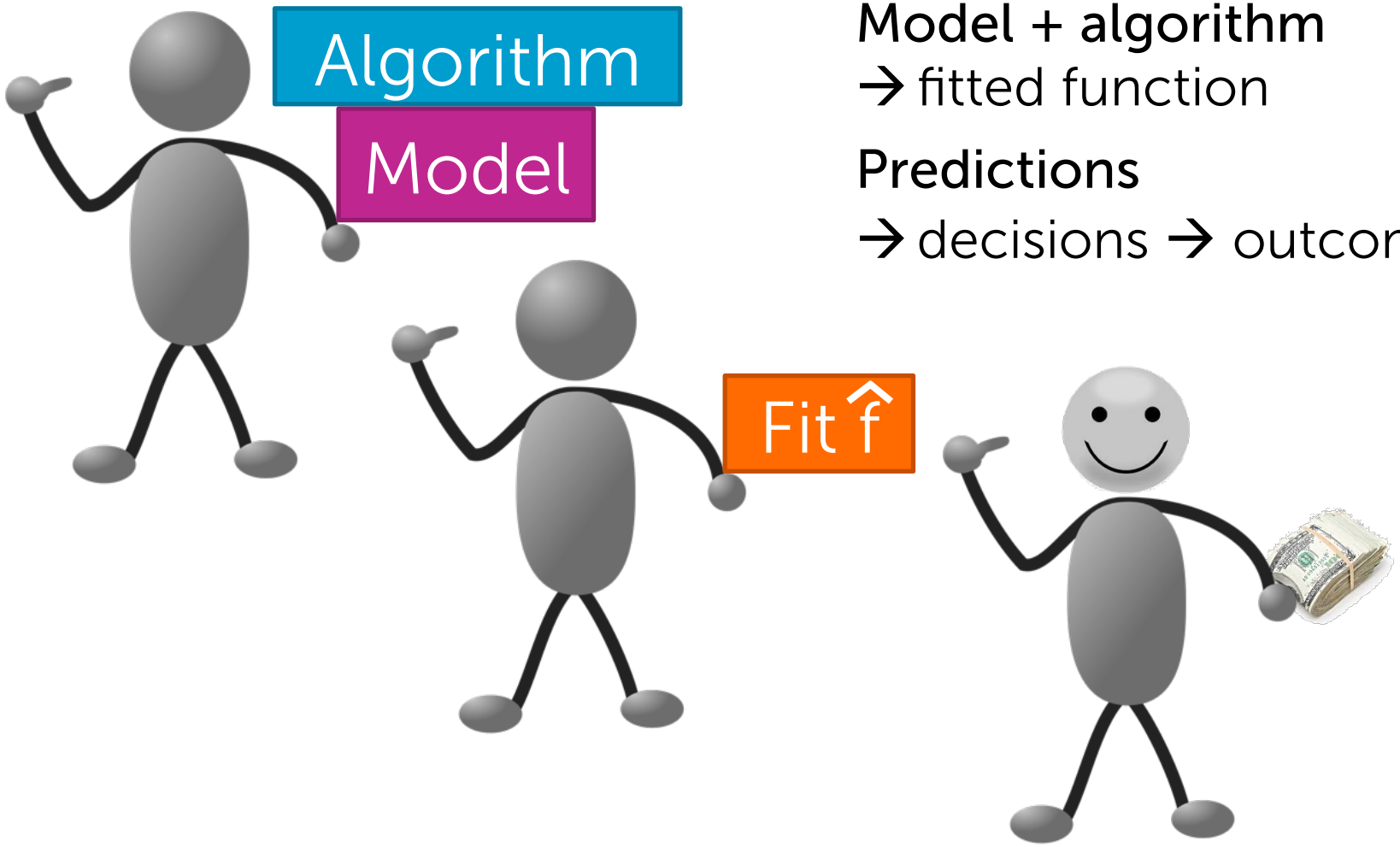
Assessing Performance

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:

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

Cost of using ŵ at x when y is true

actual value

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

Examples:

(assuming loss for underpredicting = overpredicting)

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

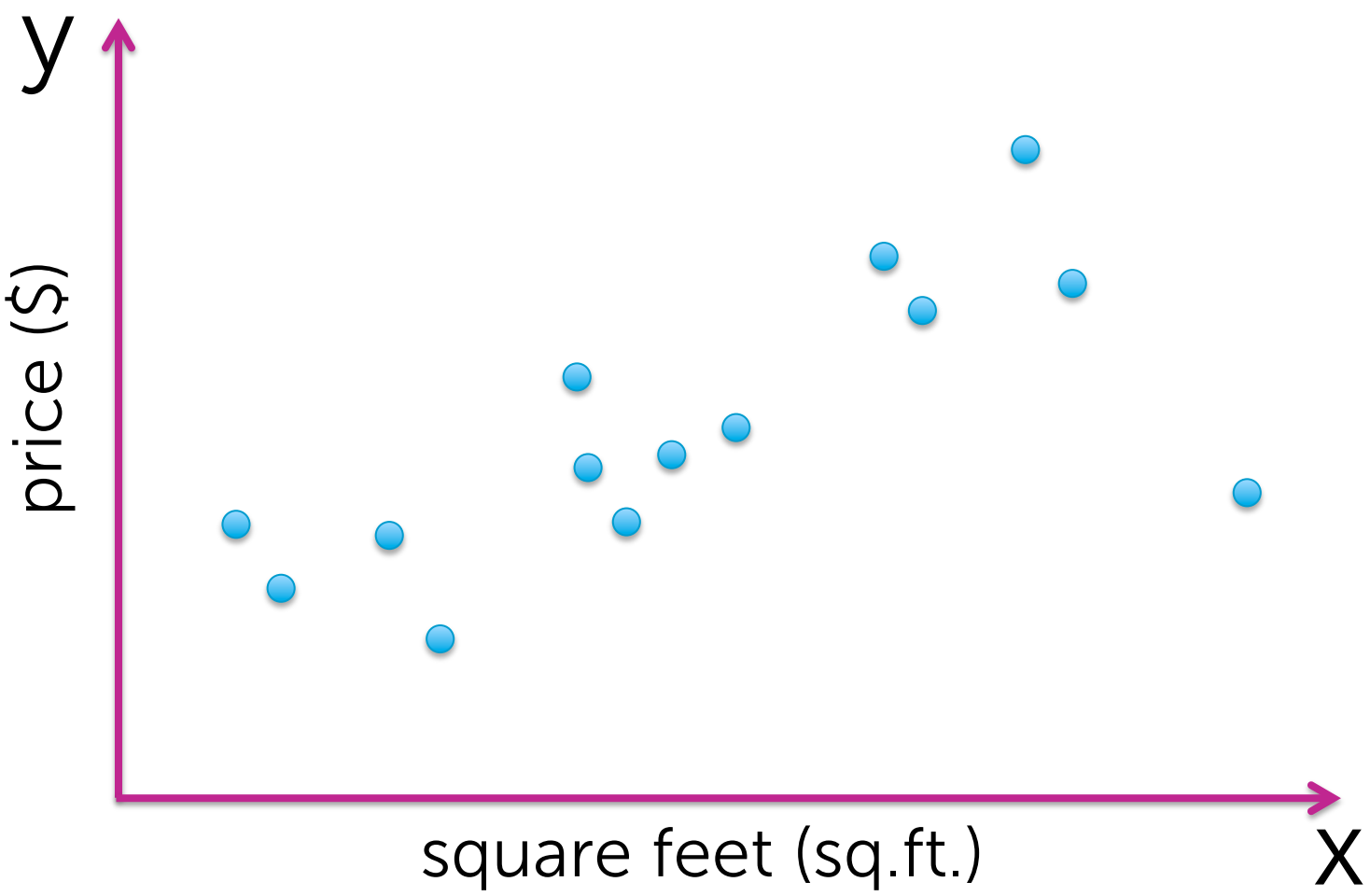
Squared error: $L(y, f_{\hat{w}}(\mathbf{x})) = (y - f_{\hat{w}}(\mathbf{x}))^2$

