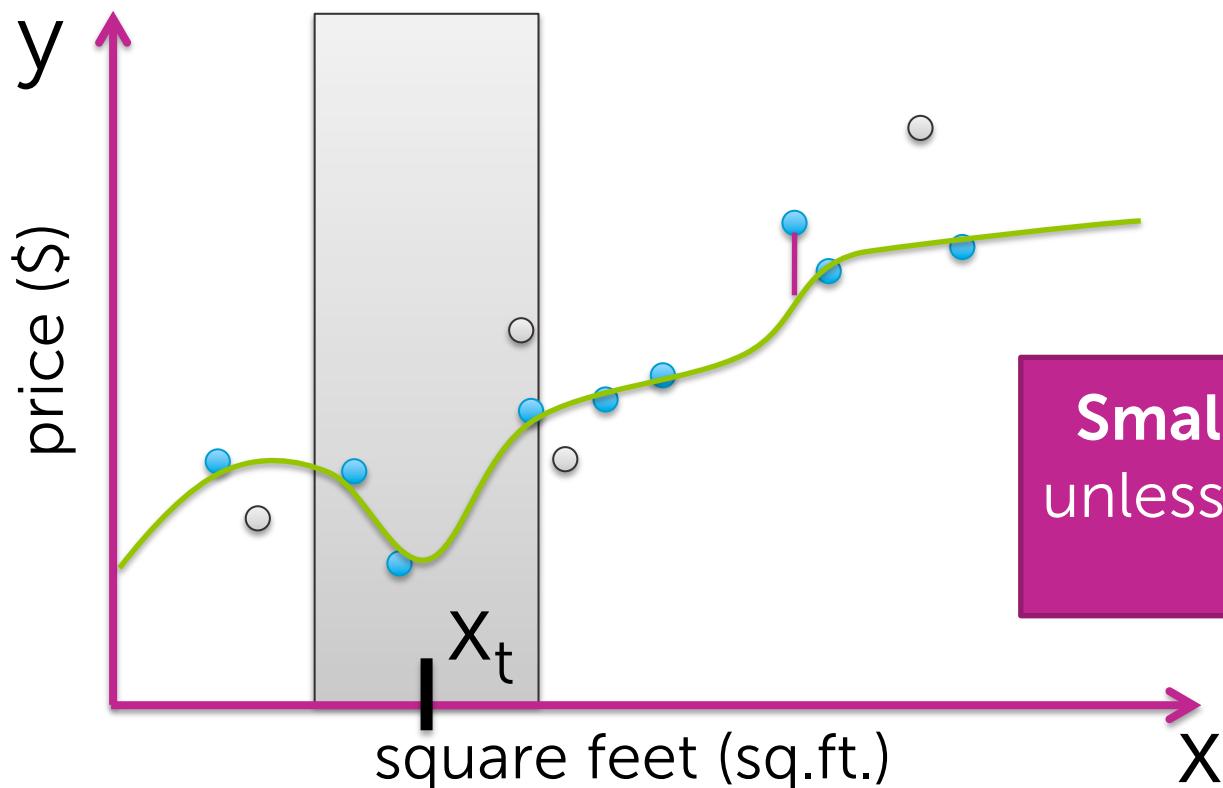
# Is training error a good measure of predictive performance?

Is there something particularly bad about having  $x_t$  square feet???



# Is training error a good measure of predictive performance?

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



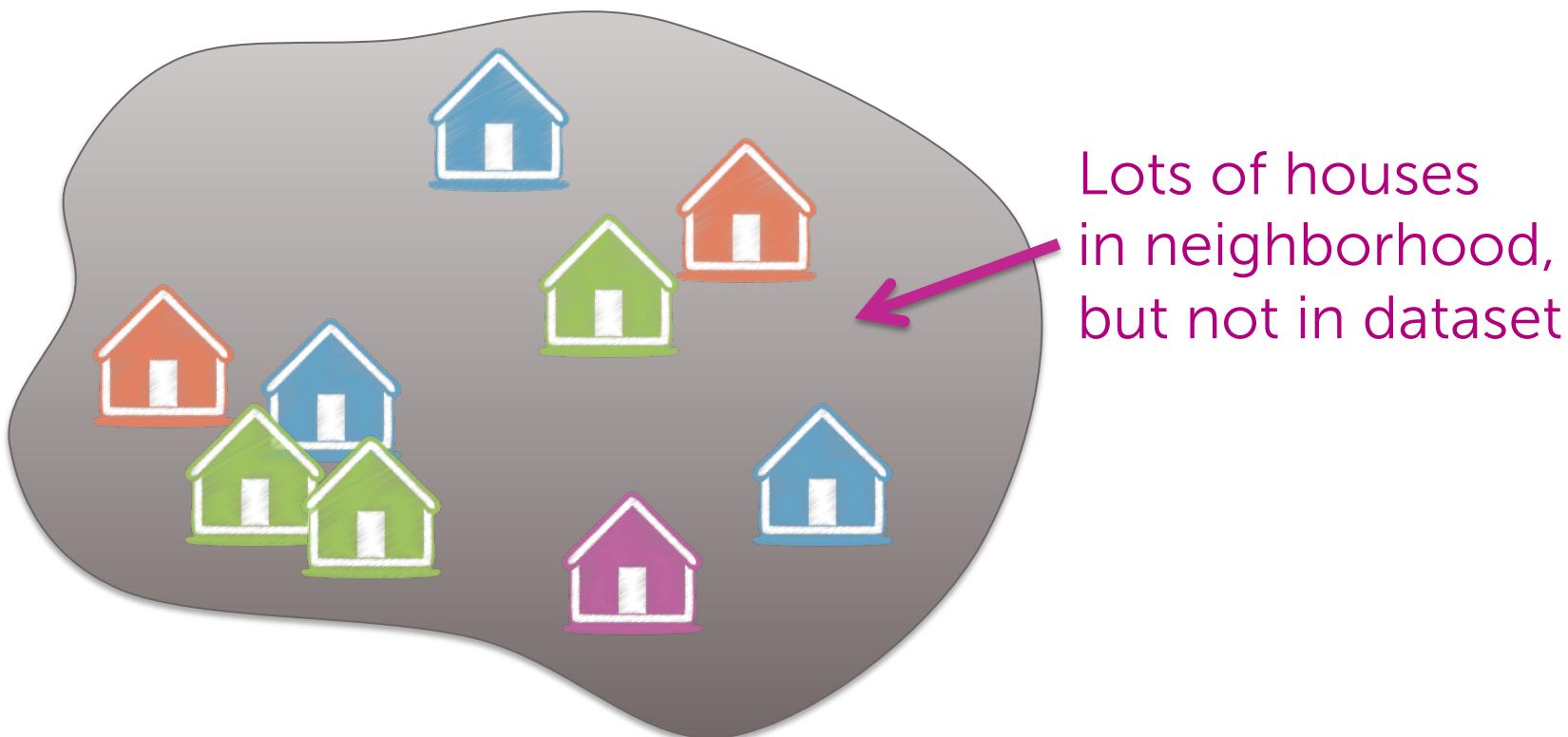
**Small training error  $\nRightarrow$  good predictions**  
unless training data includes everything you  
might ever see

# Assessing the loss

## Part 2: Generalization (true) error

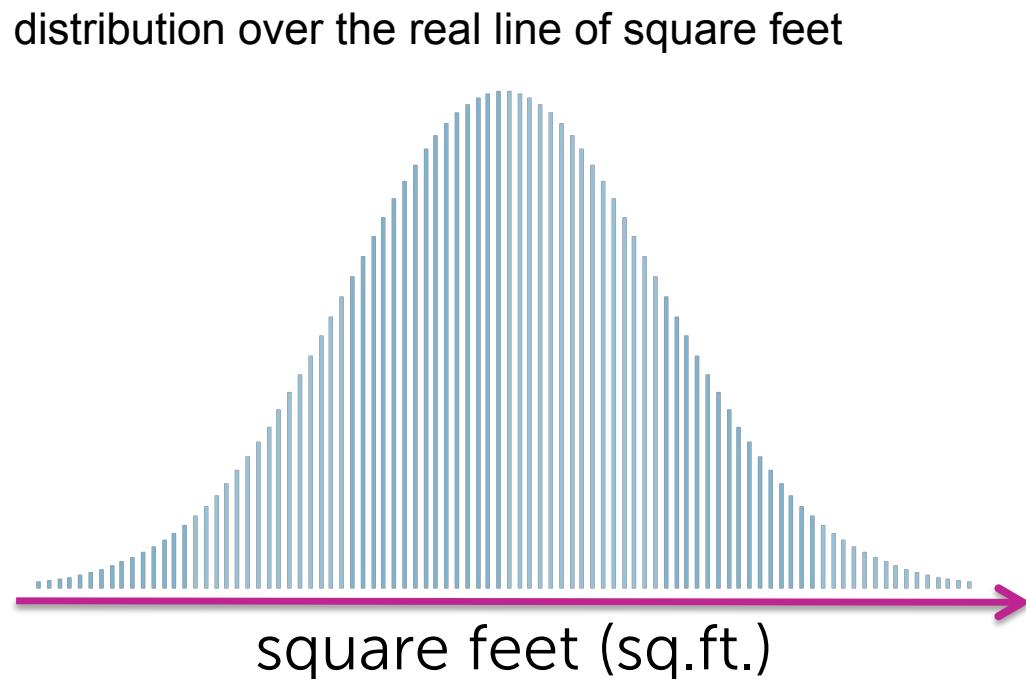
# Generalization error

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



# Distribution over houses

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



# Distribution over sales prices

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



# Generalization error definition

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

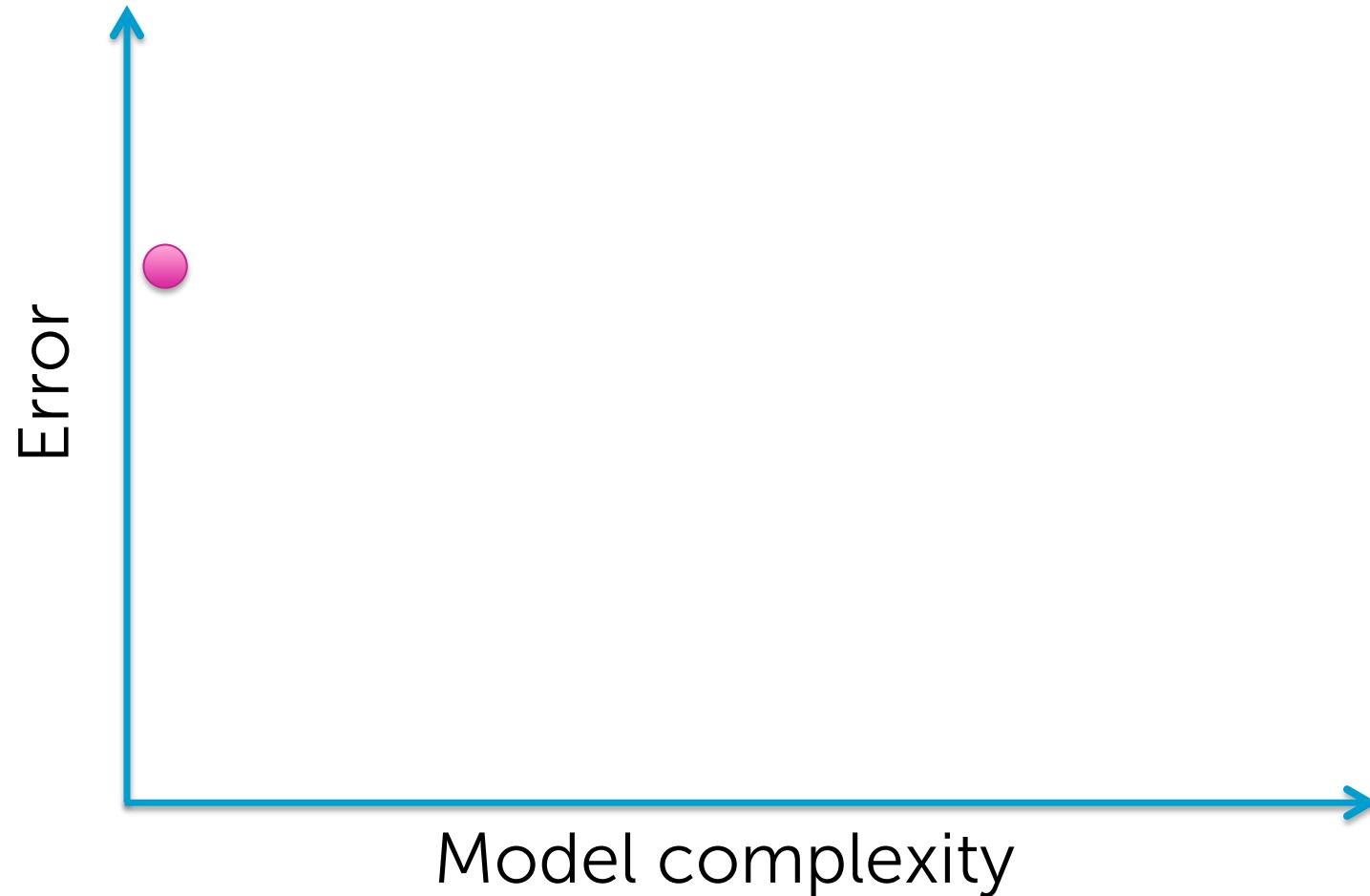
Formally:

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

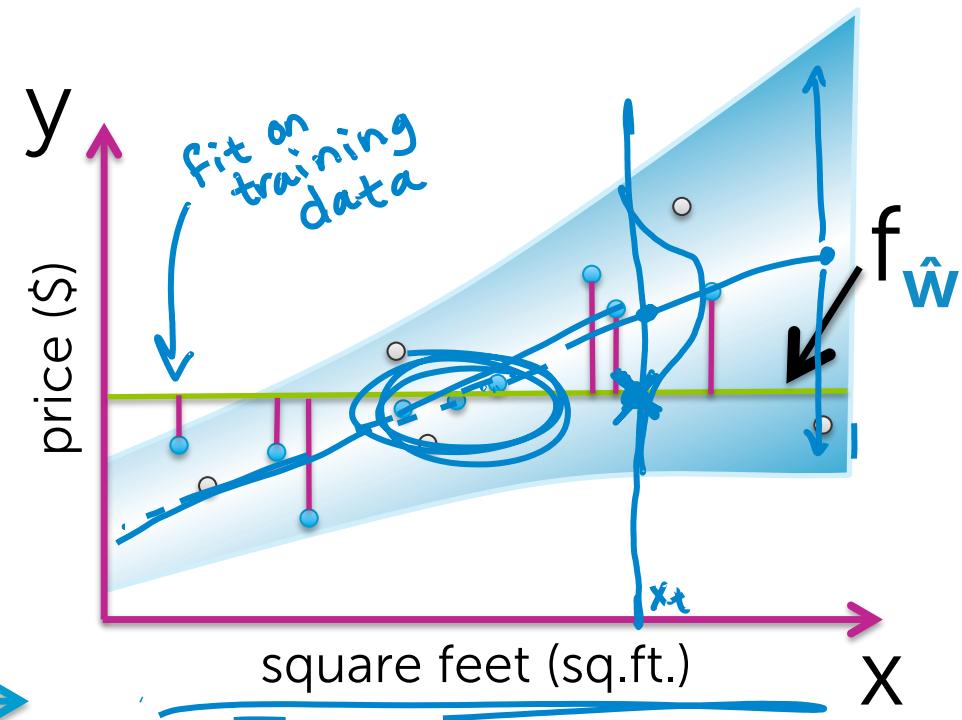
$$\text{generalization error} = E_{\mathbf{x},y}[\mathcal{L}(y, f_{\hat{\mathbf{w}}}(\mathbf{x}))]$$

fit using training data

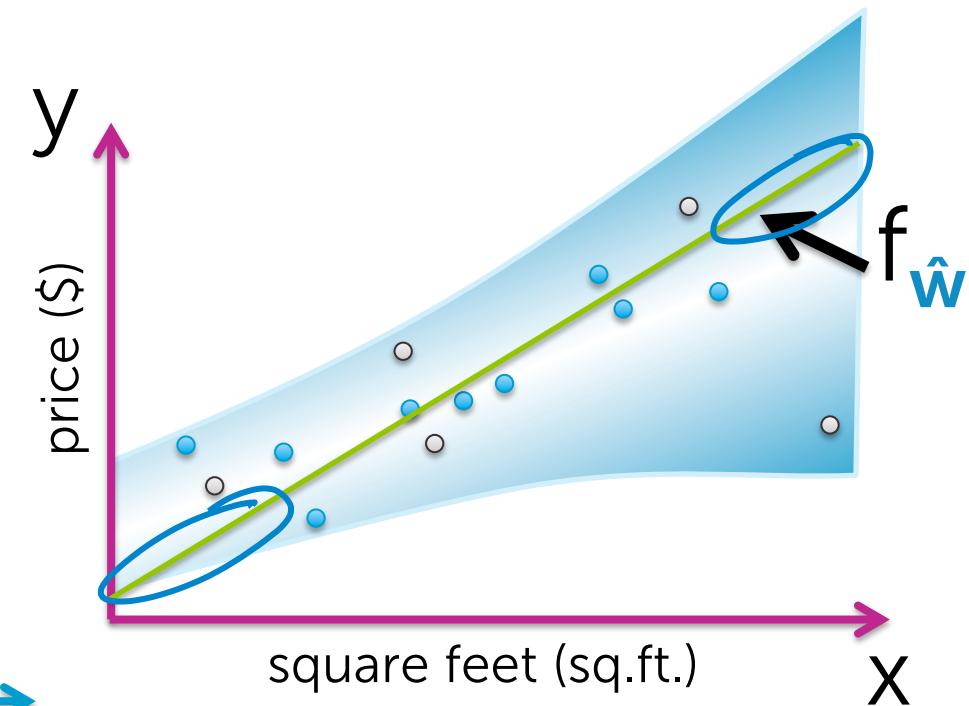
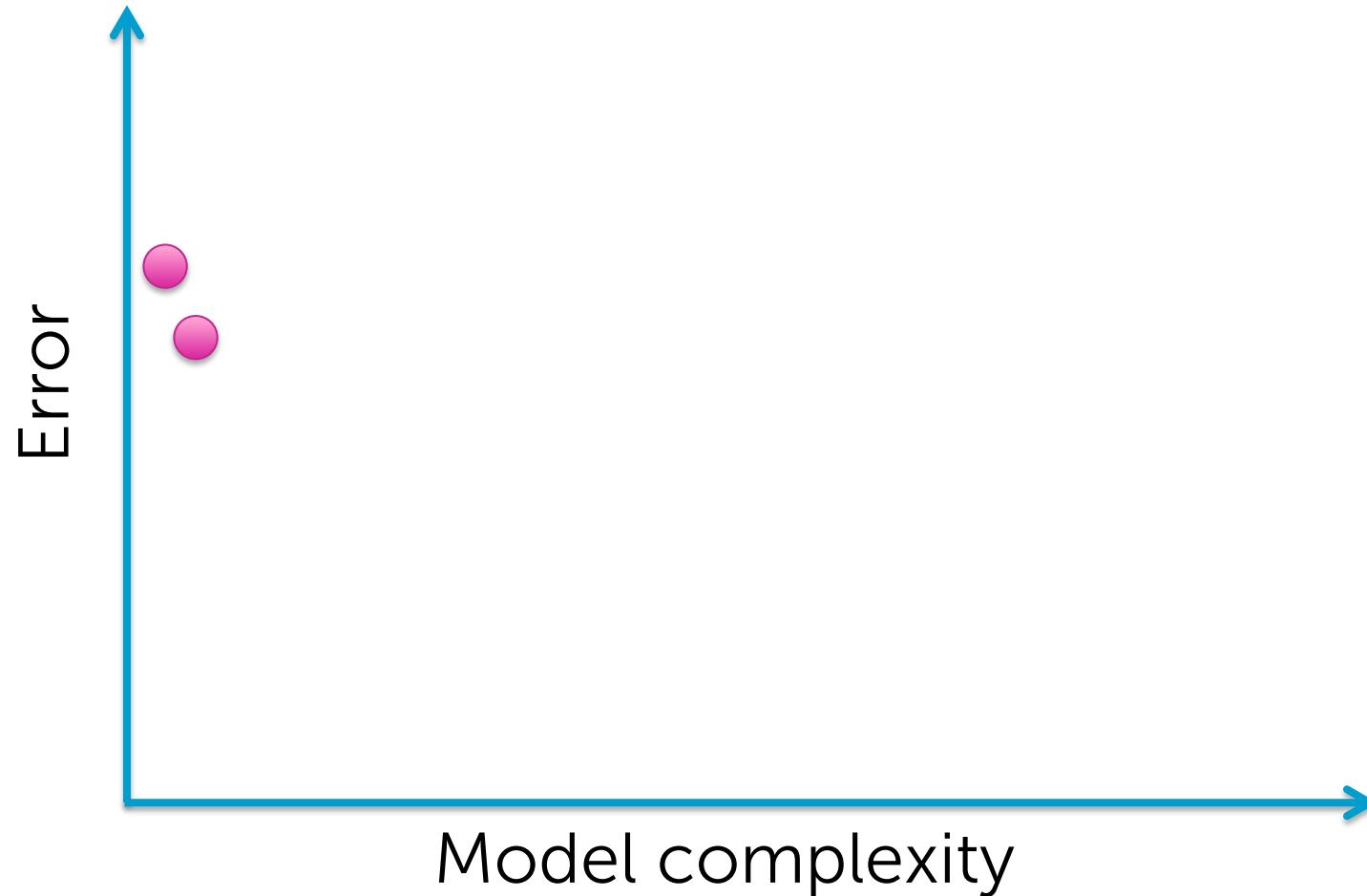
# Generalization error vs. model complexity