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

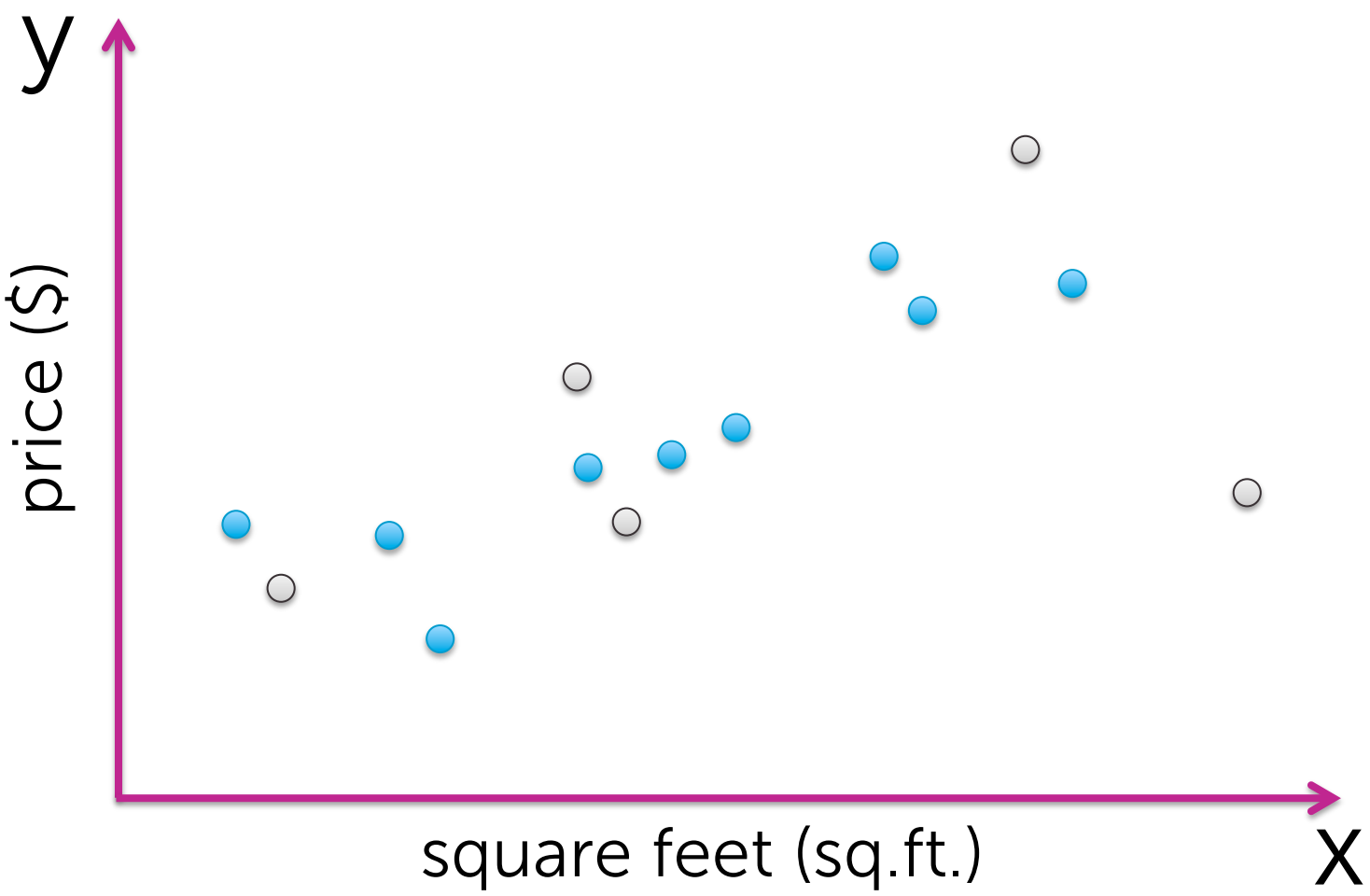
Assessing the loss

Assessing the loss
Part 1: Training error

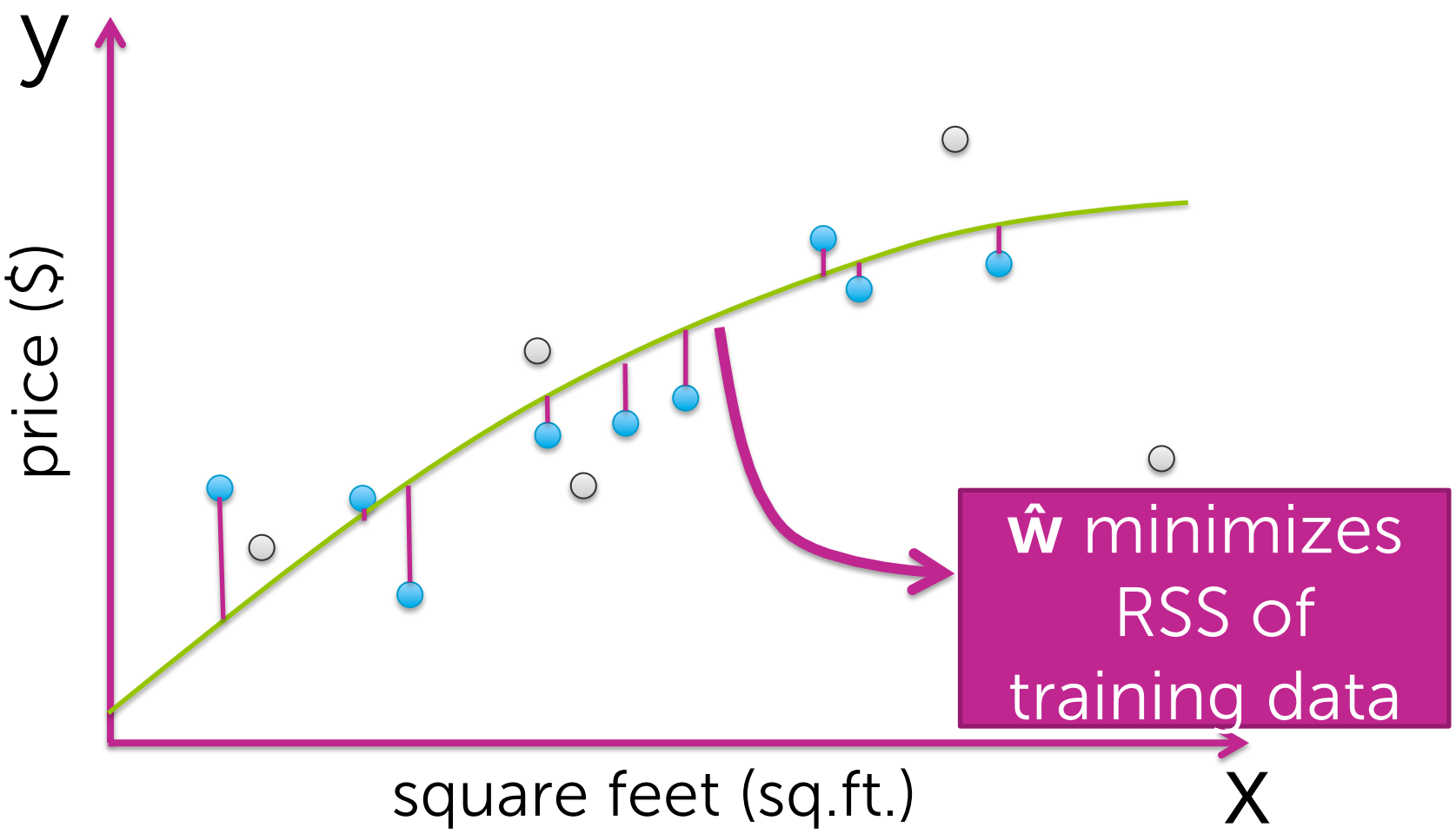
Define training data



Define training data



Example:
Fit quadratic to minimize RSS



Compute training error

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

2. Training error

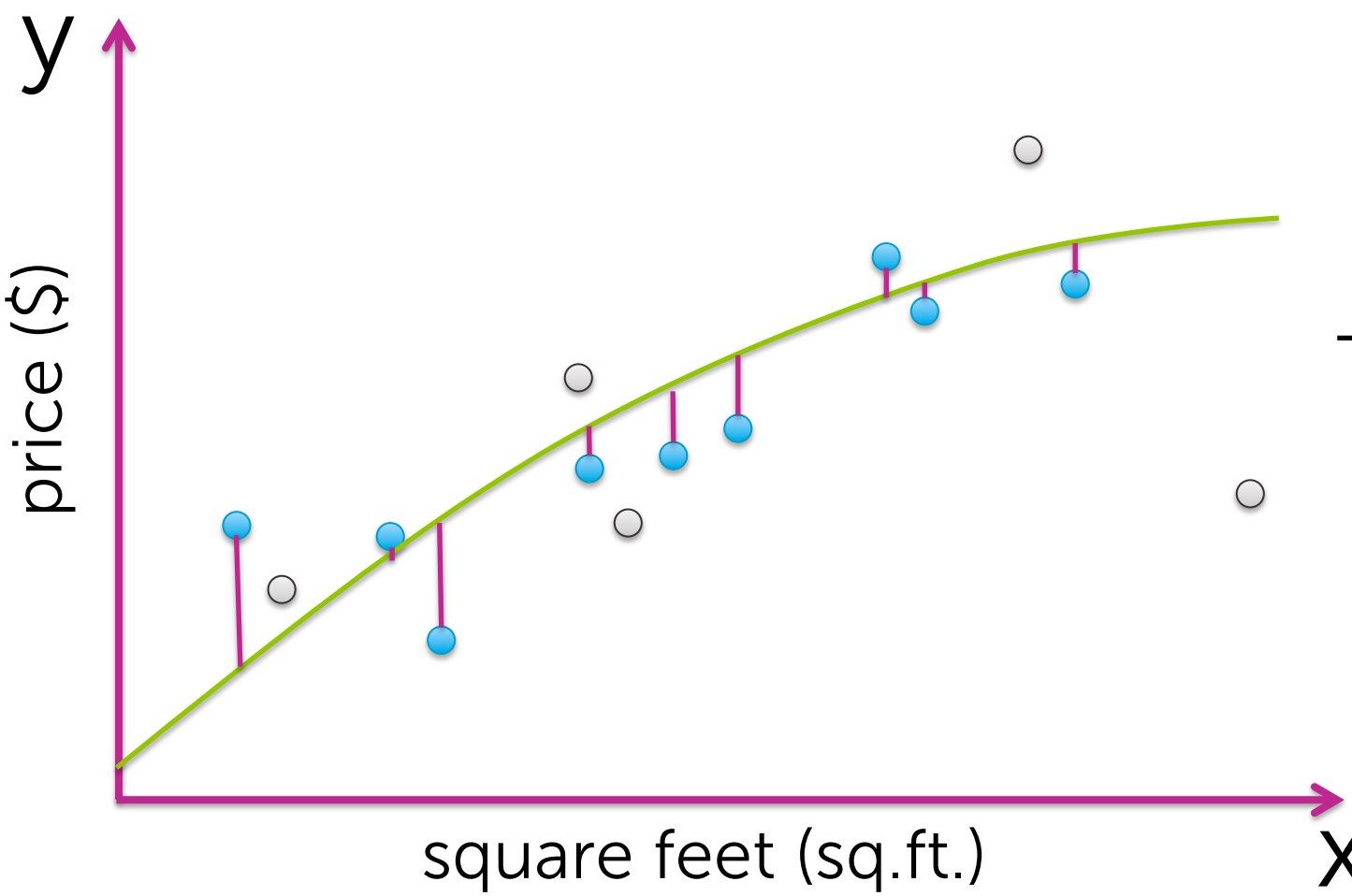
   = avg. loss on houses in training set

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

fit using training data

Example:

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



Training error ($\hat{w}$) = 1/N *

$[(\$_{train\ 1}-f_{\hat{w}}(sq.ft._{train\ 1}))^2$

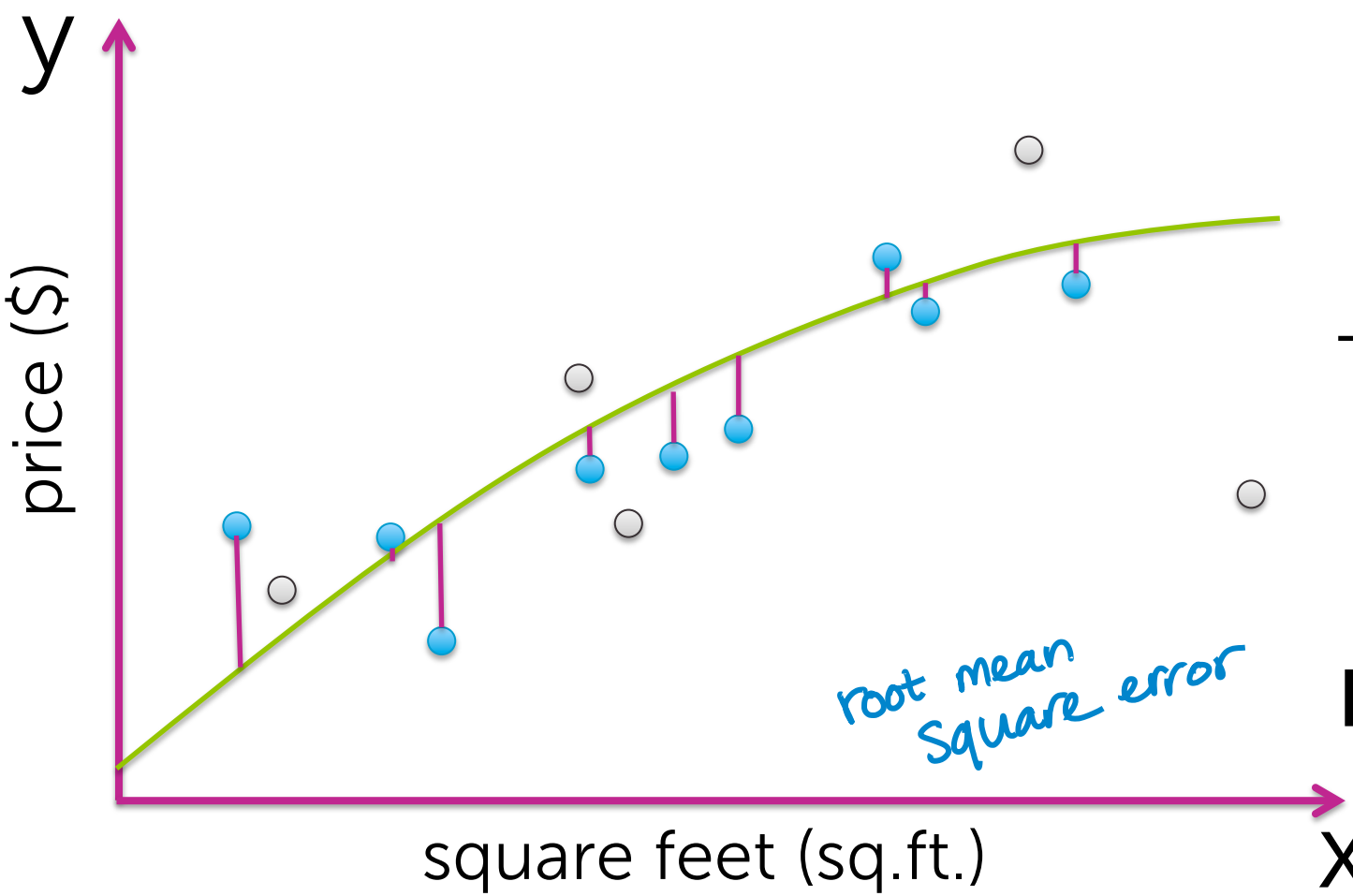
$+ (\$_{train\ 2}-f_{\hat{w}}(sq.ft._{train\ 2}))^2$

$+ (\$_{train\ 3}-f_{\hat{w}}(sq.ft._{train\ 3}))^2$

+ … include all training houses]

Example:

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



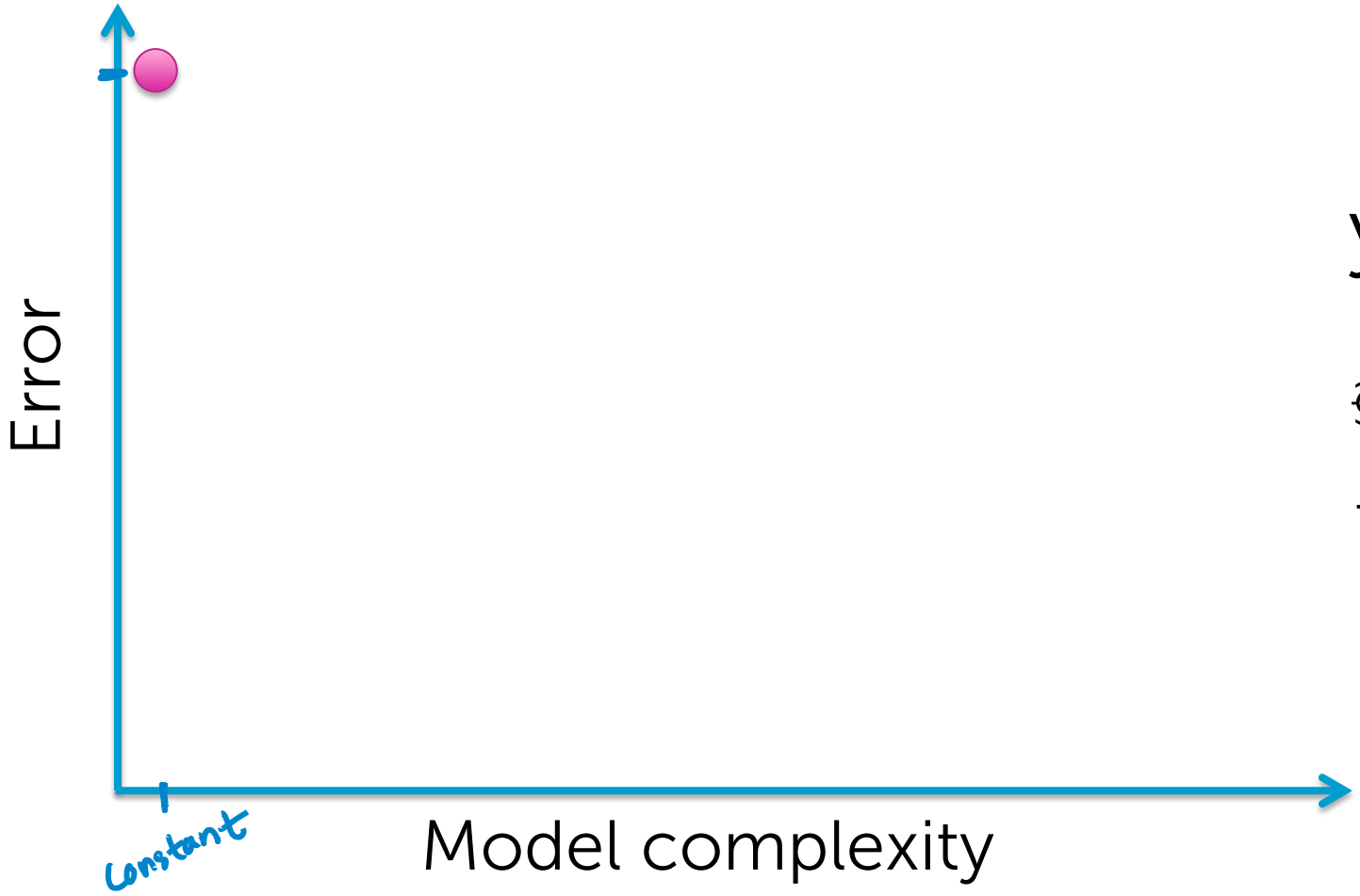
Training error ($\hat{w}$) =

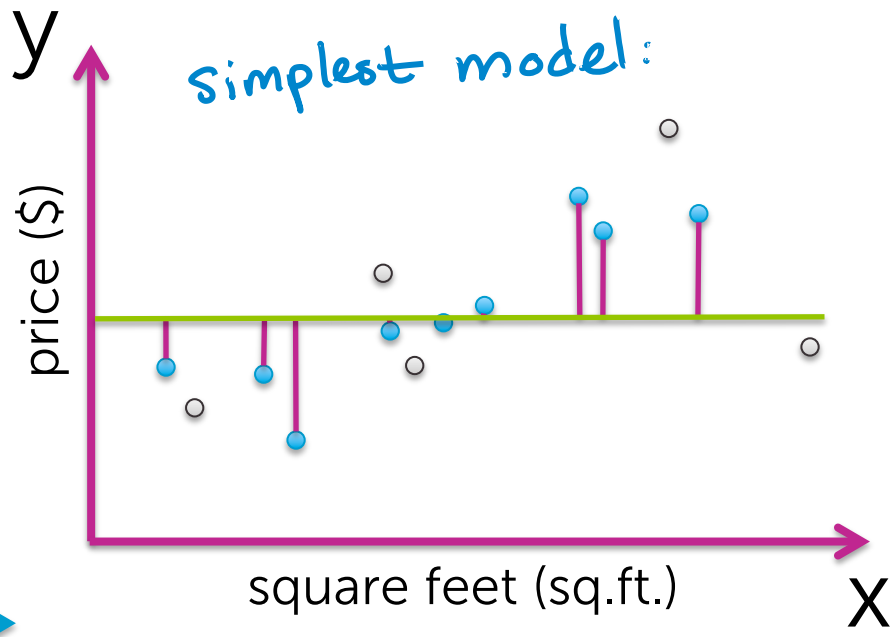
$$\frac{1}{N}\sum_{i=1}^{N}(y_i-f_{\hat{w}}(x_i))^2$$

RMSE =

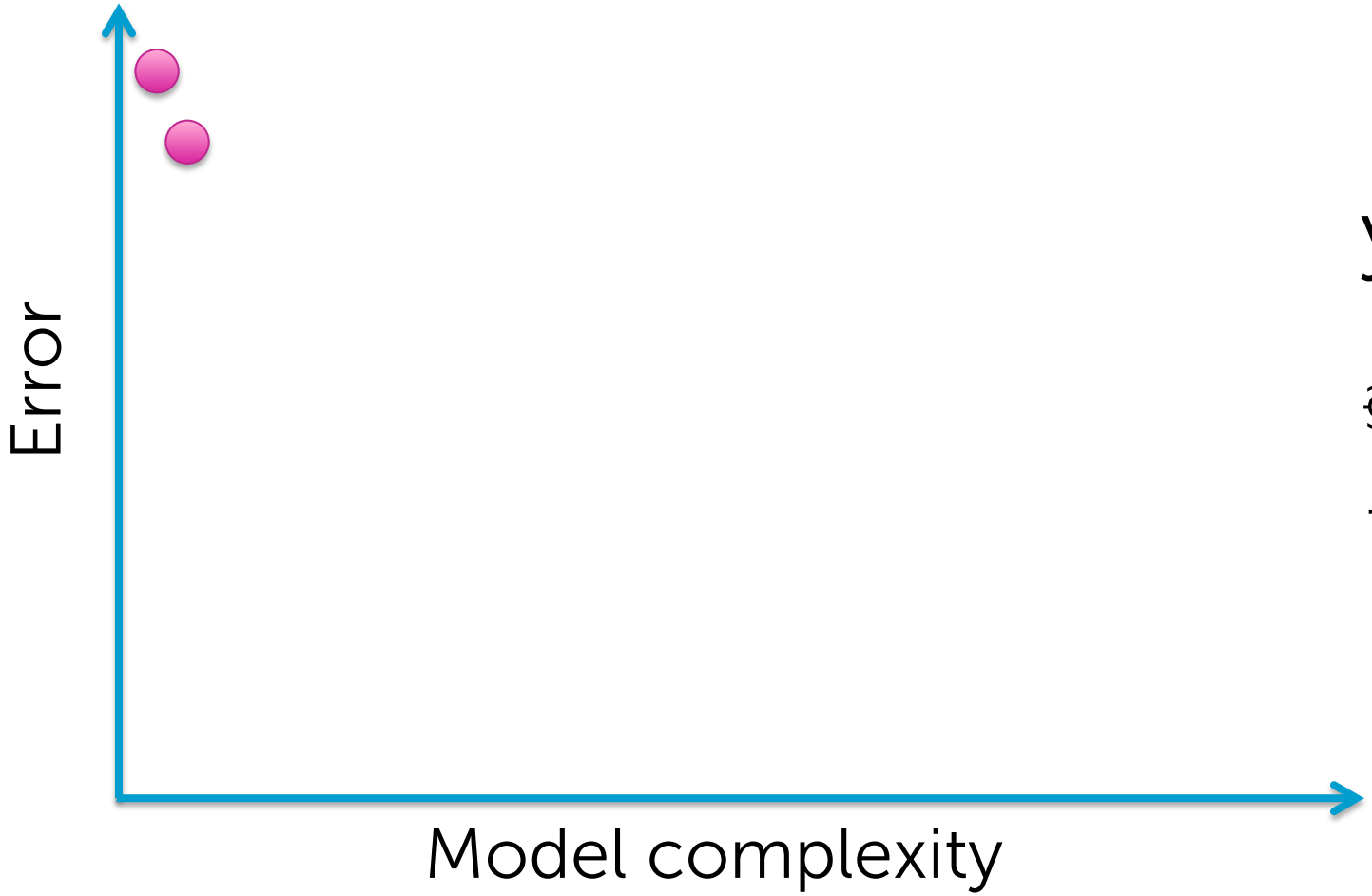
$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i-f_{\hat{w}}(x_i))^2}$$