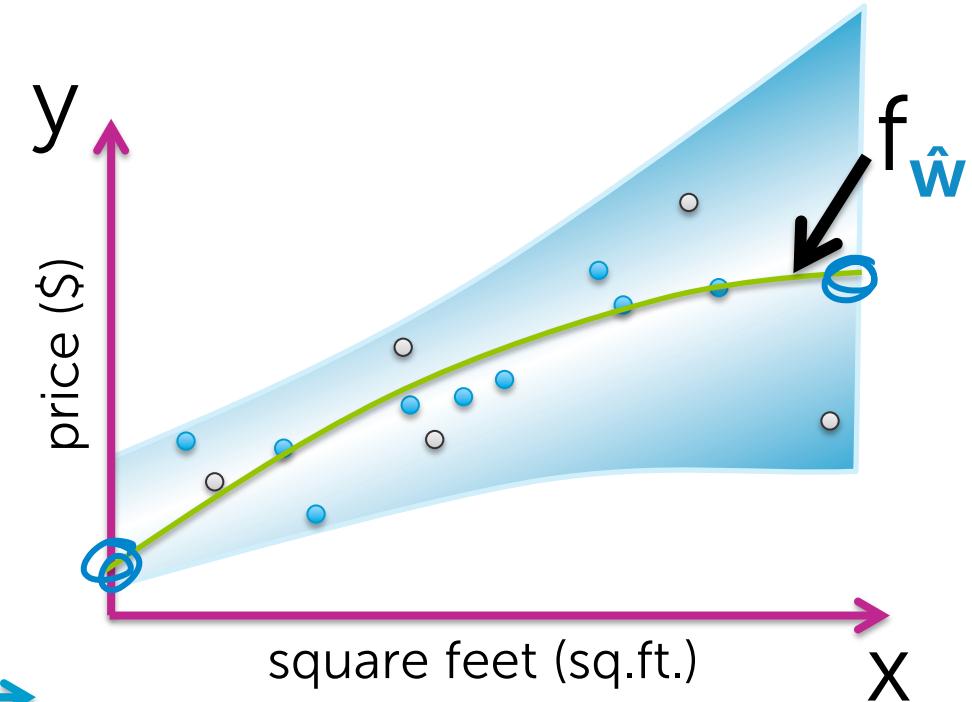
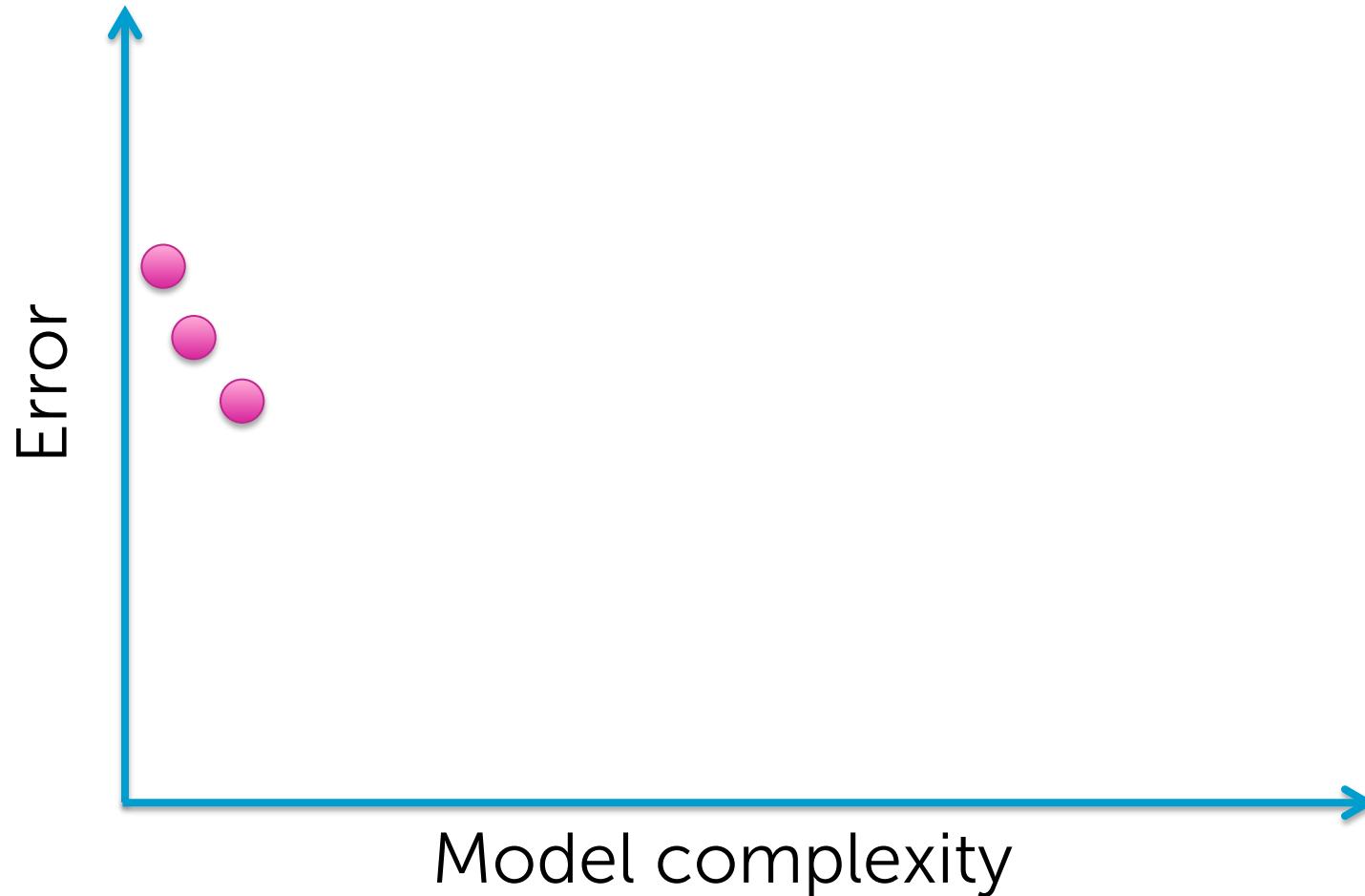
The blue region, having different gradation from white to darker blue, is the distribution of houses that I'm likely to see in my dataset.  
white - houses very likely to see  
blue - less likely sales prices given specific sq.ft.



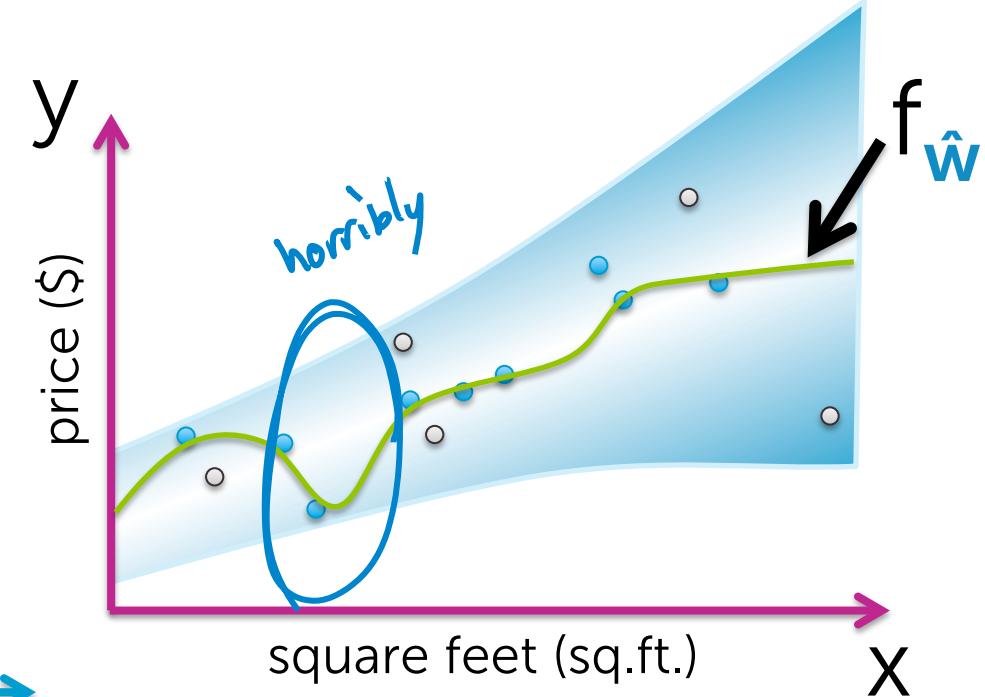
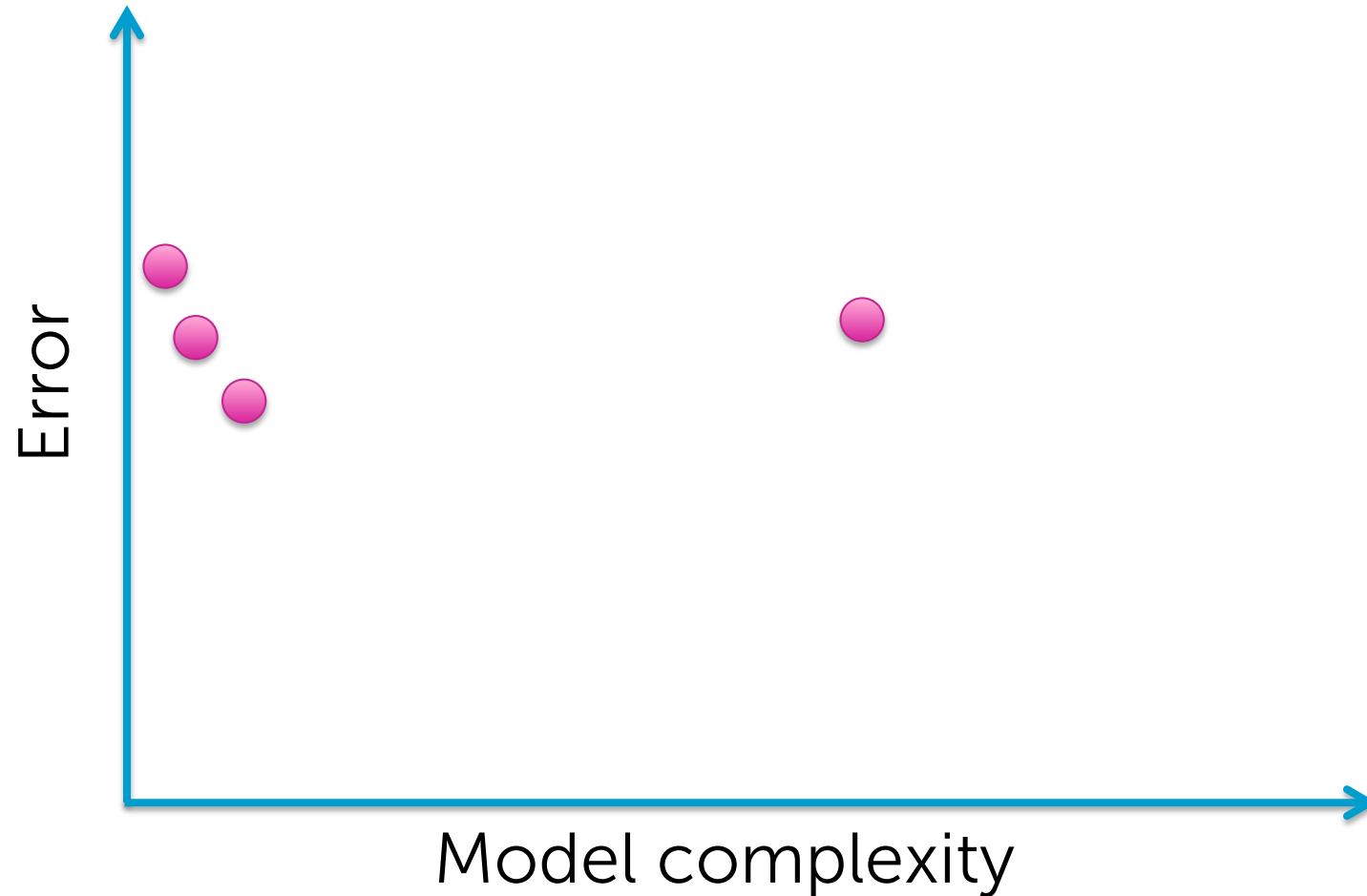
# Generalization error vs. model complexity



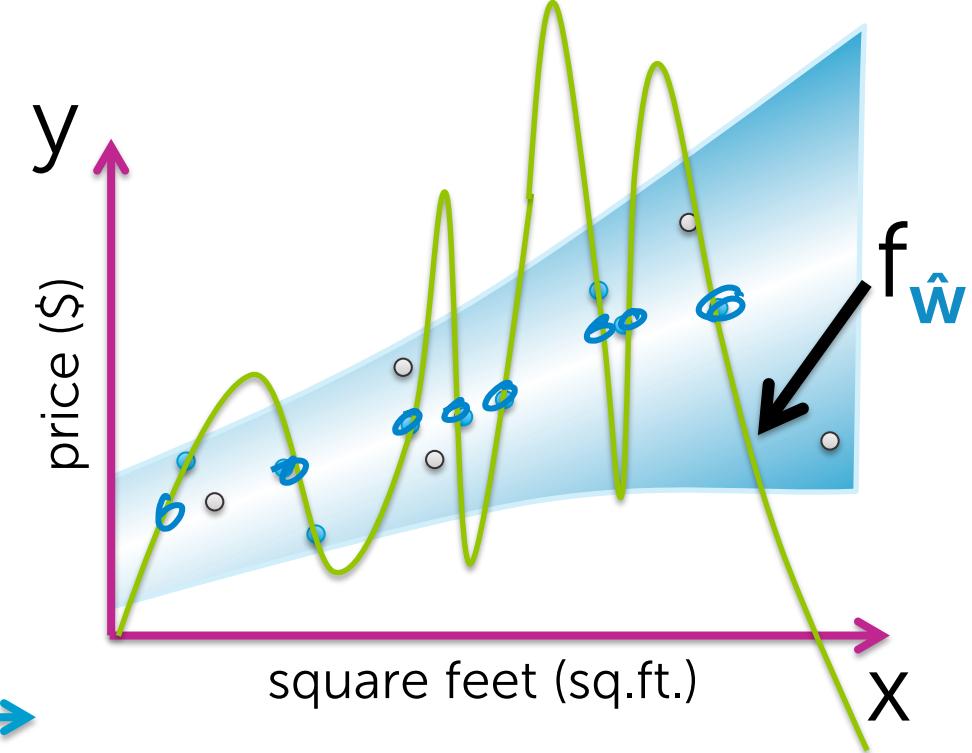
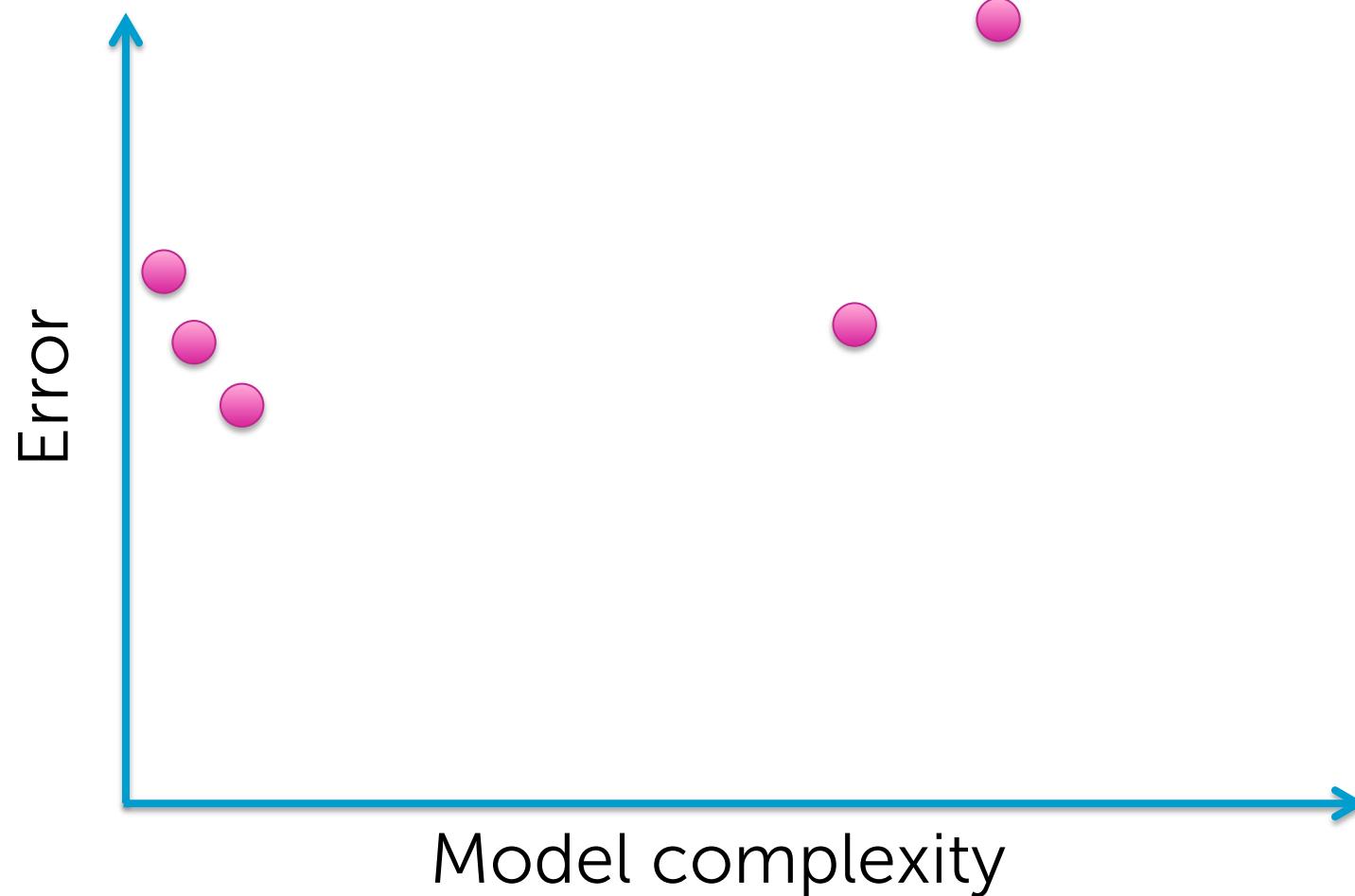
# Generalization error vs. model complexity