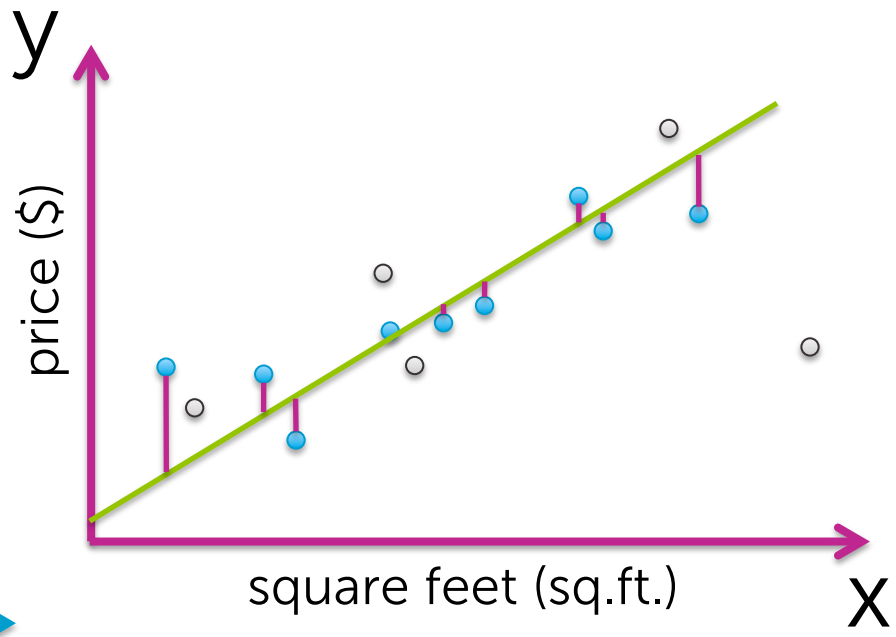
Training error vs. model complexity



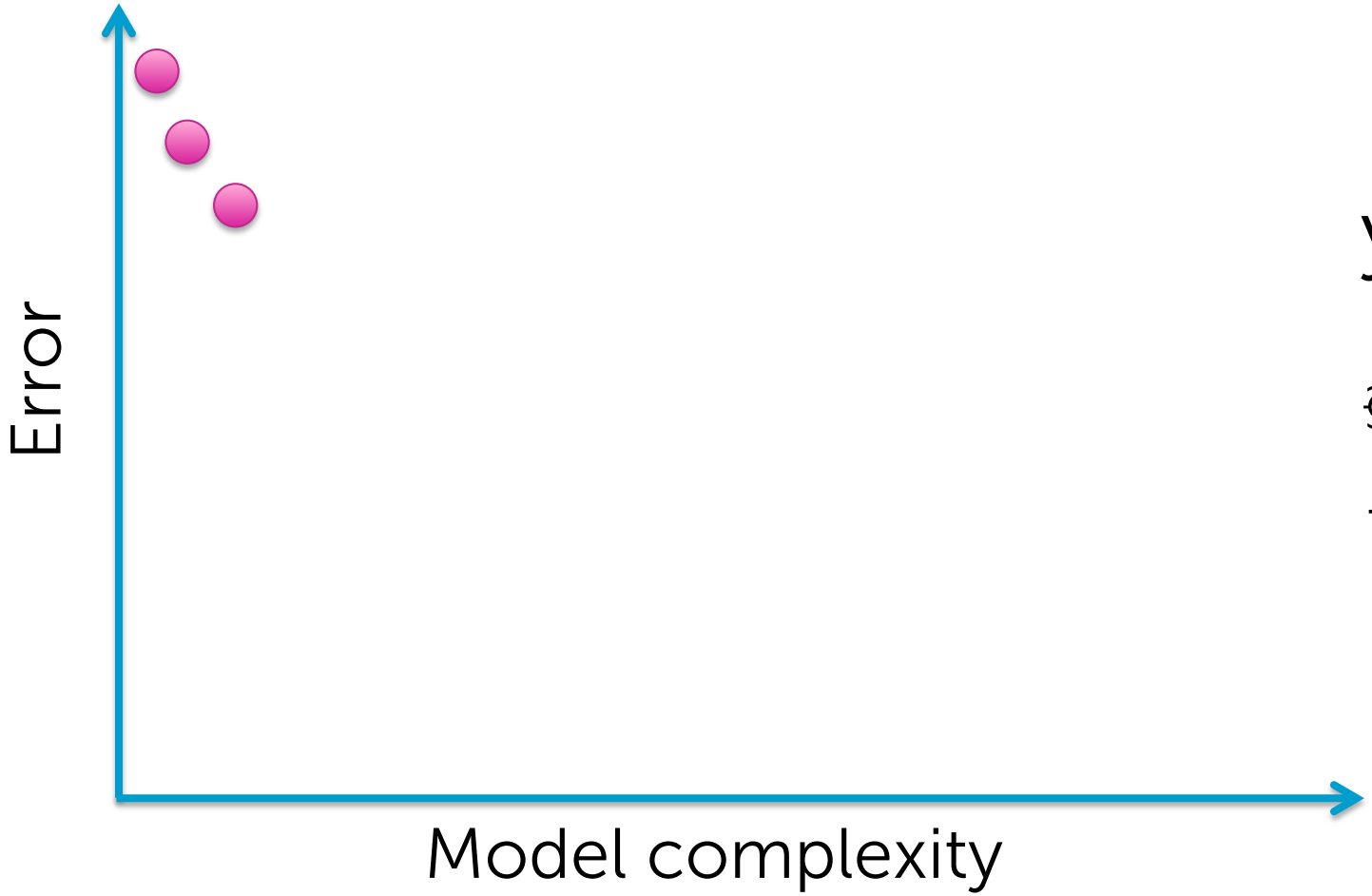


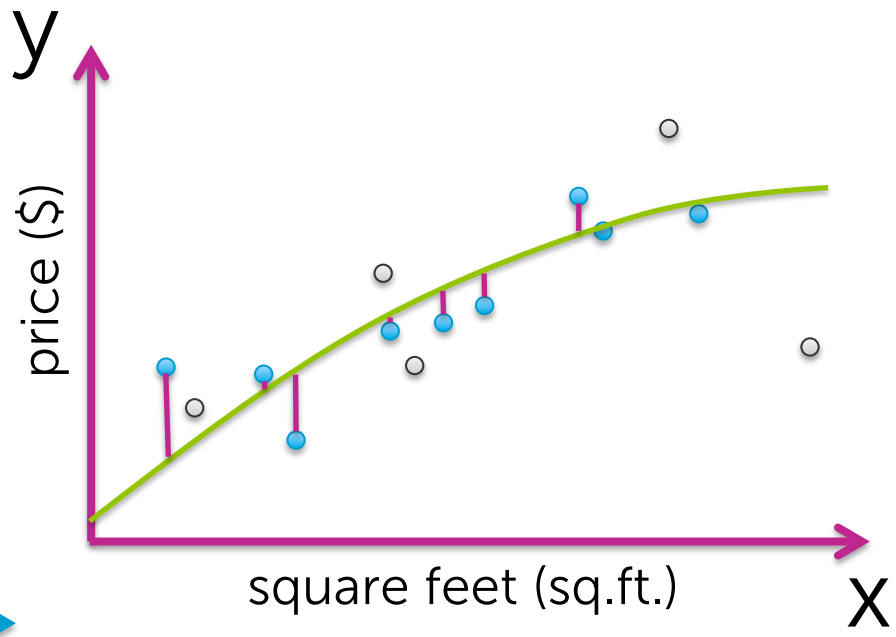
Training error vs. model complexity



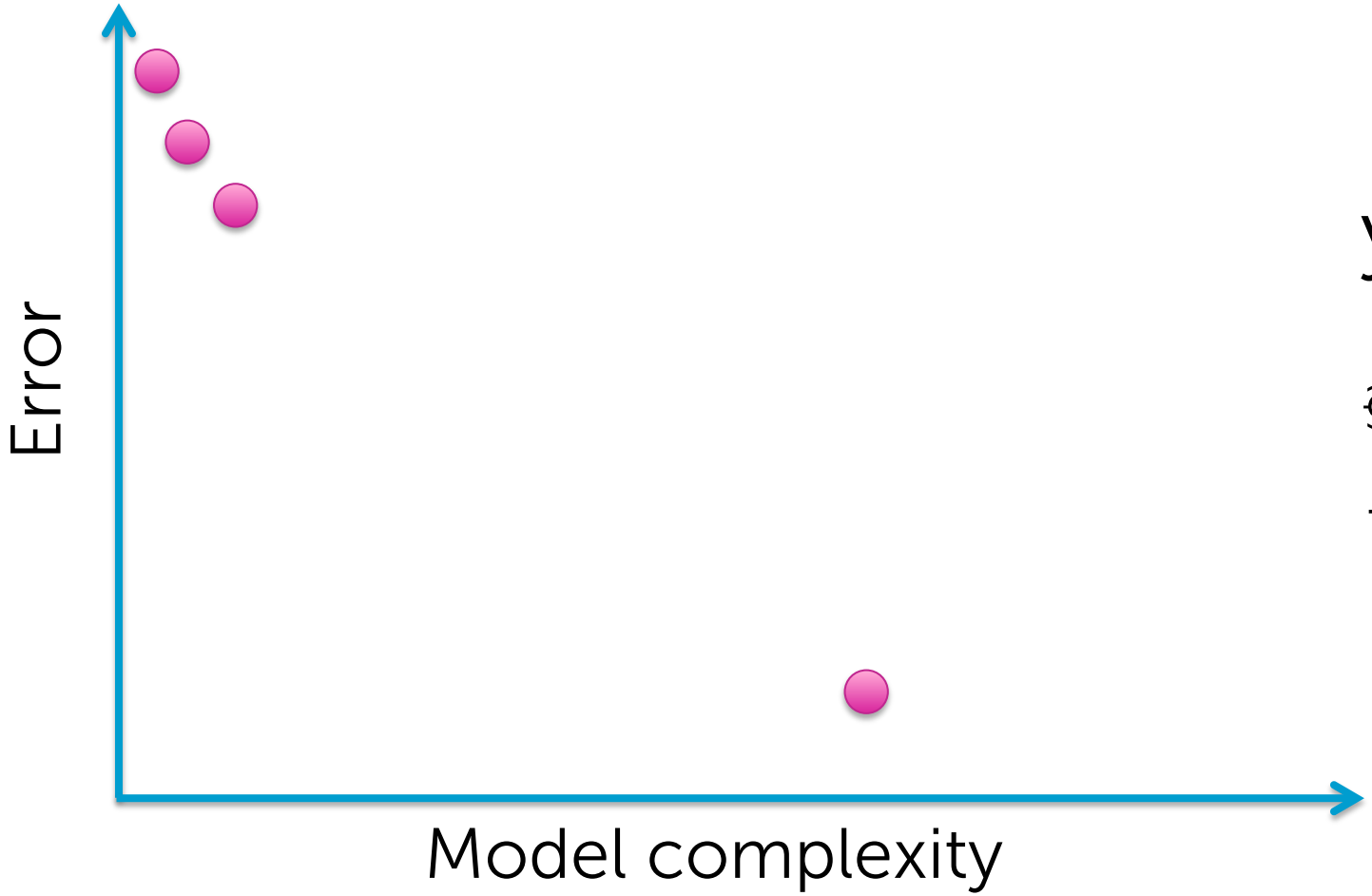