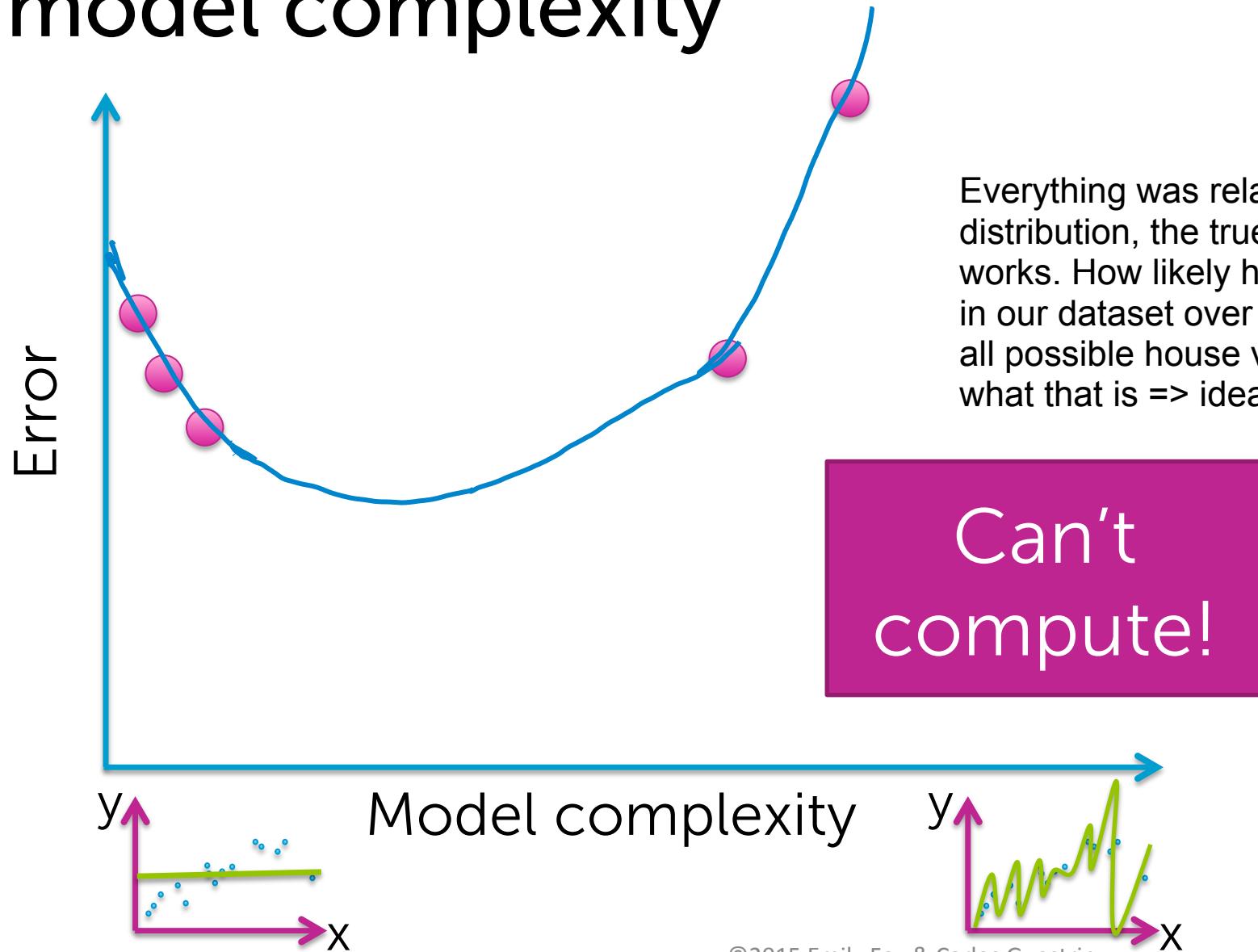
# Generalization error vs. model complexity



# Generalization error vs. model complexity



# Generalization error vs. model complexity



Everything was relative to this true distribution, the true way in which the world works. How likely houses are to appear in our dataset over all possible sq.ft. and all possible house values. We don't know what that is => ideal picture.

# Assessing the loss

## Part 3: Test error

# Approximating generalization error

Wanted estimate of loss  
over all possible (.house,\$) pairs



Approximate by looking at houses not in training set

# Forming a test set

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



Training set



Test set



# Forming a test set

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



Proxy for “everything you  
might see”

Test set



# Compute test error

Test error

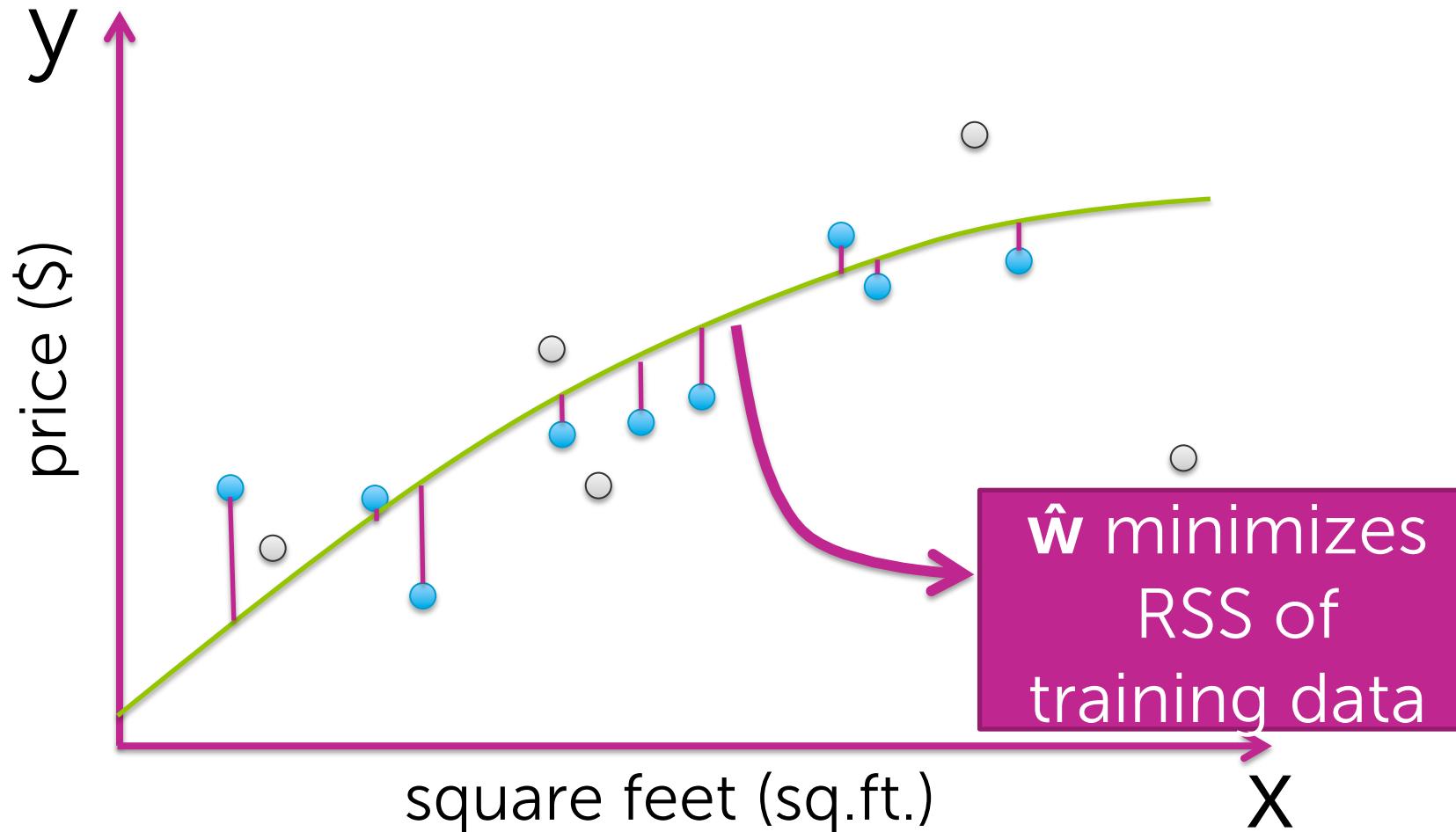
= avg. loss on houses in **test set**

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

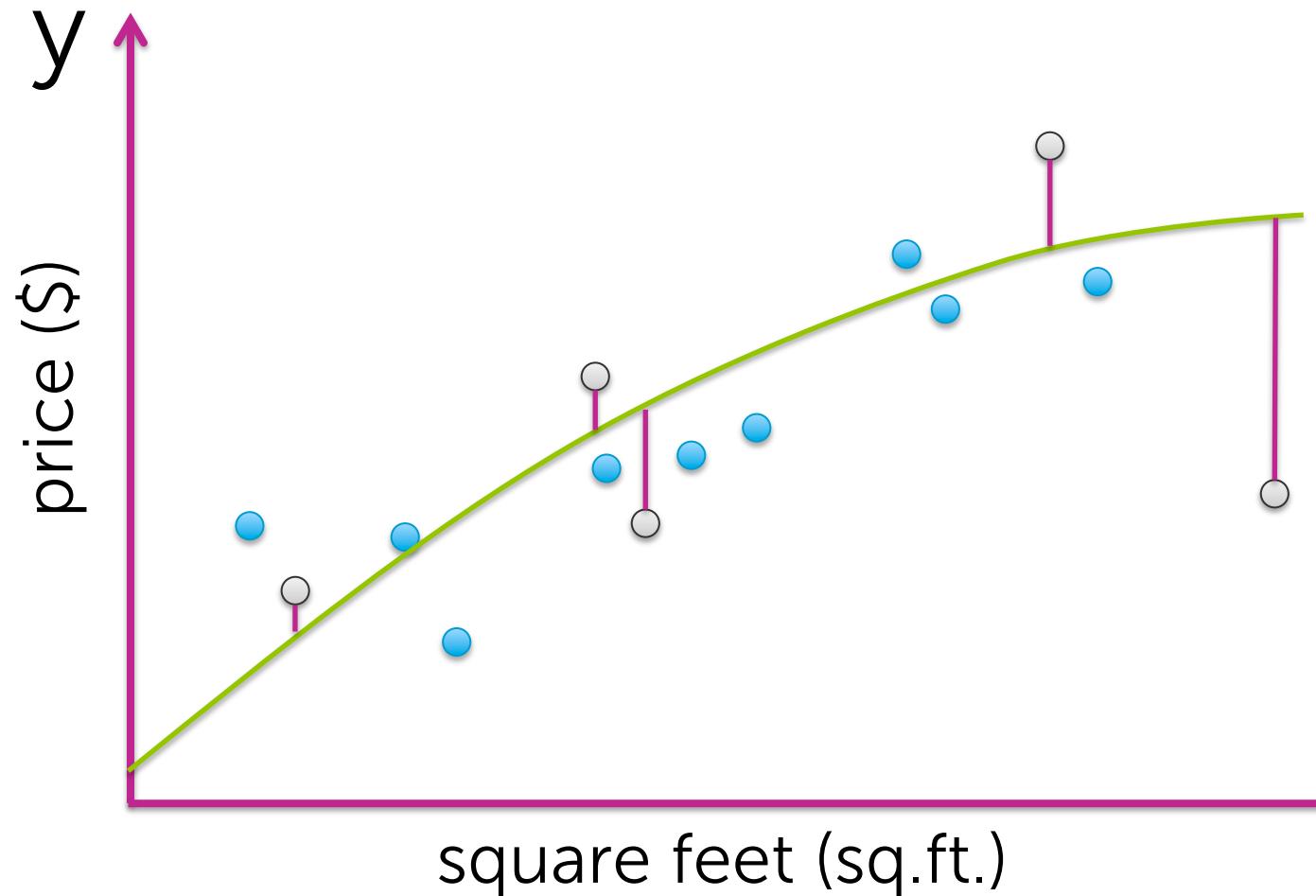
  
# test points    fit using **training data**

Fitted function **has never seen  
test data!**

# Example: As before, fit quadratic to training data

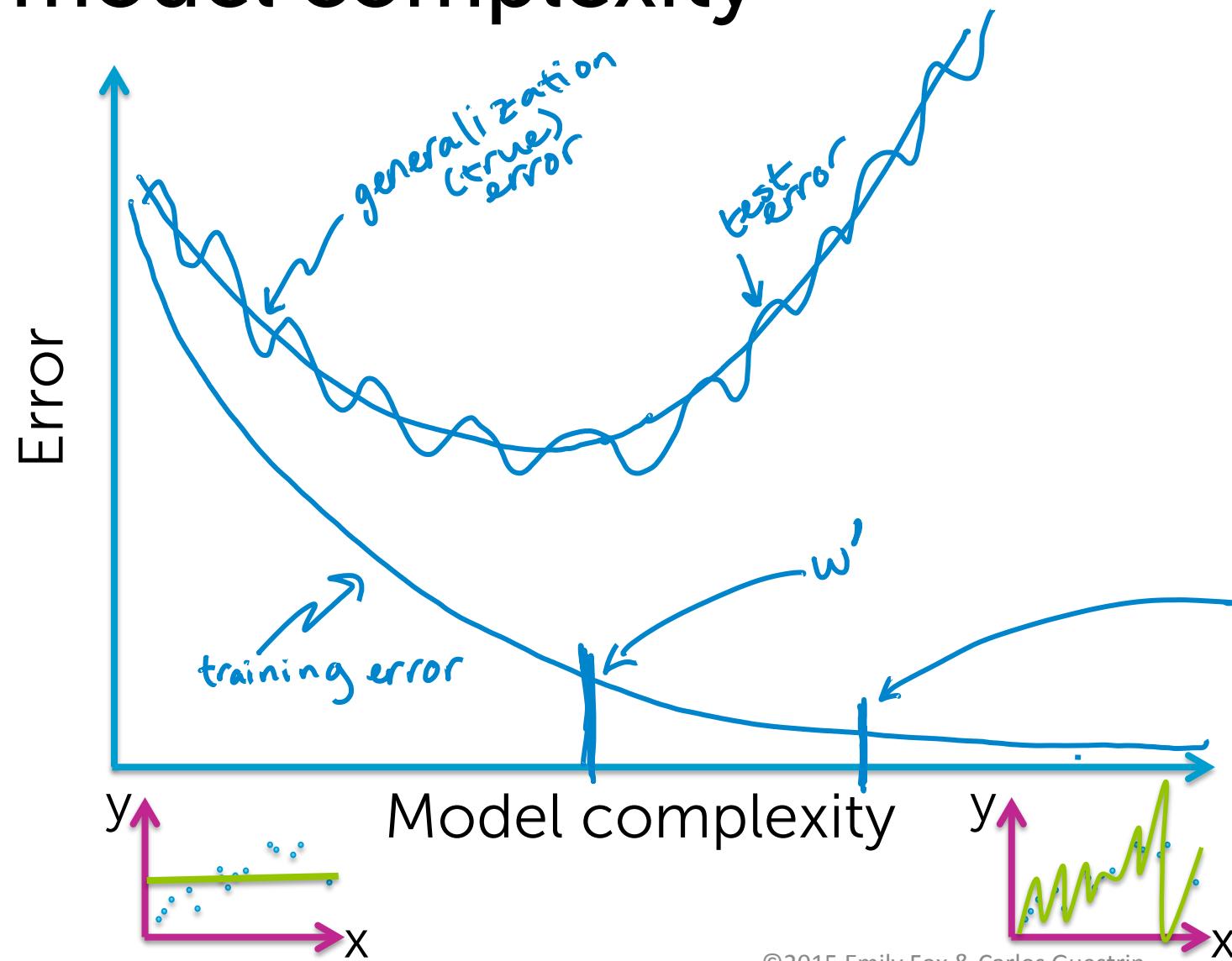


Example: As before,  
use squared error loss  $(y - f_{\hat{w}}(x))^2$



$$\begin{aligned} \text{Test error } (\hat{w}) &= 1/N * \\ & [(\$_{\text{test 1}} - f_{\hat{w}}(\text{sq.ft.}_{\text{test 1}}))^2 \\ & + (\$_{\text{test 2}} - f_{\hat{w}}(\text{sq.ft.}_{\text{test 2}}))^2 \\ & + (\$_{\text{test 3}} - f_{\hat{w}}(\text{sq.ft.}_{\text{test 3}}))^2 \\ & + \dots \text{ include all} \\ & \quad \text{test houses}] \end{aligned}$$

# Training, true, & test error vs. model complexity



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 split

# Training/test splits



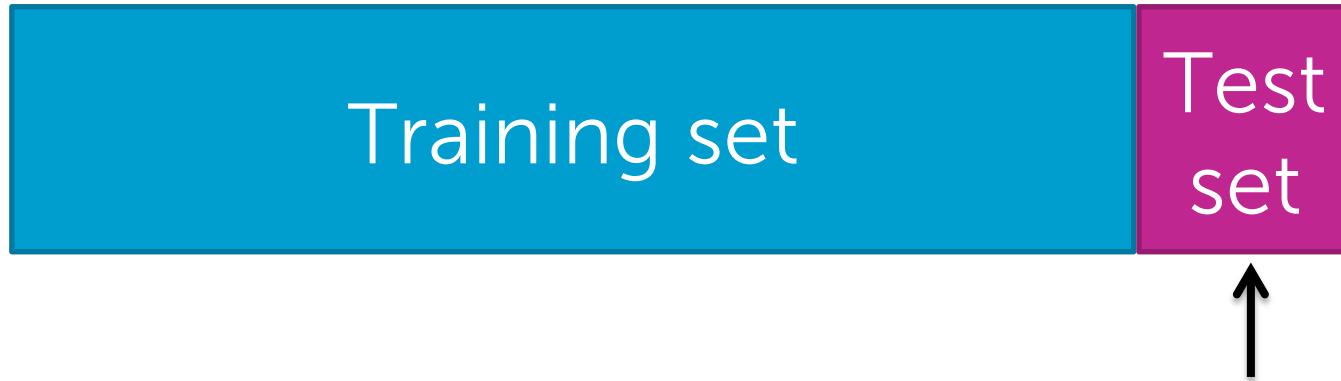
how many? vs. how many?