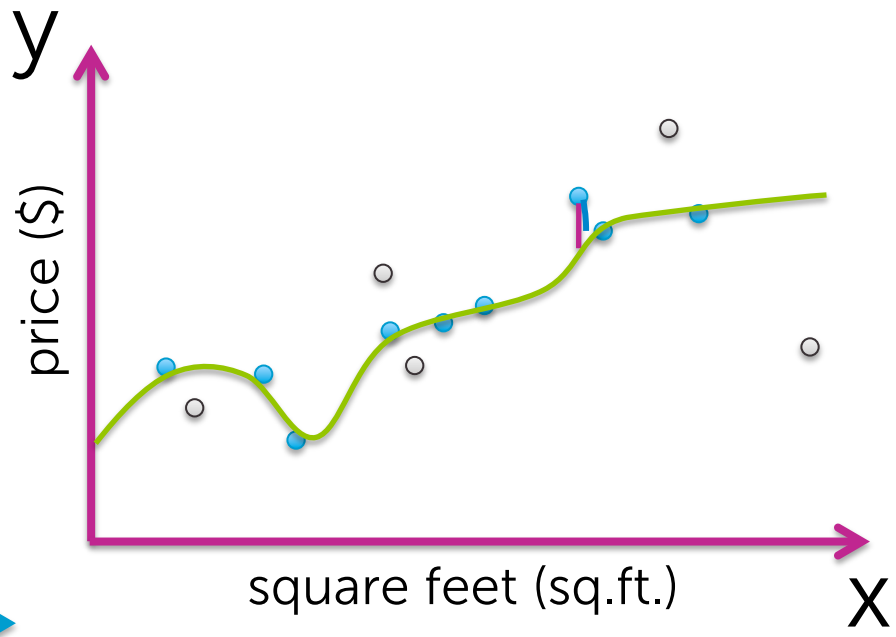
Training error vs. model complexity



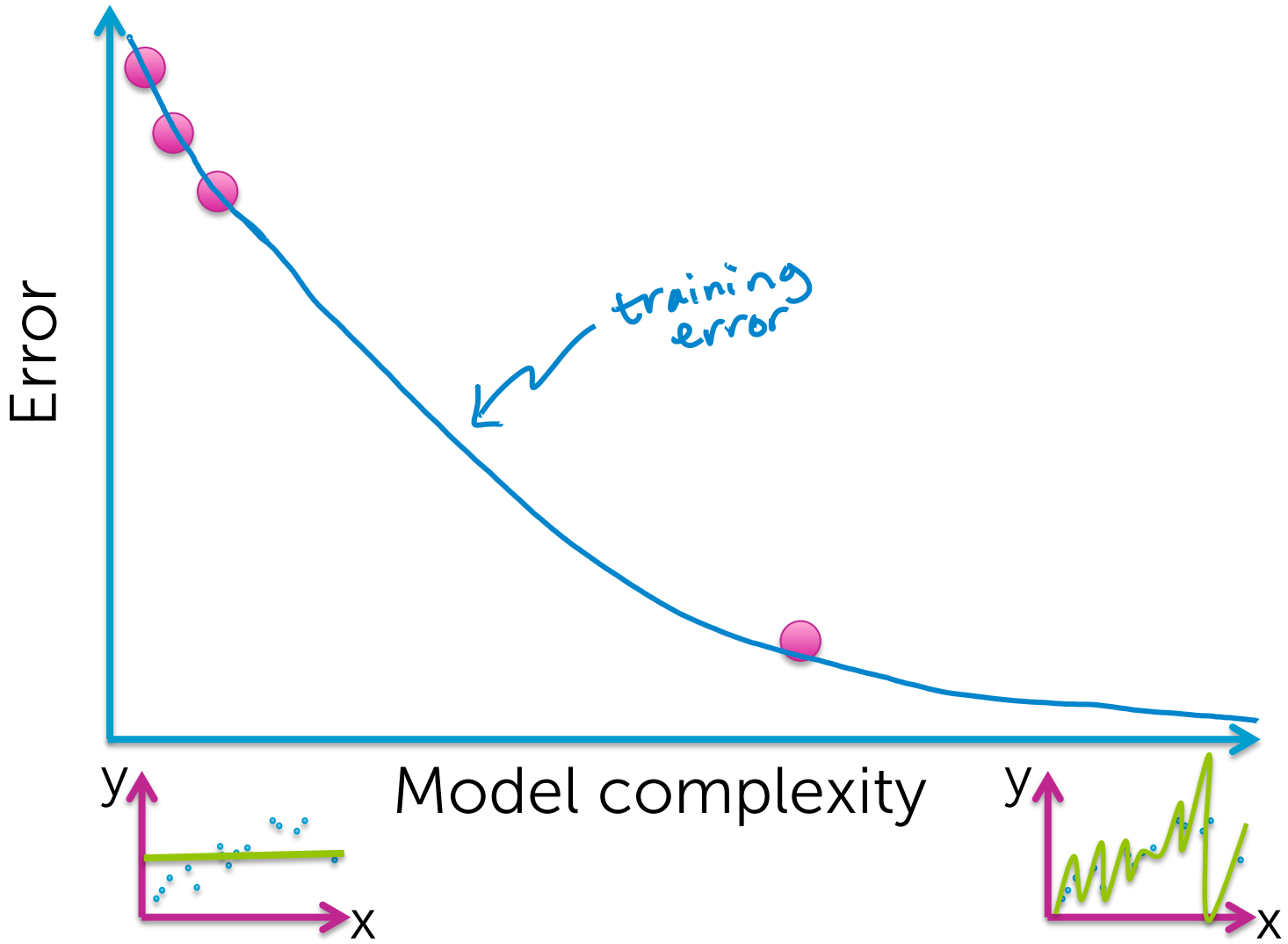


Training error vs. model complexity



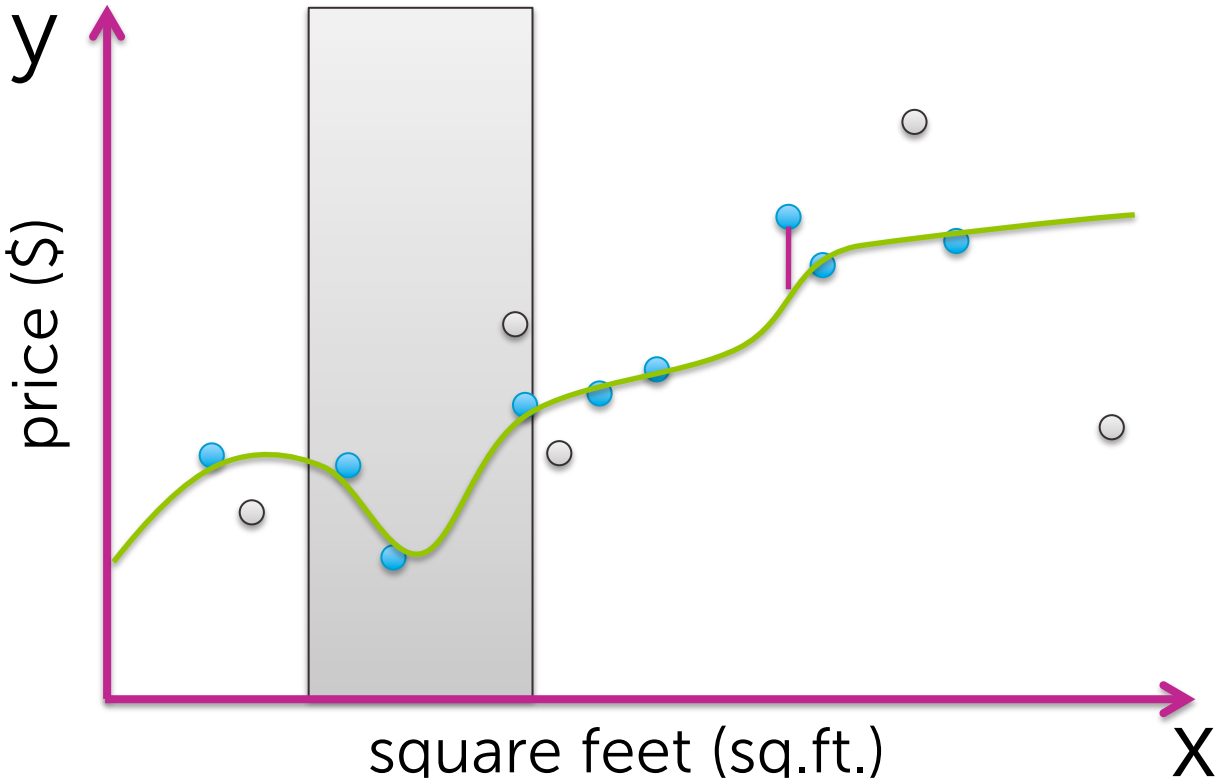