# Training/test splits



Too few  $\rightarrow \hat{w}$  poorly estimated

# Training/test splits



Too few → test error bad approximation  
of generalization error

# Training/test splits



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...)

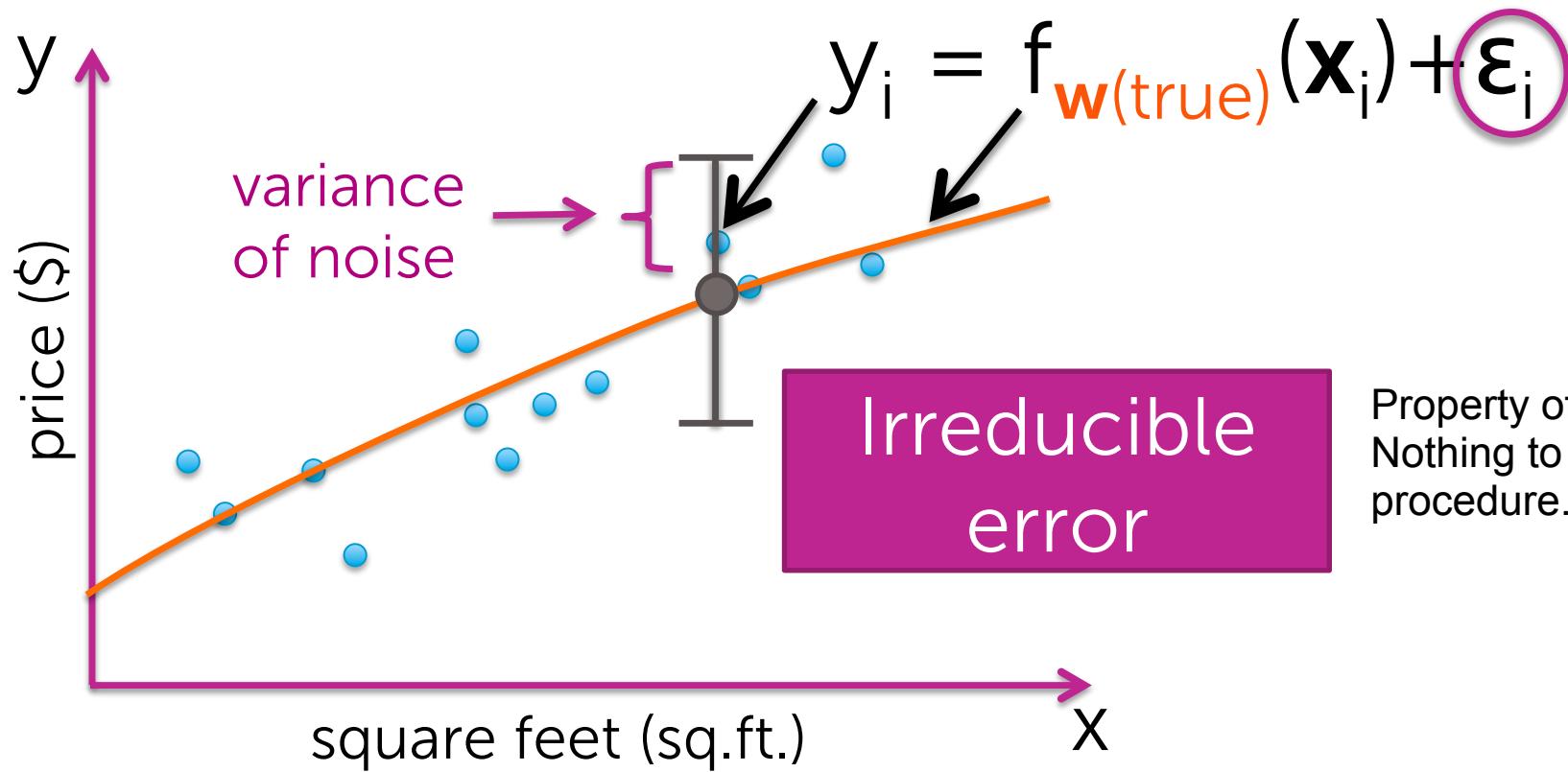
# 3 sources of error + the bias-variance tradeoff

# 3 sources of error

In forming predictions, there  
are 3 sources of error:

1. Noise
2. Bias
3. Variance

# Data inherently noisy

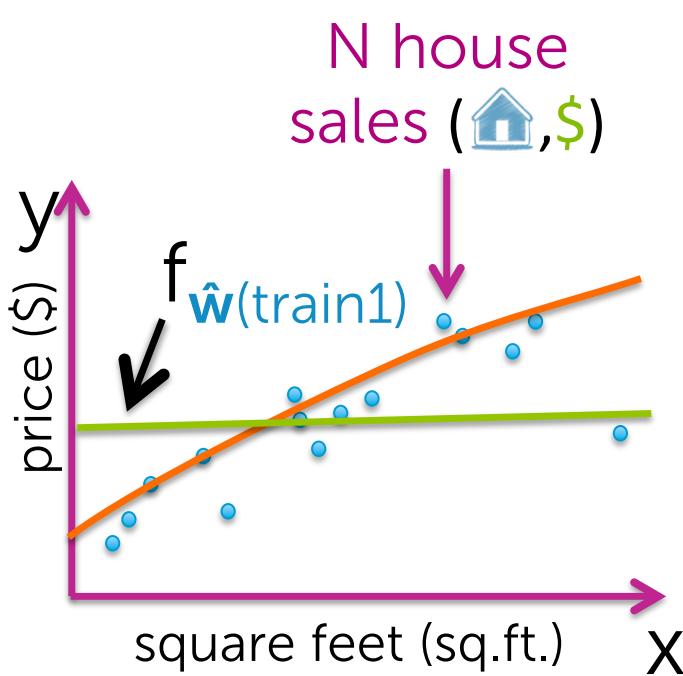


Property of data, we don't have control over this.  
Nothing to do with our model or estimation procedure.

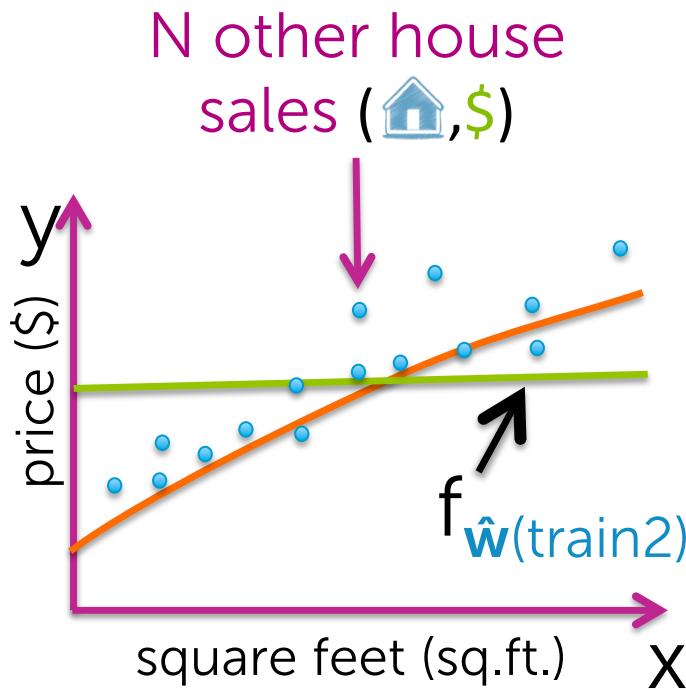
# Bias contribution

Assume we fit a constant function

Thinking about bias, it's intuitive to start with a very simple model of just a constant function.



Tend to draw points below the true relationship, so that houses in dataset happened to have values less than what the world specifies as typical.

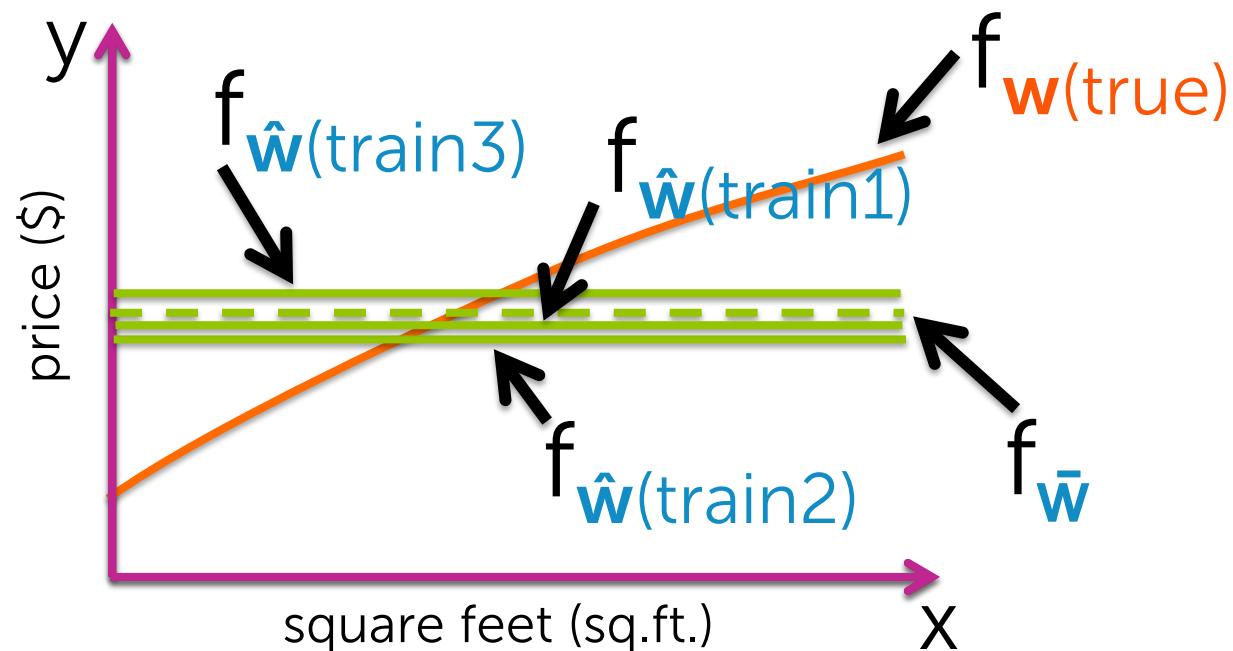


Tend to draw points above the line

Train1 and train2 are extremely different datasets, but fits are pretty similar.

# Bias contribution

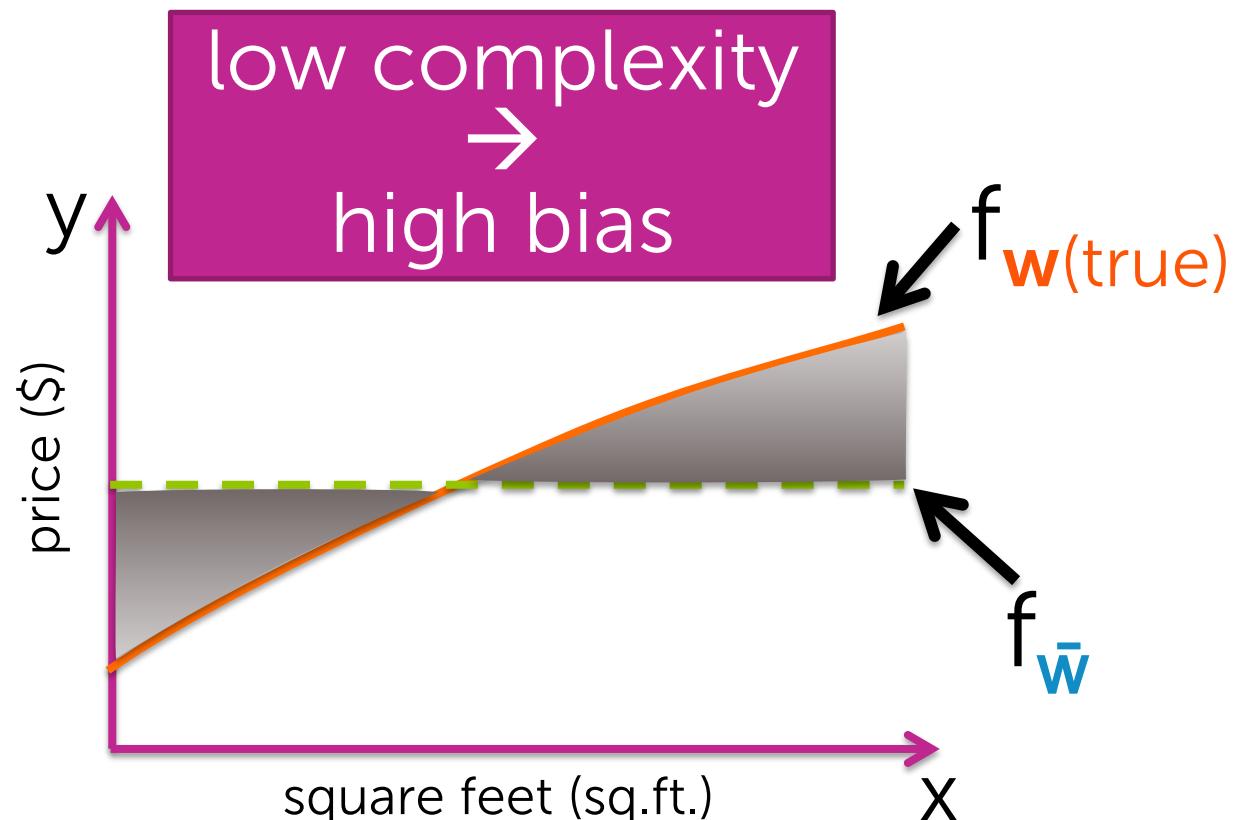
Over all possible size N training sets,  
what do I expect my fit to be?



# Bias contribution

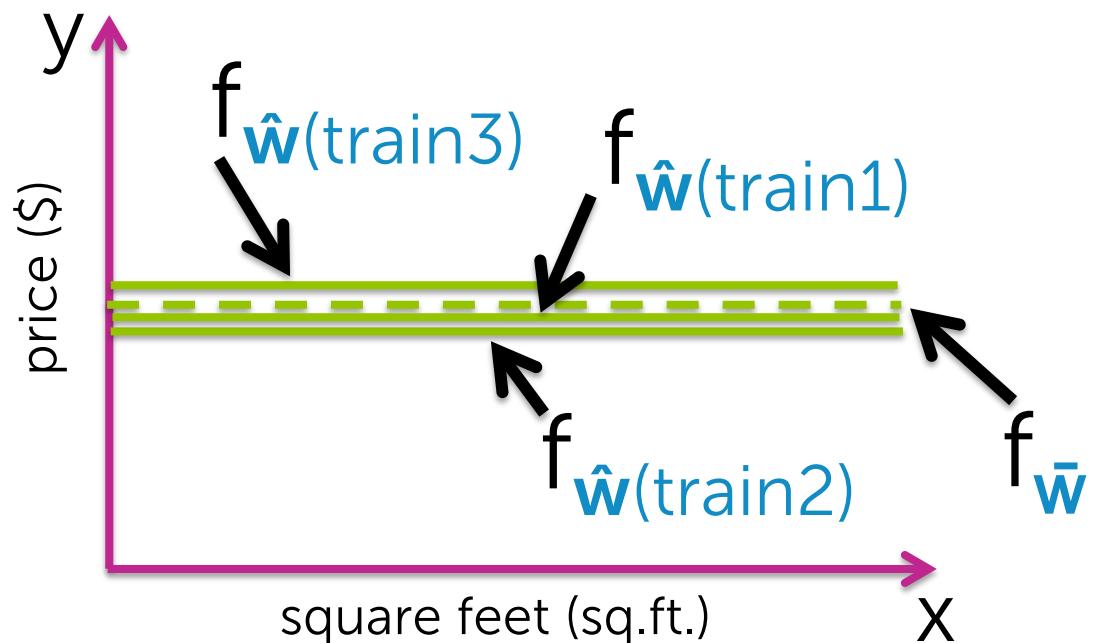
$$\text{Bias}(\mathbf{x}) = f_{\mathbf{w}(\text{true})}(\mathbf{x}) - f_{\bar{\mathbf{w}}}(\mathbf{x}) \leftarrow$$

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



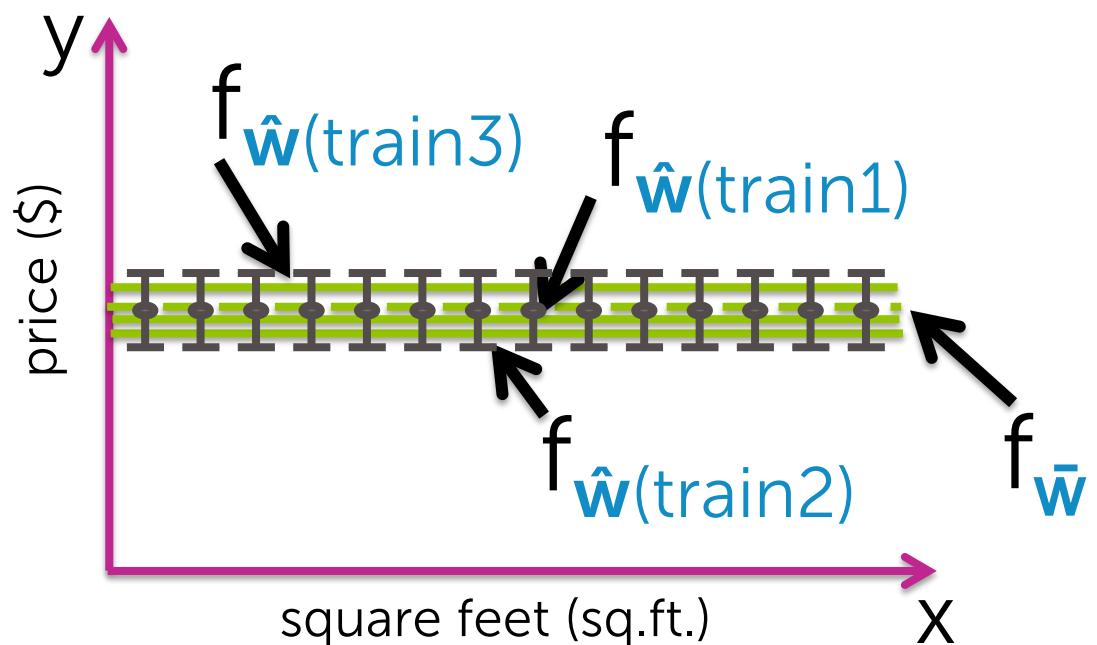
# Variance contribution

How much do specific fits vary from the expected fit?