Training error vs. model complexity



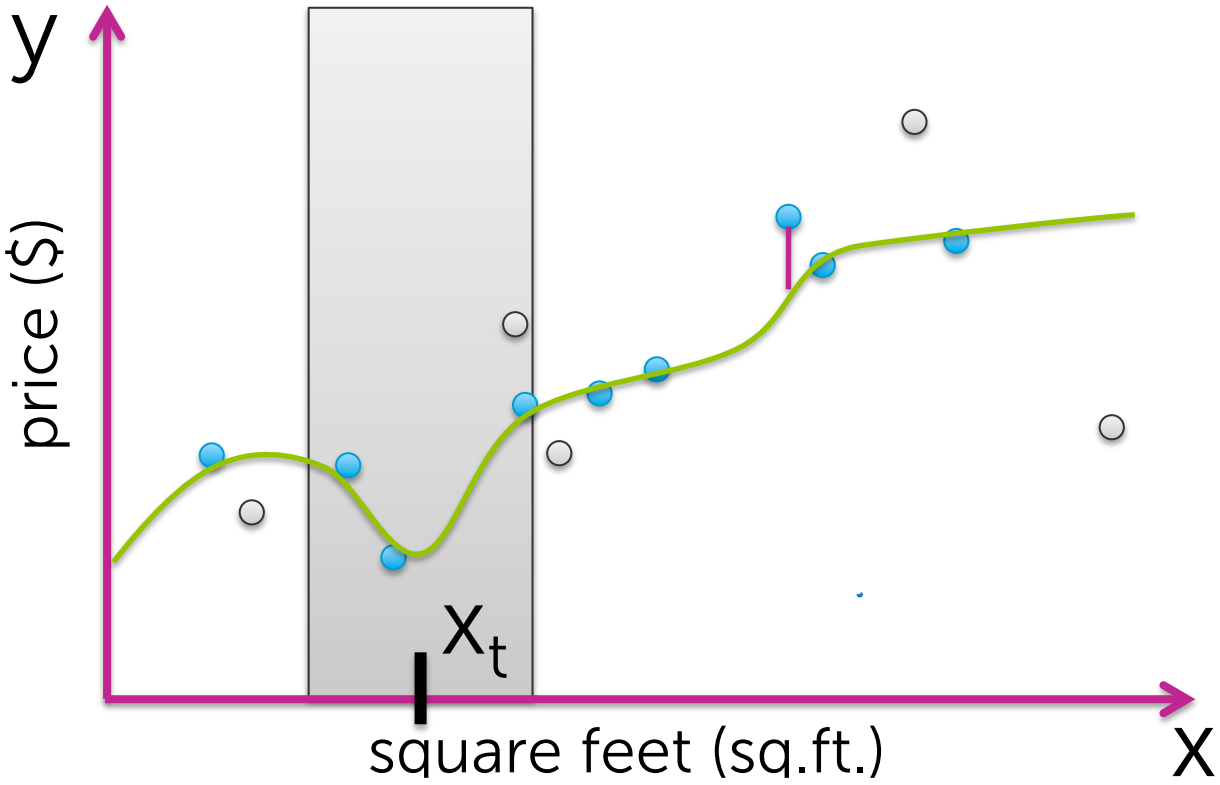
Is training error a good measure of predictive performance?

How do we expect to perform on a new house?



Is training error a good measure of predictive performance?