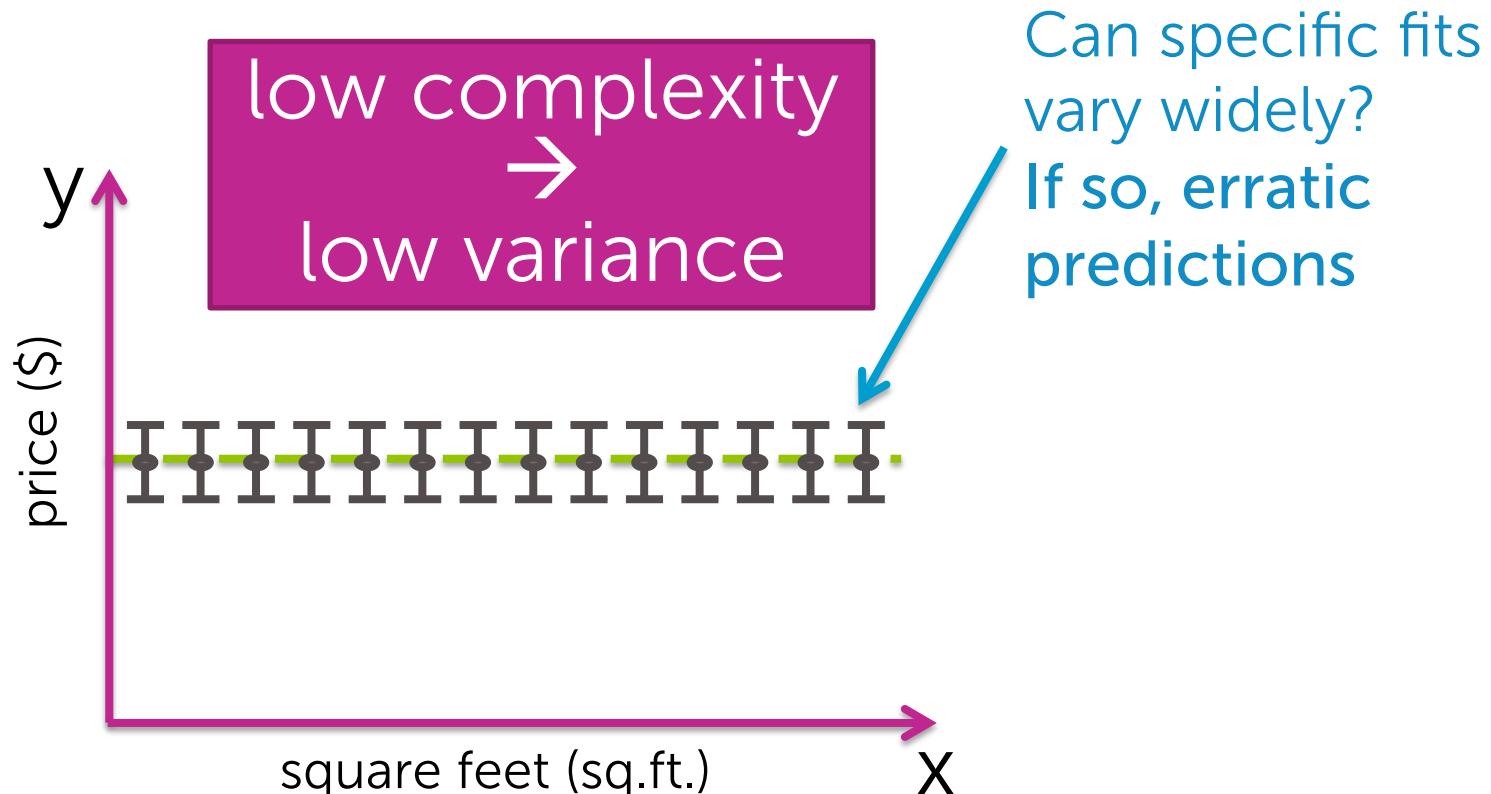
# Variance contribution

How much do specific fits vary from the expected fit?



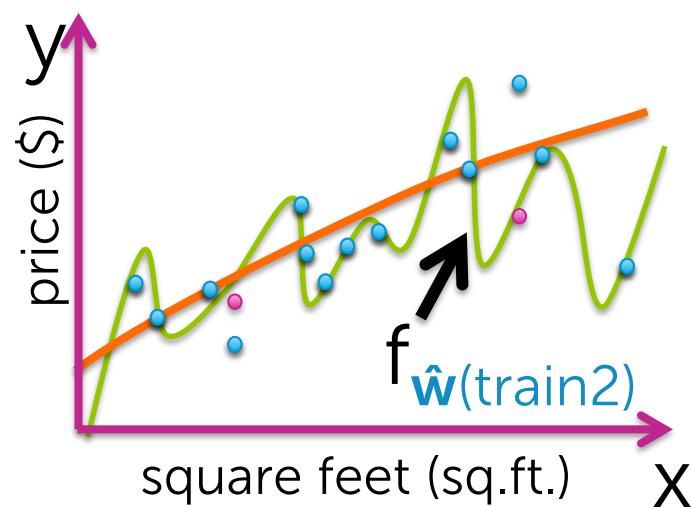
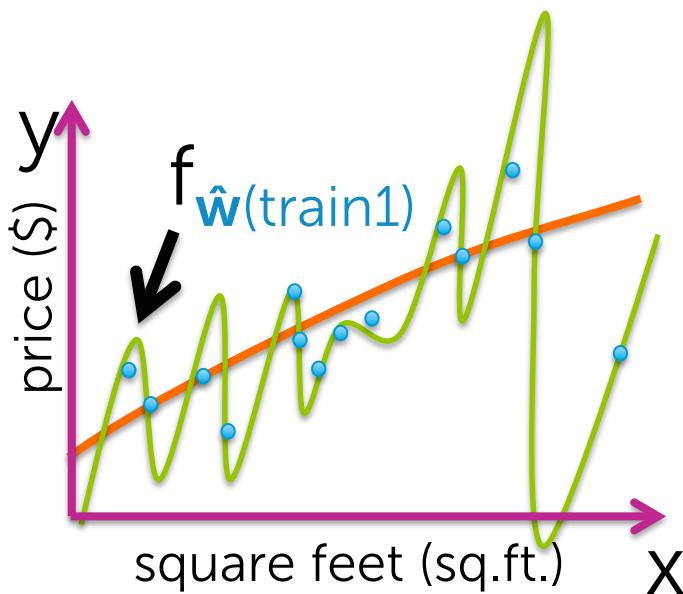
# Variance contribution

How much do specific fits vary from the expected fit?



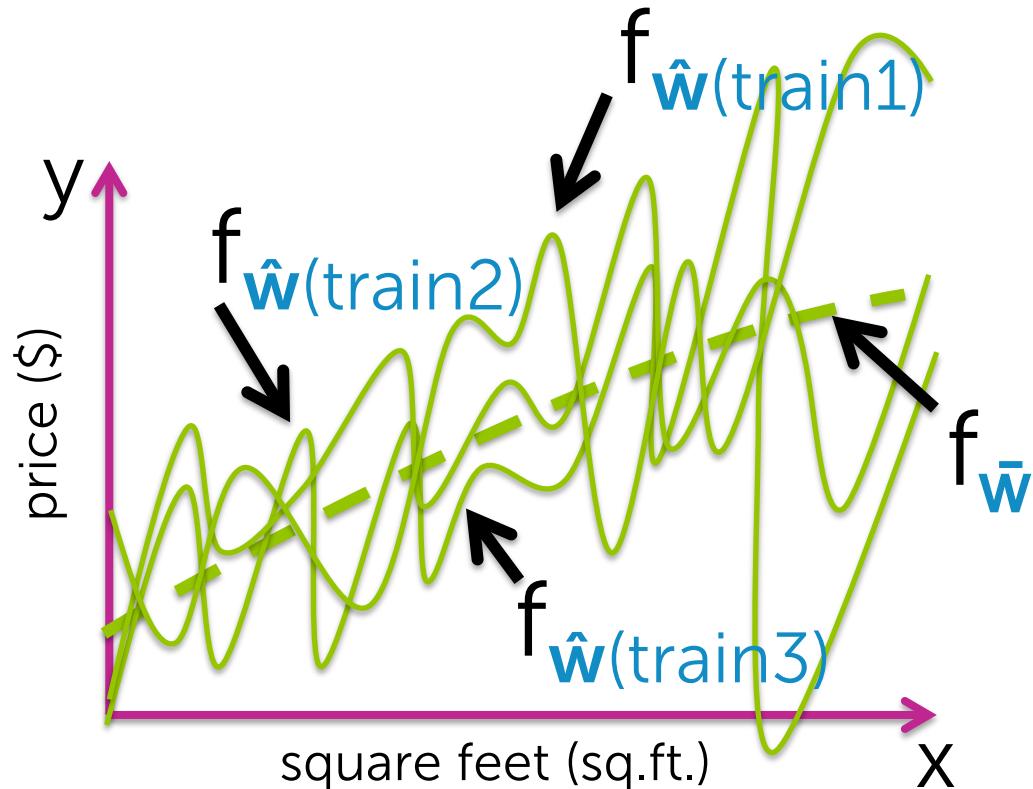
# Variance of high-complexity models

Assume we fit a high-order polynomial

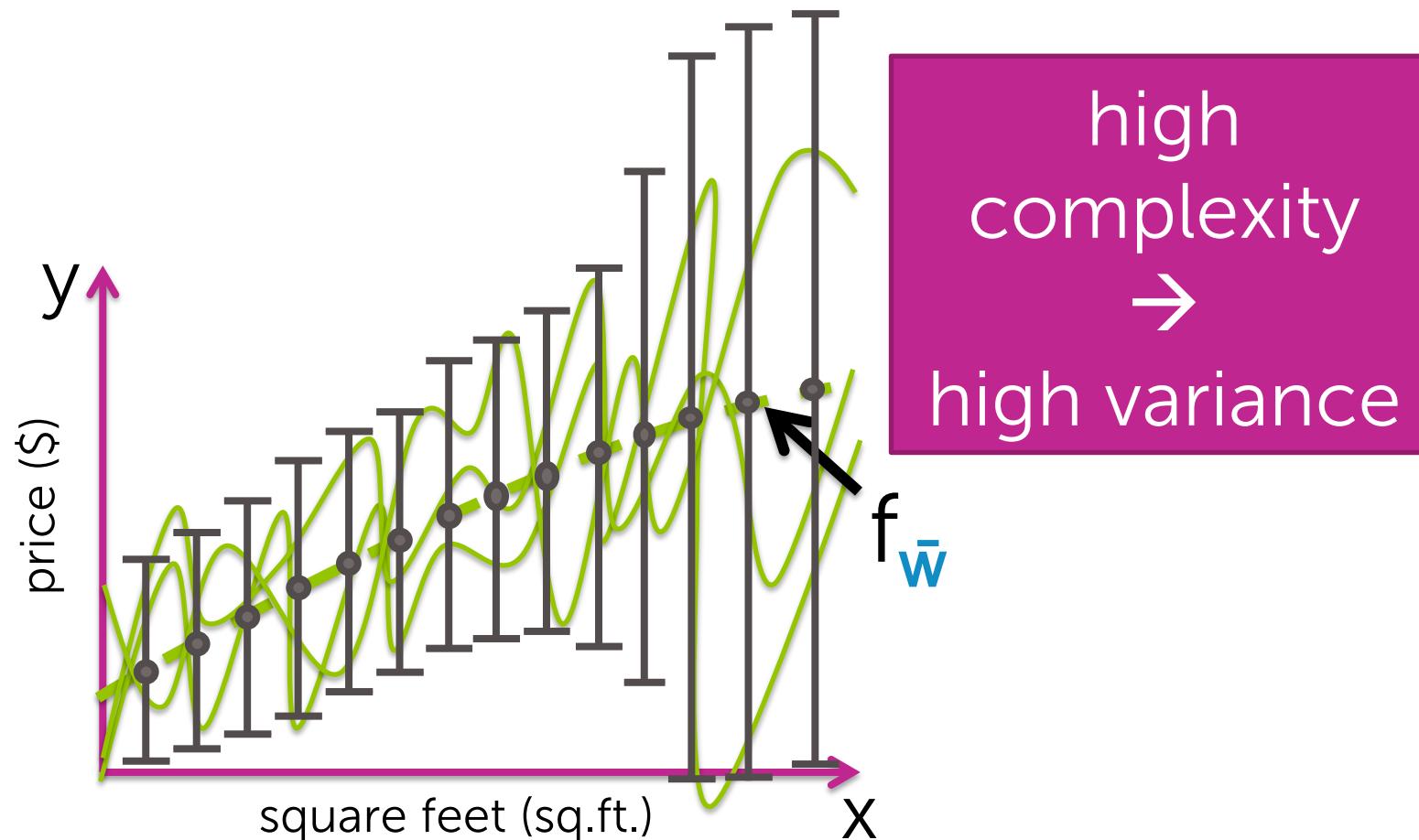


# Variance of high-complexity models

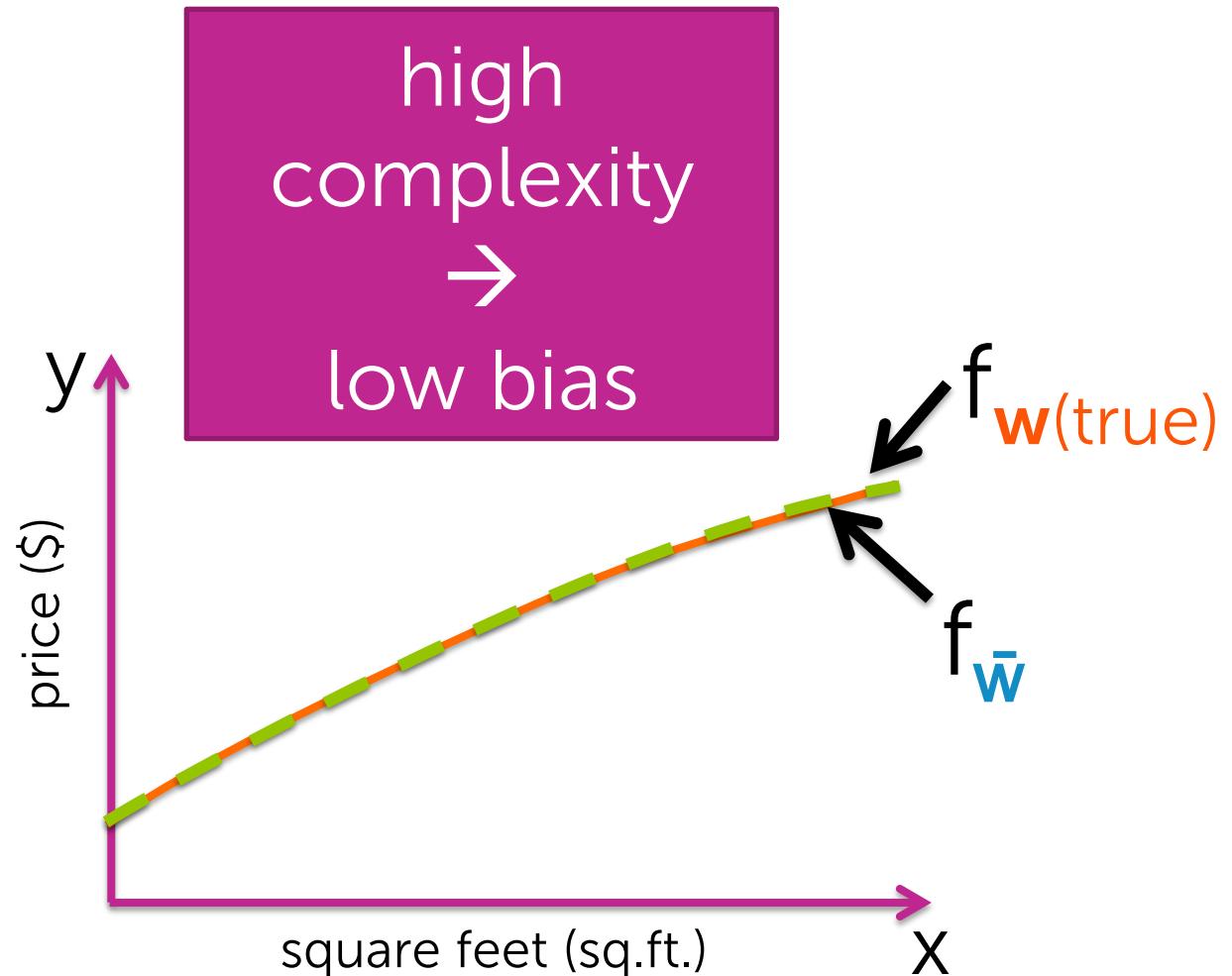
Assume we fit a high-order polynomial



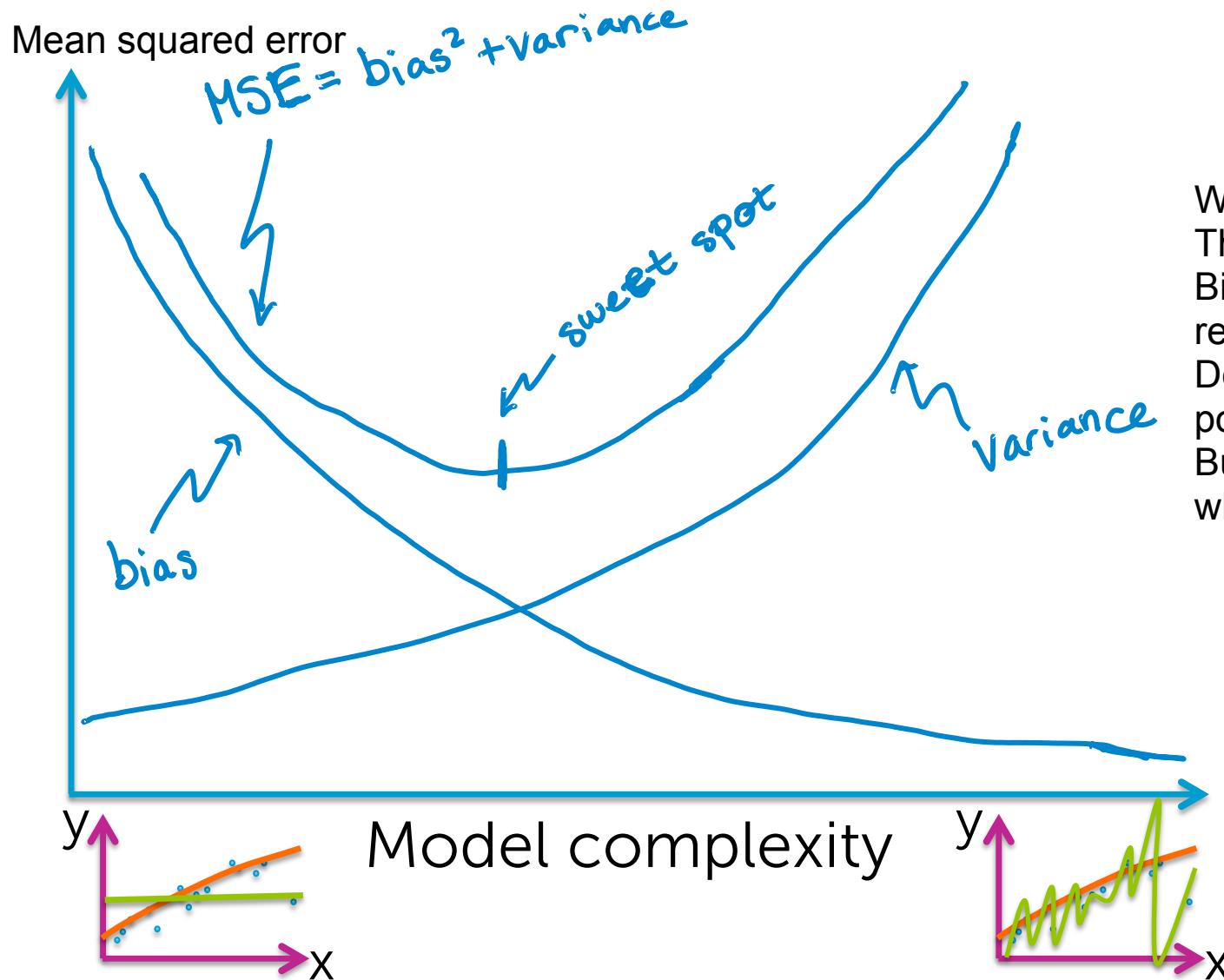
# Variance of high-complexity models



# Bias of high-complexity models



# Bias-variance tradeoff



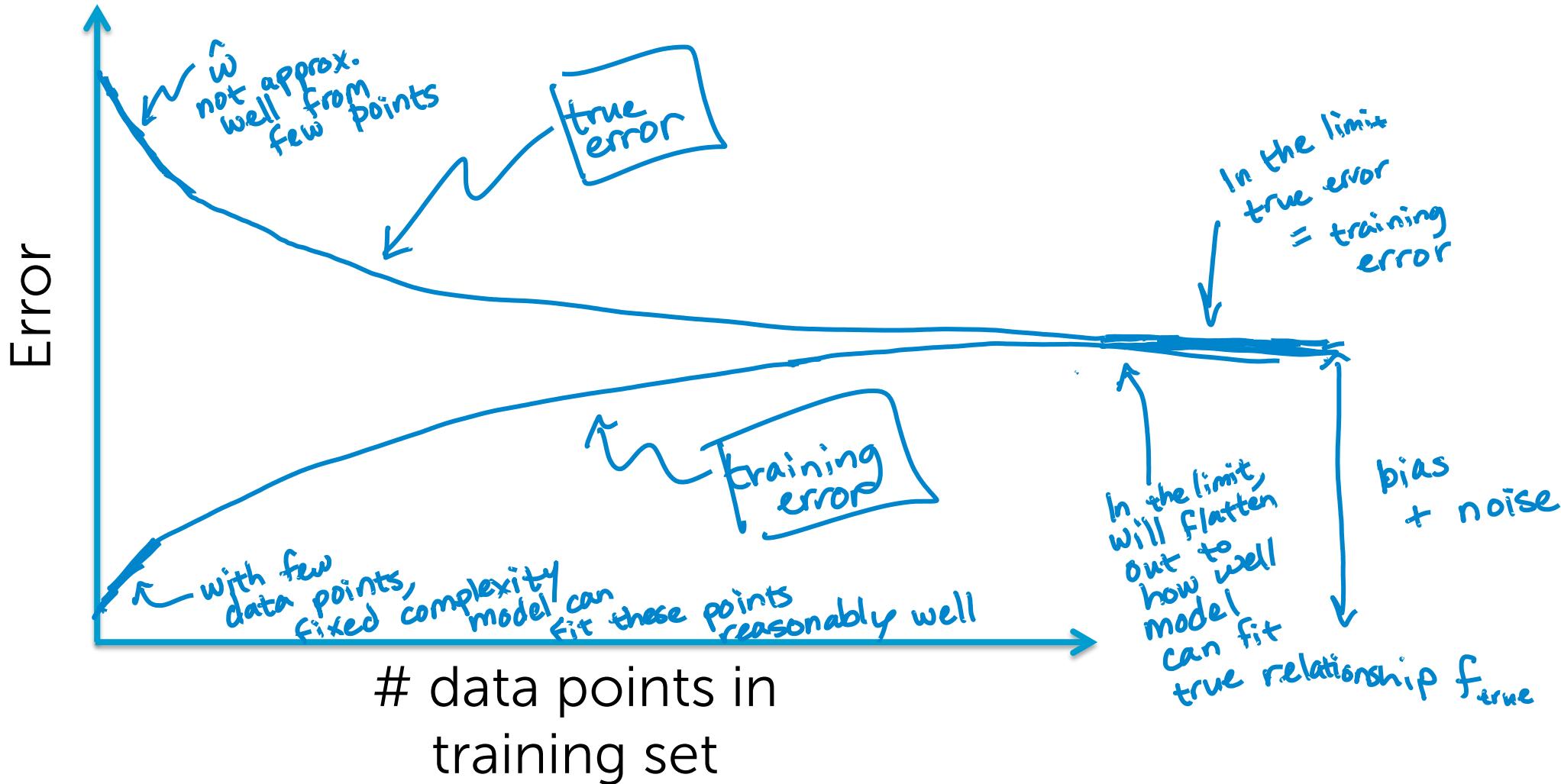
Why cannot compute?

They are defined in terms of the true function.  
Bias is defined very explicitly in terms of the relationship relative to the true function.  
Defining variance, we have to average over all possible datasets, same was true for bias too.  
But all possible datasets of size  $n$ , don't know what that is.

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

# Error vs. amount of data

for a fixed model complexity



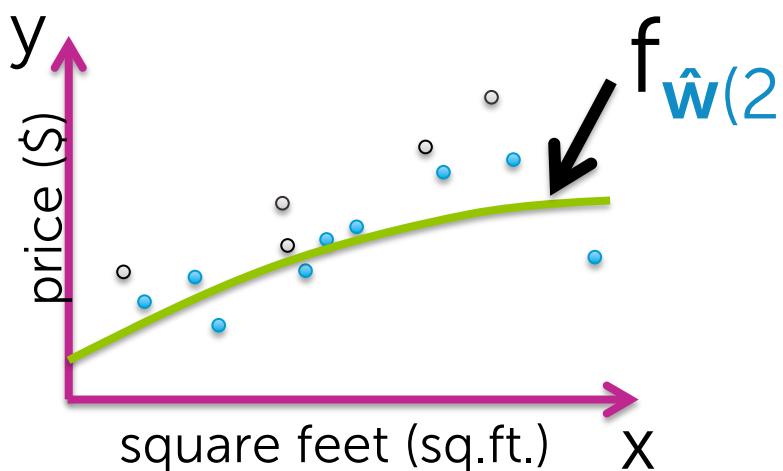
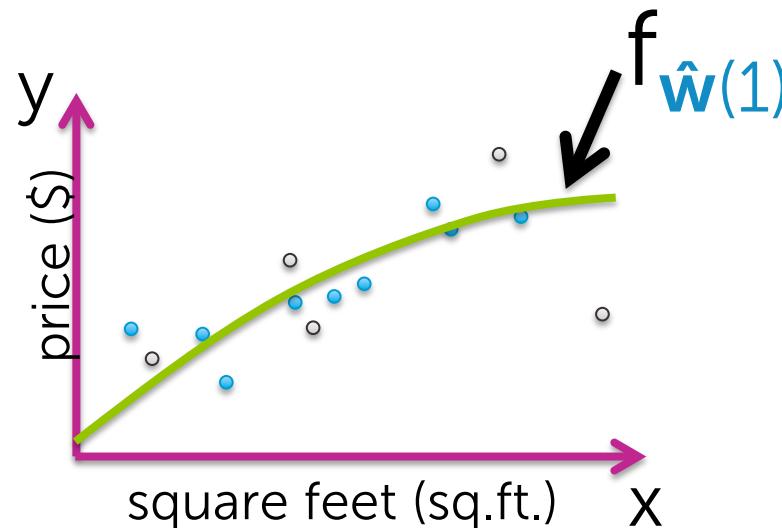
More in depth on the  
3 sources of errors...

**OPTIONAL**

# Accounting for training set randomness

Training set was just a random sample of  $N$  houses sold

What if  $N$  other houses had been sold and recorded?

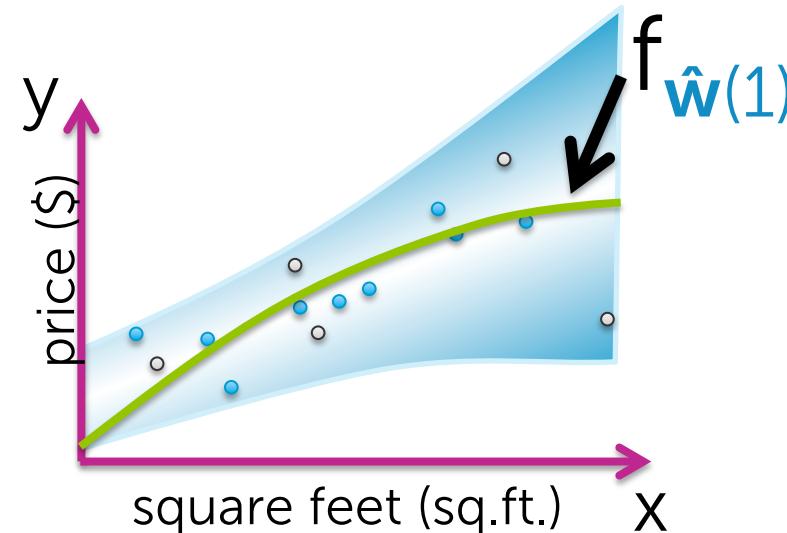


# Accounting for training set randomness

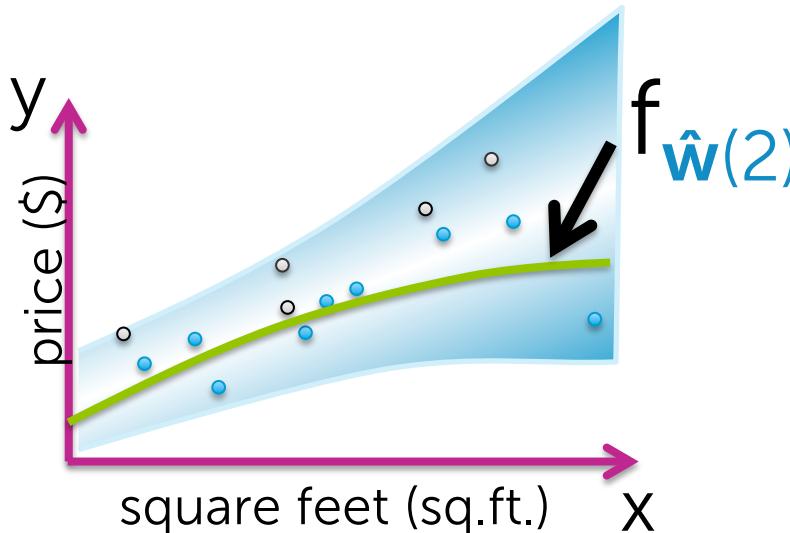
Training set was just a random sample of  $N$  houses sold

What if  $N$  other houses had been sold and recorded?

generalization error of  $\hat{w}(1)$



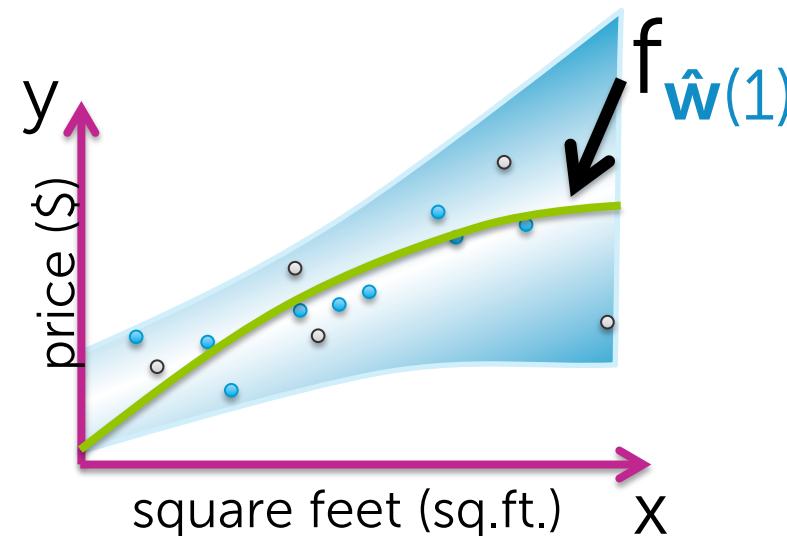
generalization error of  $\hat{w}(2)$



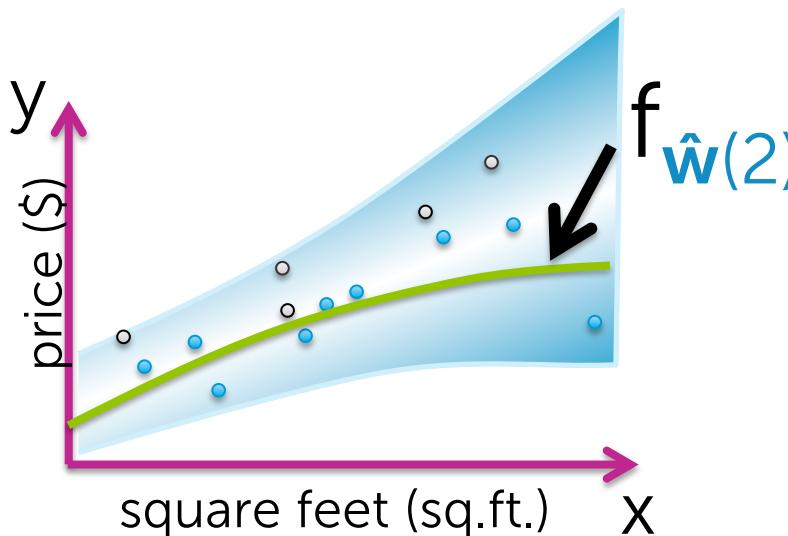
# Accounting for training set randomness

Ideally, want performance averaged over all possible training sets of size N

generalization error of  $\hat{w}(1)$



generalization error of  $\hat{w}(2)$

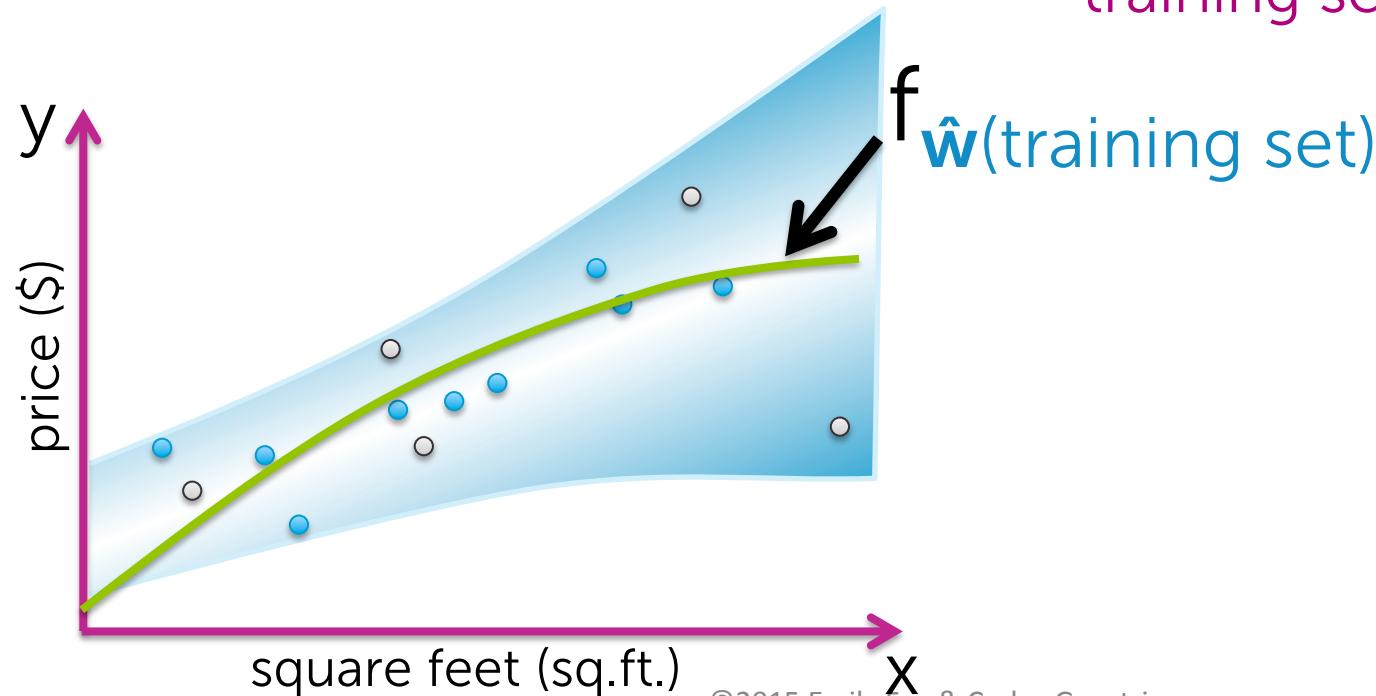


# Expected prediction error

$E_{\text{training set}}[\text{generalization error of } \hat{w}(\text{training set})]$

↑  
averaging over all training sets  
(weighted by how likely each is)

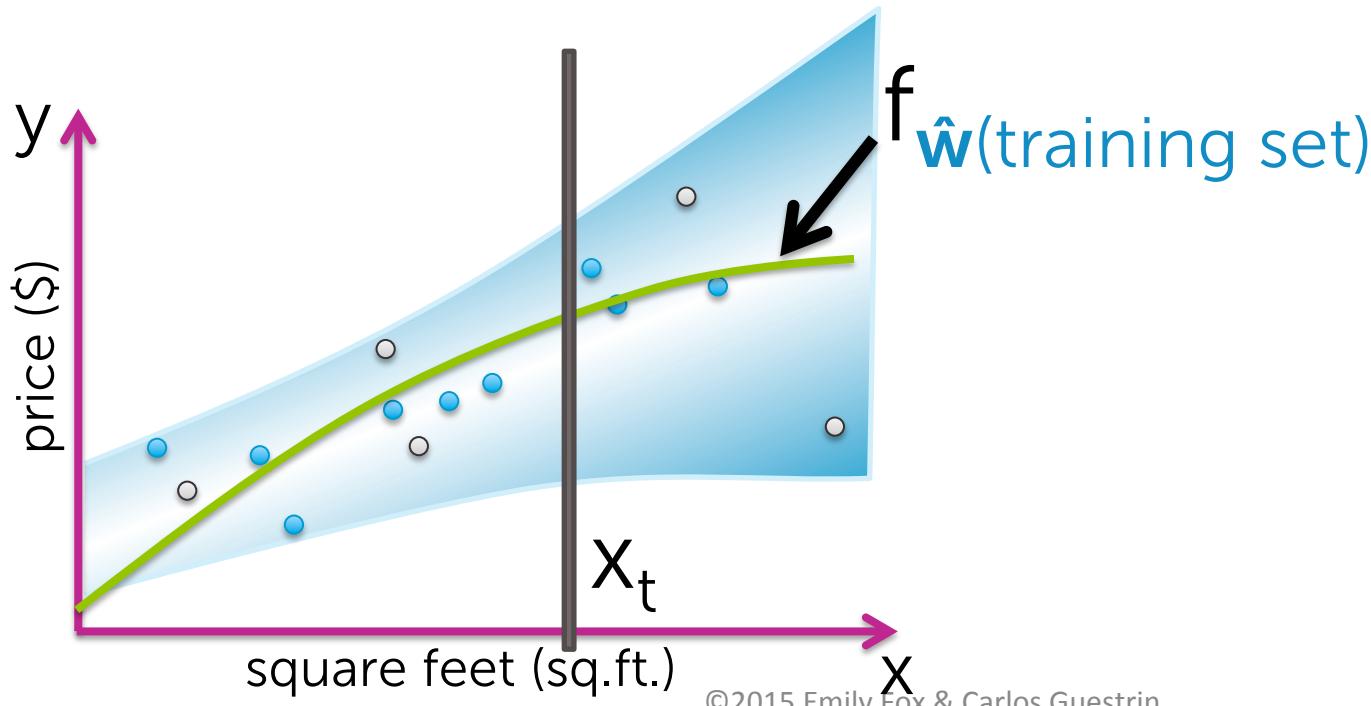
↑  
parameters fit  
on a specific  
training set



# Prediction error at target input

Start by considering:

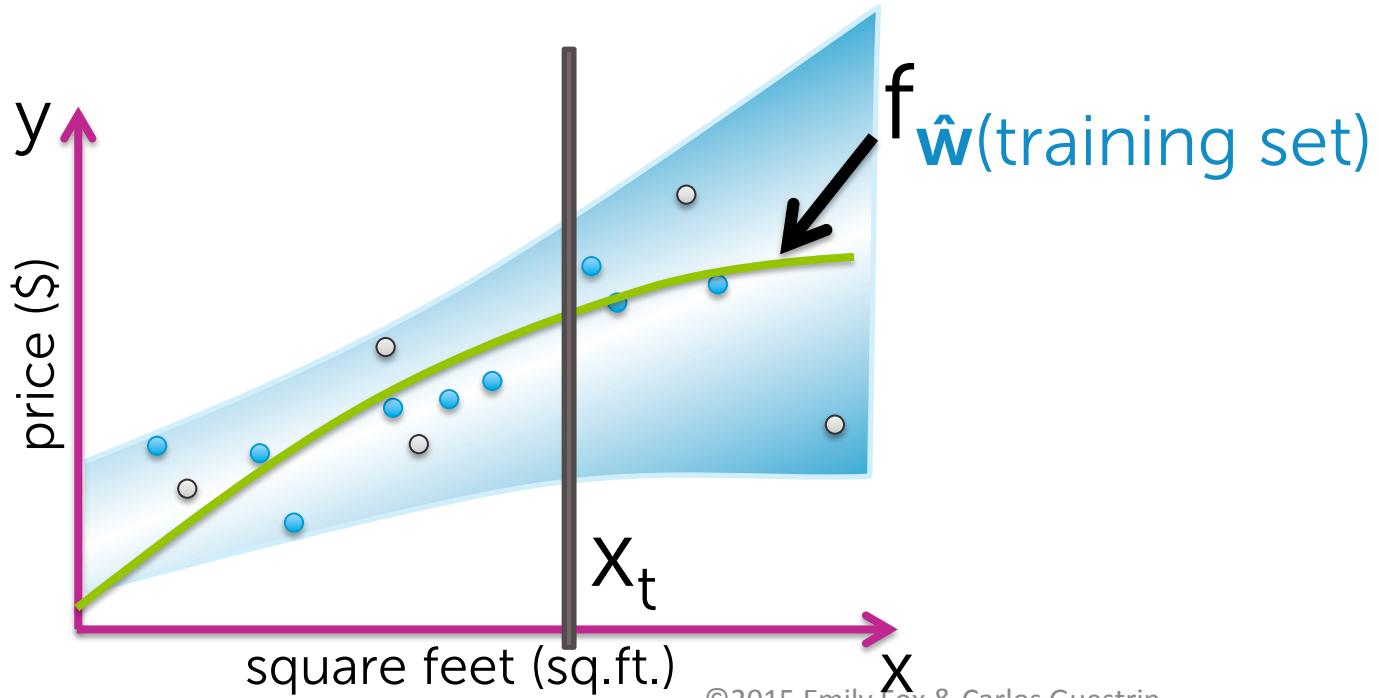
1. Loss at target  $\mathbf{x}_t$  (e.g. 2640 sq.ft.)
2. Squared error loss  $L(y, f_{\hat{\mathbf{w}}}(\mathbf{x})) = (y - f_{\hat{\mathbf{w}}}(\mathbf{x}))^2$



# Sum of 3 sources of error

Average prediction error at  $\mathbf{x}_t$

$$= \sigma^2 + [\text{bias}(f_{\hat{\mathbf{w}}}(\mathbf{x}_t))]^2 + \text{var}(f_{\hat{\mathbf{w}}}(\mathbf{x}_t))$$

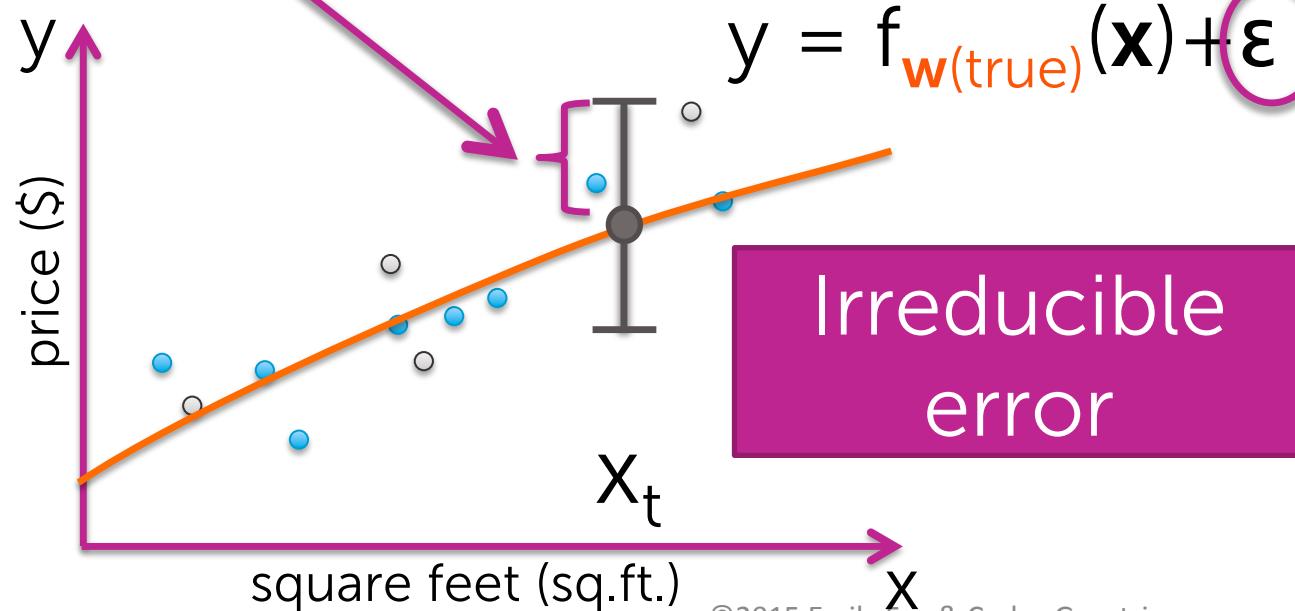


# Error variance of the model

Average prediction error at  $\mathbf{x}_t$

$$= \sigma^2 + [\text{bias}(f_{\hat{\mathbf{w}}}(\mathbf{x}_t))]^2 + \text{var}(f_{\hat{\mathbf{w}}}(\mathbf{x}_t))$$

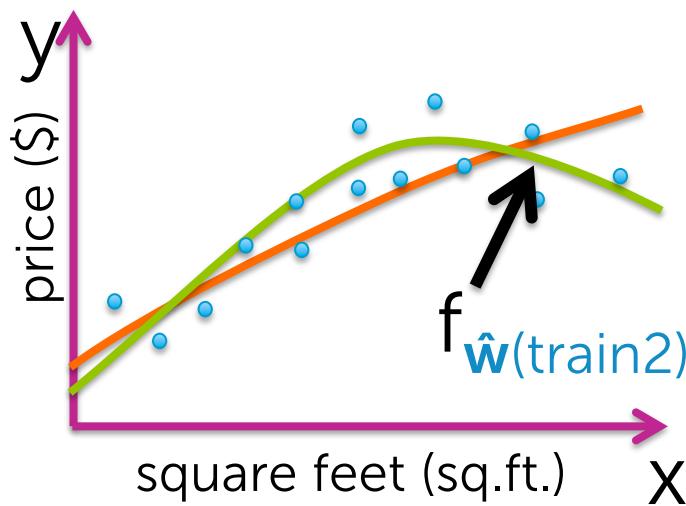
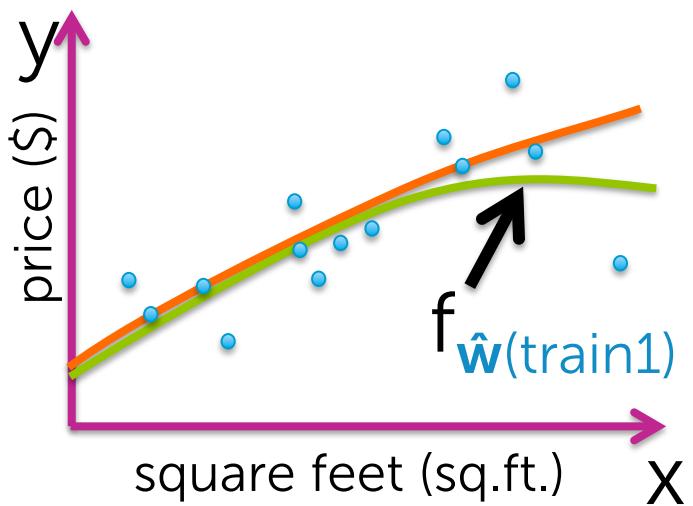
$\sigma^2 =$   
"variance" variance of noise epsilon



# Bias of function estimator

Average prediction error at  $\mathbf{x}_t$

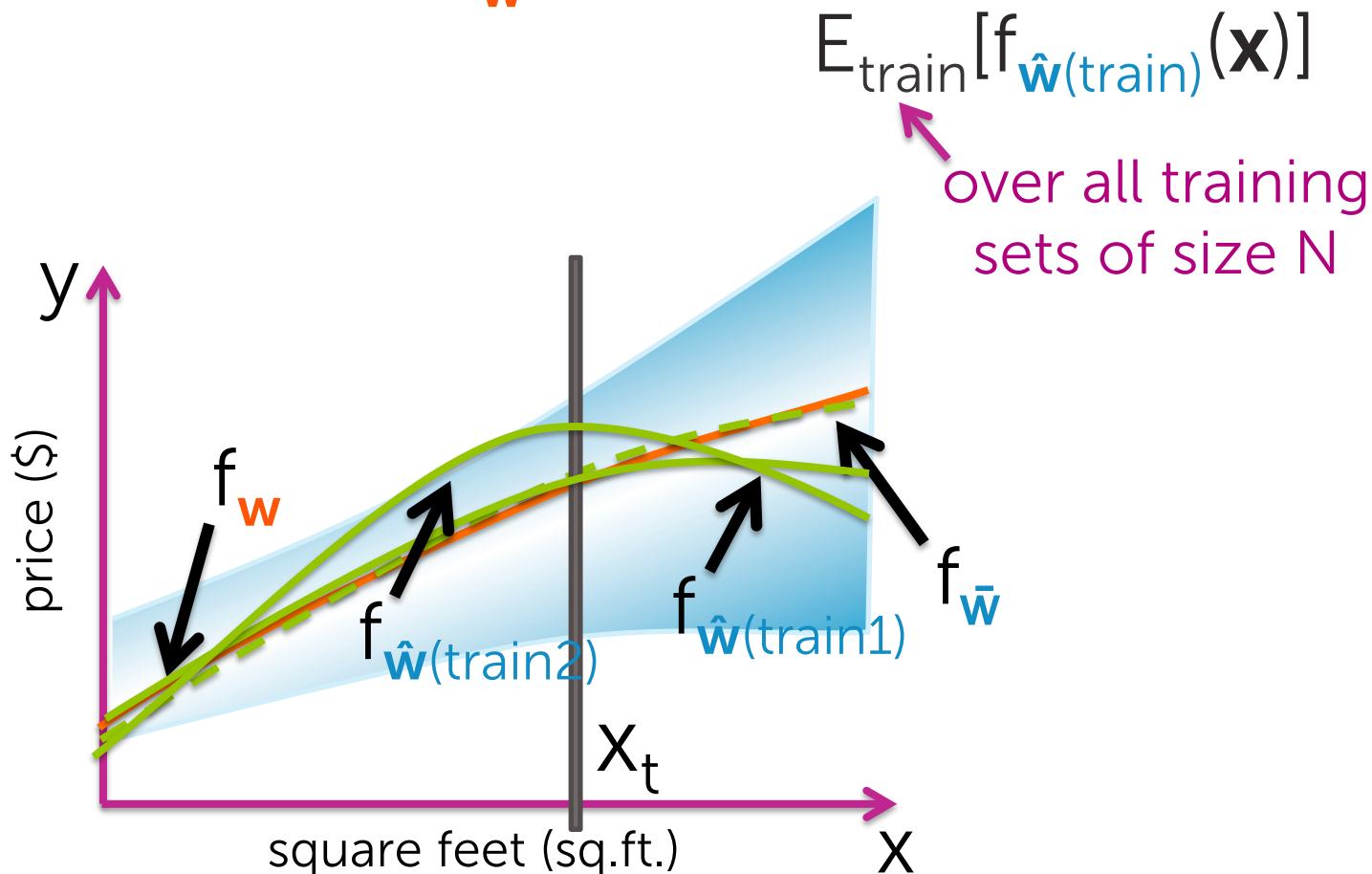
$$= \sigma^2 + [\text{bias}(f_{\hat{\mathbf{w}}}(\mathbf{x}_t))]^2 + \text{var}(f_{\hat{\mathbf{w}}}(\mathbf{x}_t))$$



# Bias of function estimator

Average estimated function =  $f_{\bar{w}}(x)$

True function =  $f_w(x)$

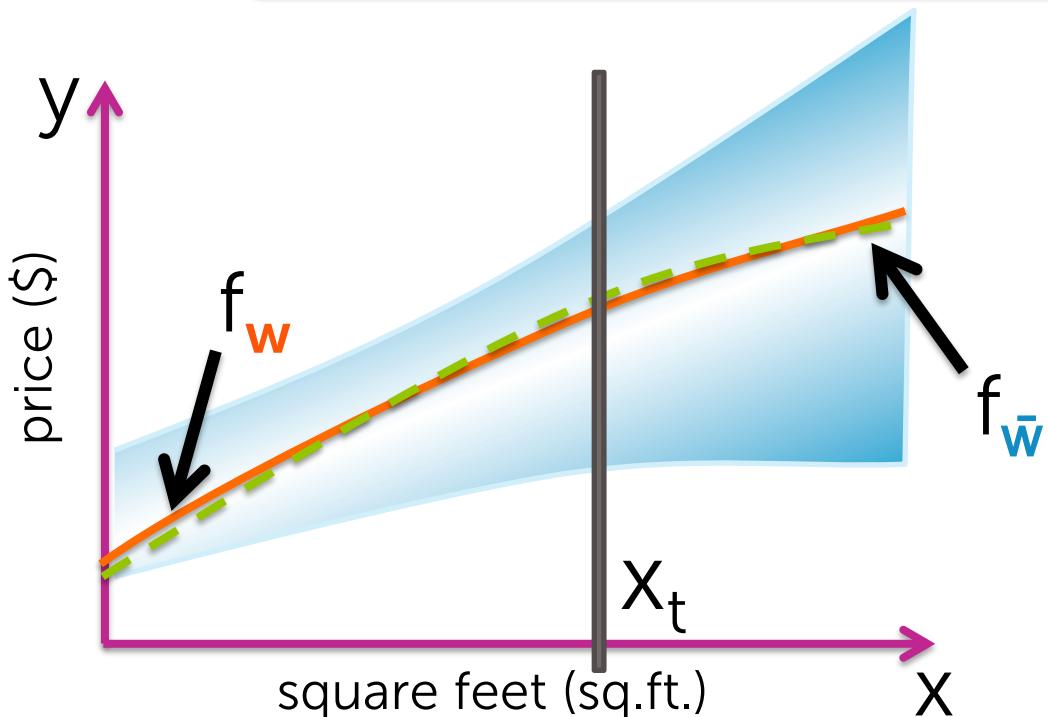


# Bias of function estimator

Average estimated function =  $f_{\bar{w}}(x)$

True function =  $f_w(x)$

$$\text{bias}(f_{\hat{w}}(x_t)) = f_w(x_t) - f_{\bar{w}}(x_t)$$



# Bias of function estimator

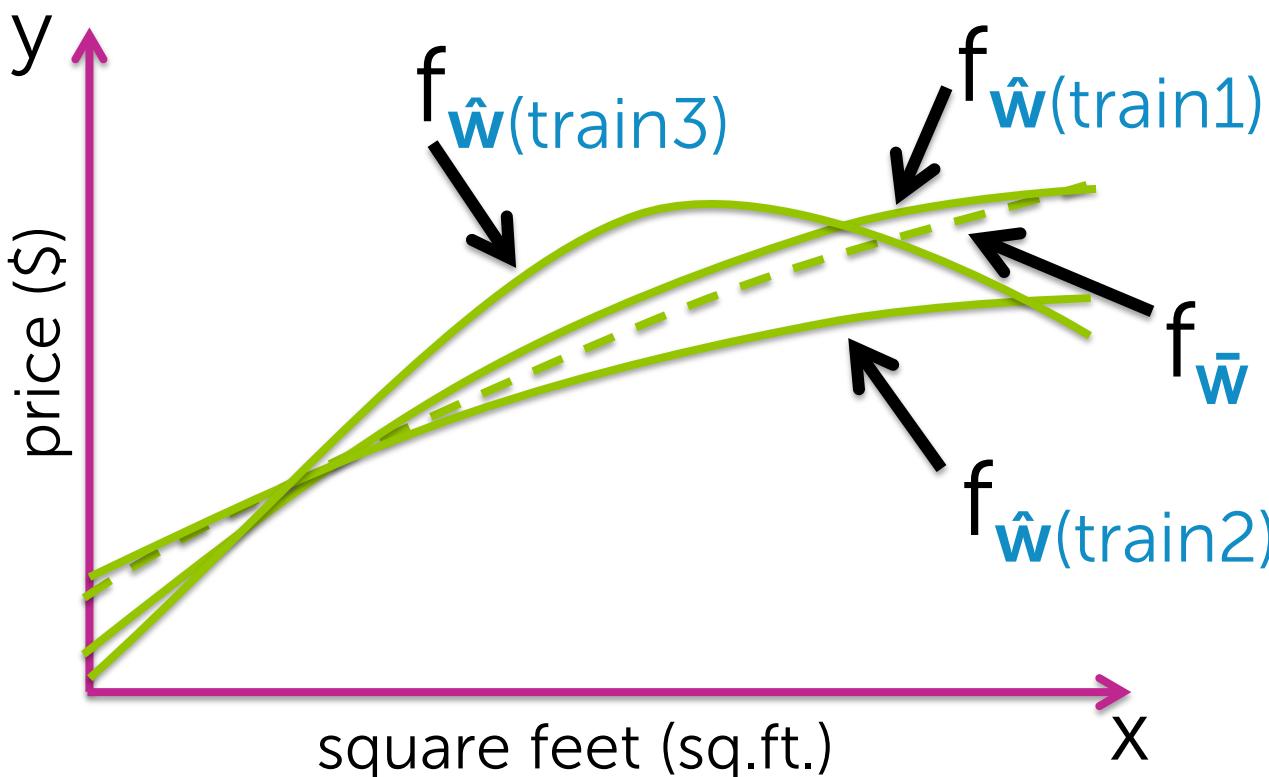
Average prediction error at  $\mathbf{x}_t$

$$= \sigma^2 + [\text{bias}(f_{\hat{\mathbf{w}}}(\mathbf{x}_t))]^2 + \text{var}(f_{\hat{\mathbf{w}}}(\mathbf{x}_t))$$

# Variance of function estimator

Average prediction error at  $\mathbf{x}_t$

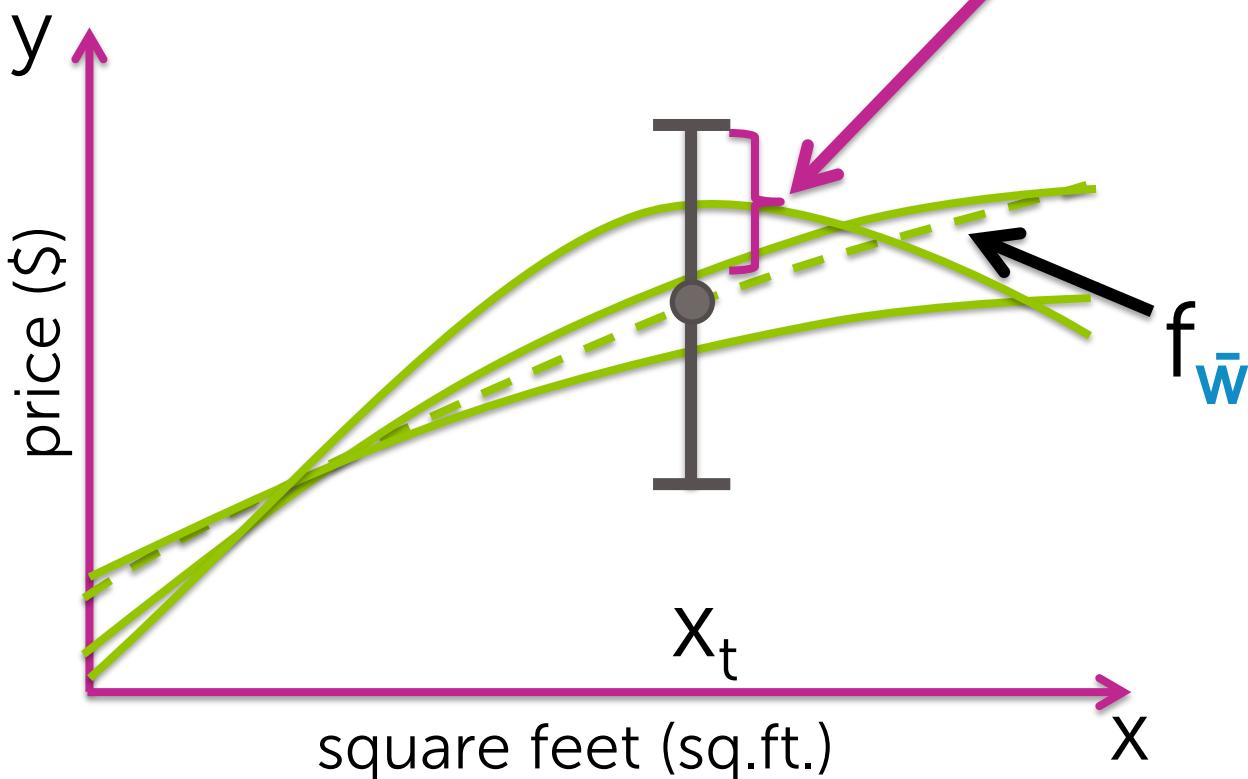
$$= \sigma^2 + [\text{bias}(f_{\hat{\mathbf{w}}}(\mathbf{x}_t))]^2 + \text{var}(f_{\hat{\mathbf{w}}}(\mathbf{x}_t))$$



# Variance of function estimator

Average prediction error at  $\mathbf{x}_t$

$$= \sigma^2 + [\text{bias}(f_{\hat{\mathbf{w}}}(\mathbf{x}_t))]^2 + \text{var}(f_{\hat{\mathbf{w}}}(\mathbf{x}_t))$$

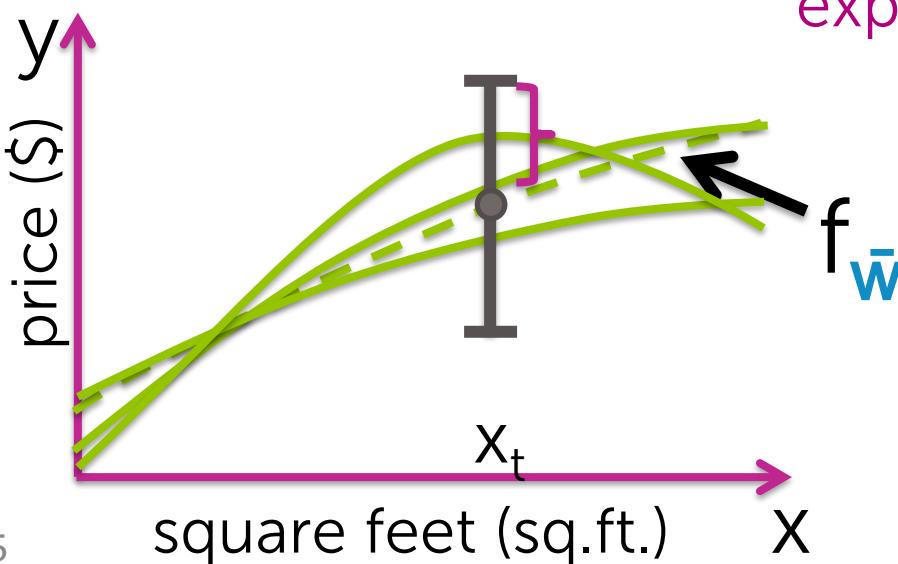


# Variance of function estimator

$$\text{var}(f_{\hat{\mathbf{w}}}(\mathbf{x}_t)) = E_{\text{train}}[(f_{\hat{\mathbf{w}}(\text{train})}(\mathbf{x}_t) - f_{\bar{\mathbf{w}}}(\mathbf{x}_t))^2]$$

over all training sets of size N

deviation of specific fit from expected fit at  $\mathbf{x}_t$



# Why 3 sources of error? A formal derivation

**OPTIONAL**

# Deriving expected prediction error

Expected prediction error

$$= E_{\text{train}} [\text{generalization error of } \hat{\mathbf{w}}(\text{train})]$$

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

1. Look at specific  $\mathbf{x}_t$
2. Consider  $L(y, f_{\hat{\mathbf{w}}}(\mathbf{x})) = (y - f_{\hat{\mathbf{w}}}(\mathbf{x}))^2$

Expected prediction error at  $\mathbf{x}_t$

$$= E_{\text{train}, y_t} [(y_t - f_{\hat{\mathbf{w}}(\text{train})}(\mathbf{x}_t))^2]$$

# Deriving expected prediction error

Expected prediction error at  $\mathbf{x}_t$

$$= E_{\text{train}, y_t} [(y_t - f_{\hat{\mathbf{w}}(\text{train})}(\mathbf{x}_t))^2]$$

$$= E_{\text{train}, y_t} [((y_t - f_{\mathbf{w}(\text{true})}(\mathbf{x}_t)) + (f_{\mathbf{w}(\text{true})}(\mathbf{x}_t) - f_{\hat{\mathbf{w}}(\text{train})}(\mathbf{x}_t)))^2]$$

$$\begin{aligned} &= \cancel{E_{\text{train}, y} [(y - f)^2]}_{\text{by definition } \sigma^2} + 2 E_{\text{train}, y} [(y - f)(f - \hat{f})] + E_{\text{train}, y} [(f - \hat{f})^2] \\ &\quad \underbrace{E[\epsilon] E[f - \hat{f}]}_0 \\ &\quad \triangleq \text{MSE}(\hat{f}) \text{ mean square error} \end{aligned}$$

$$= \sigma^2 + \underline{\text{MSE}(\hat{f})}$$

Aside:

$$\begin{aligned} E[a+b] &\stackrel{\text{ind. r.v.'s}}{=} E[a]E[b] \\ E[(a+b)^2] &= E[a^2 + 2ab + b^2] \\ &= E[a^2] + 2E[ab] + E[b^2] \end{aligned}$$

Shorthand:

$$\begin{aligned} y_t &\rightarrow y \\ f_{\mathbf{w}(\text{true})} &\rightarrow f \\ f_{\hat{\mathbf{w}}(\text{train})} &\rightarrow \hat{f} \end{aligned}$$

# Equating MSE with bias and variance

$$\begin{aligned} \text{MSE}[f_{\hat{\mathbf{w}}(\text{train})}(\mathbf{x}_t)] &= E_{\text{train}}[(f_{\mathbf{w}(\text{true})}(\mathbf{x}_t) - f_{\hat{\mathbf{w}}(\text{train})}(\mathbf{x}_t))^2] \\ &= E_{\text{train}}[((f_{\mathbf{w}(\text{true})}(\mathbf{x}_t) - \bar{f}_{\hat{\mathbf{w}}}(\mathbf{x}_t)) + (\bar{f}_{\hat{\mathbf{w}}}(\mathbf{x}_t) - f_{\hat{\mathbf{w}}(\text{train})}(\mathbf{x}_t)))^2] \\ &= E_{\text{train}}[(f - \bar{f})^2] + 2E_{\text{train}}[(f - \bar{f})(\bar{f} - \hat{f})] + E_{\text{train}}[(\bar{f} - \hat{f})^2] \\ &\quad \underbrace{(f - \bar{f})^2}_{\text{by definition} = \text{bias}^2(\hat{f})} \quad \underbrace{2(f - \bar{f})E_{\text{train}}[\bar{f} - \hat{f}]}_{\substack{\text{not a fun of} \\ \text{fun of training data}}} \quad \underbrace{E[(\bar{f} - \hat{f})^2]}_{\substack{\uparrow \\ \text{random function} \\ \text{at } \mathbf{x}_t \\ = \text{random variable}}} = \text{var}(\hat{f}) \\ &= \text{bias}^2(\hat{f}) + \text{Var}(\hat{f}) \end{aligned}$$

shorthand  
new notation  
on this slide

E<sub>train</sub>[ $\hat{f}$ ] =  $\bar{f}$

E<sub>train</sub>[ $\bar{f} - \hat{f}$ ] = 0

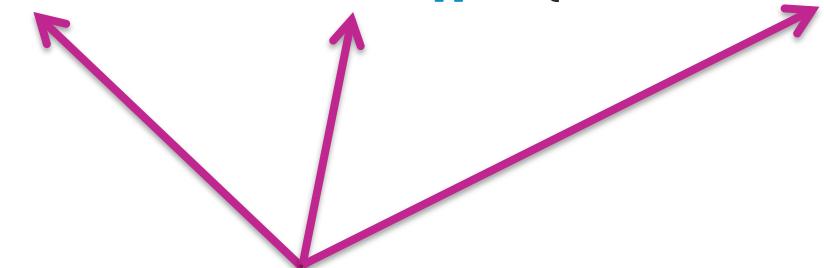
E<sub>train</sub>[ $(\bar{f} - \bar{f})^2$ ] =  $\text{var}(\bar{f})$

# Putting it all together

Expected prediction error at  $\mathbf{x}_t$

$$= \sigma^2 + \text{MSE}[f_{\hat{\mathbf{w}}}(\mathbf{x}_t)]$$

$$= \sigma^2 + [\text{bias}(f_{\hat{\mathbf{w}}}(\mathbf{x}_t))]^2 + \text{var}(f_{\hat{\mathbf{w}}}(\mathbf{x}_t))$$



# Summary of tasks

# The regression/ML workflow

## 1. Model selection

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

## 2. Model assessment

Having selected a model, assess the generalization error

# Hypothetical implementation

Training set

Test set

## 1. Model selection

For each considered model complexity  $\lambda$  :

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

## 2. Model assessment

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

# Hypothetical implementation

Training set

Test set

## 1. Model selection

For each considered model complexity  $\lambda$  :

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

Overly optimistic!

## 2. Model assessment

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

# Hypothetical implementation

Training set

Test set

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

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

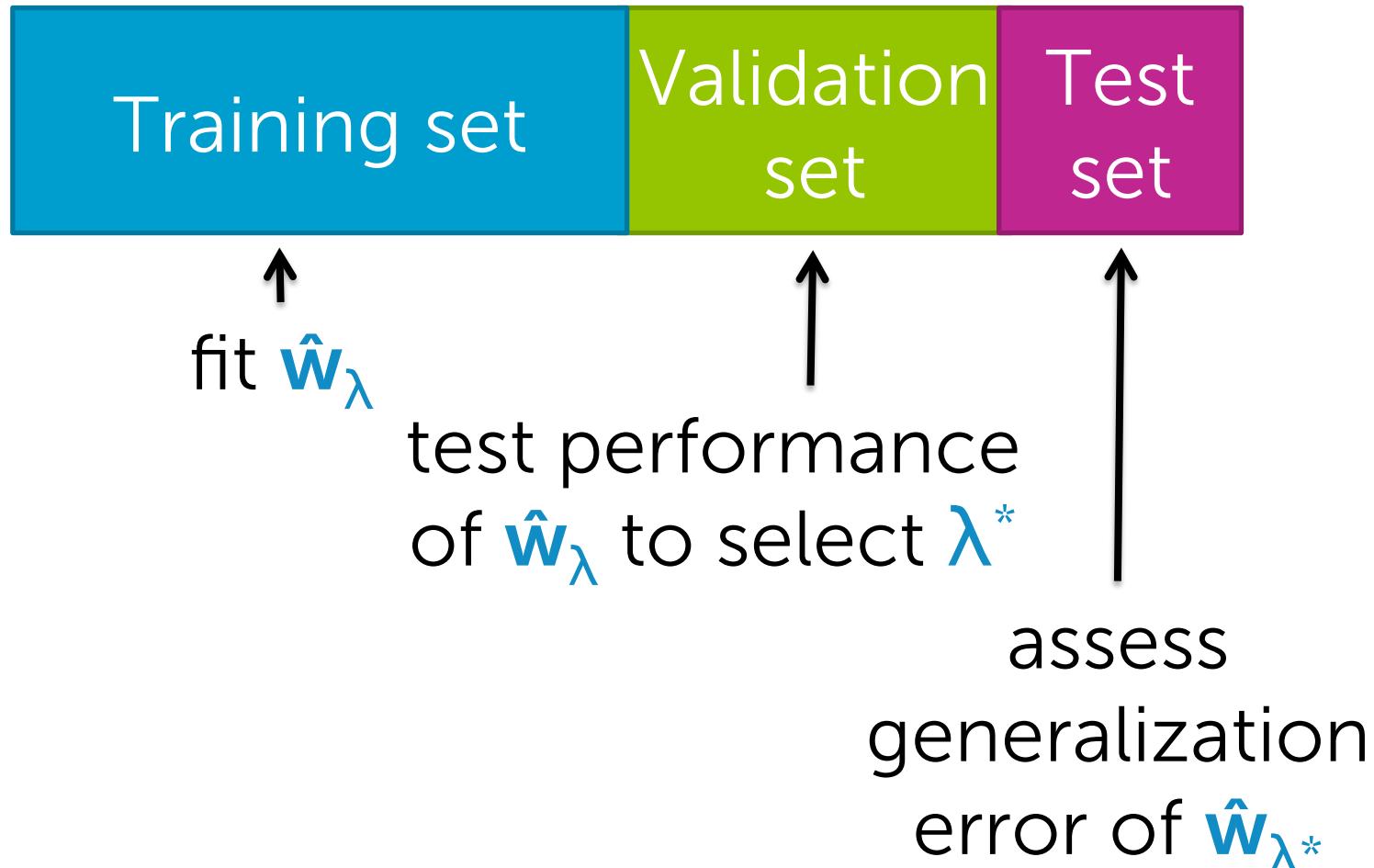
# Practical implementation



**Solution:** Create two “test” sets!

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

# Practical implementation



# Typical splits



80%

10%

10%

50%

25%

25%

# Summary of assessing performance

# What you can do now...

- 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