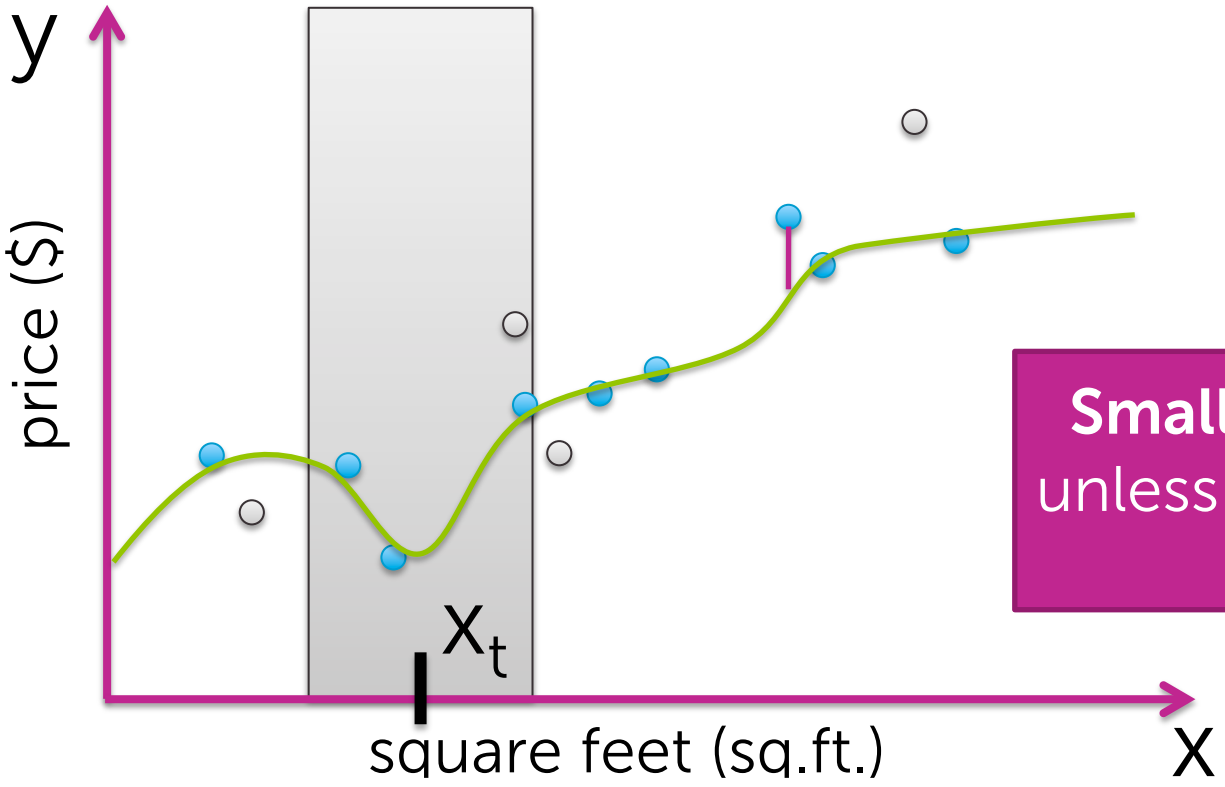
Is there something particularly bad about having x_t square feet???



Is training error a good measure of predictive performance?

Issue: Training error is overly optimistic

because $\hat{\mathbf{w}}$ was fit to training data



Small training error ≠> good predictions
unless training data includes everything you might ever see

Assessing the loss
Part 2: Generalization (true) error

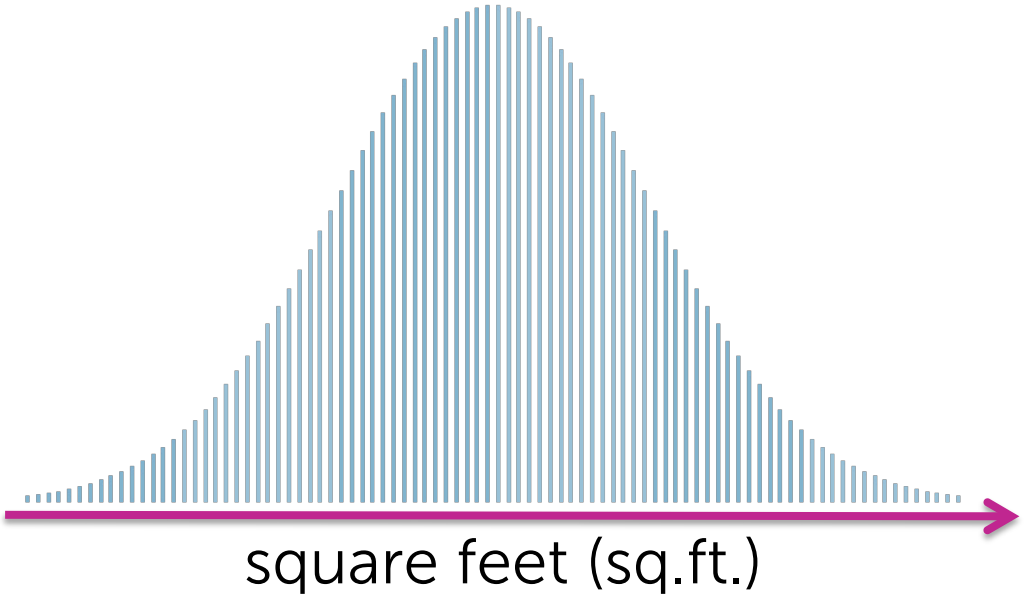
Generalization error

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

Lots of houses
in neighborhood,
but not in dataset

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

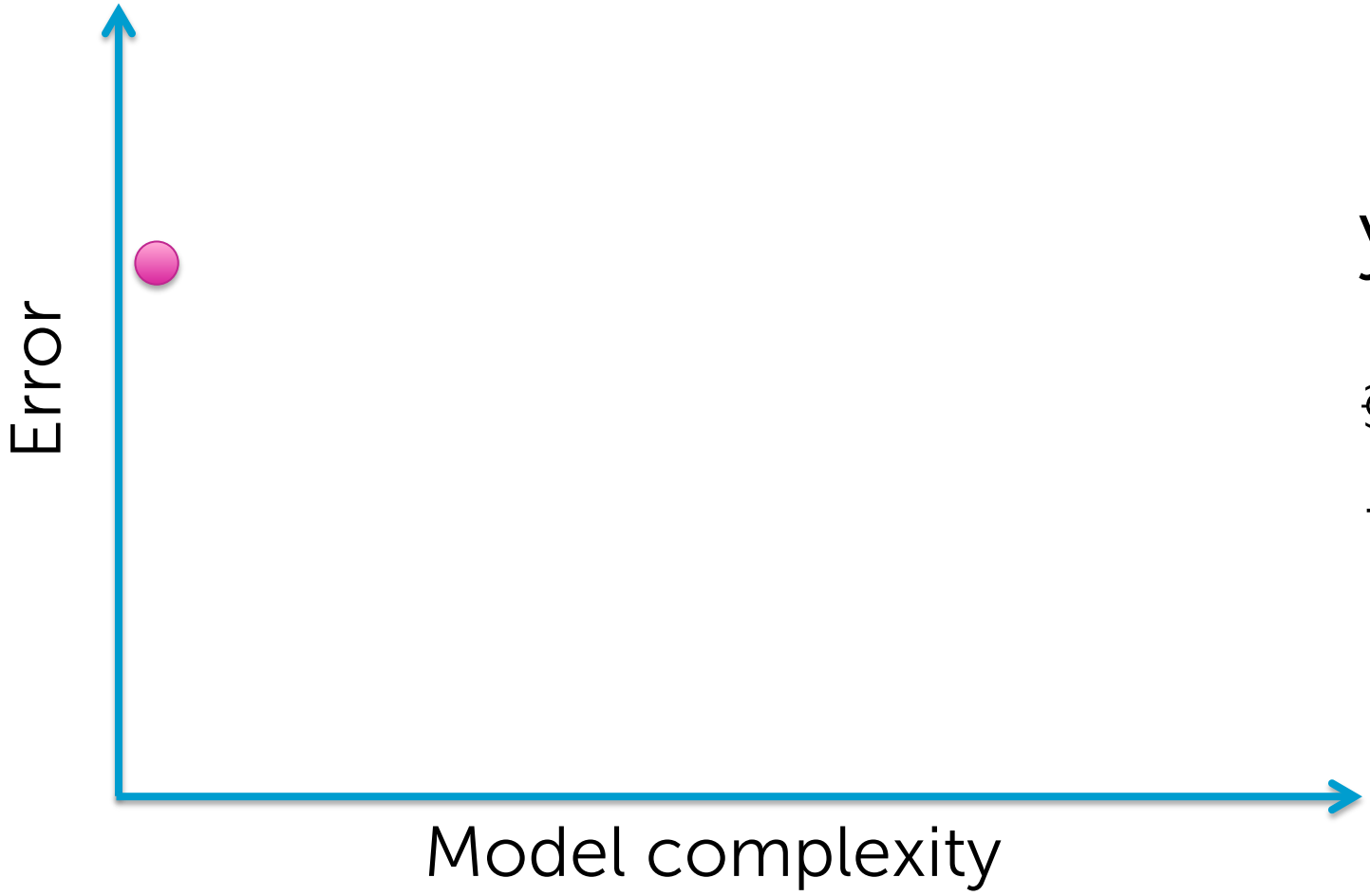
Formally:

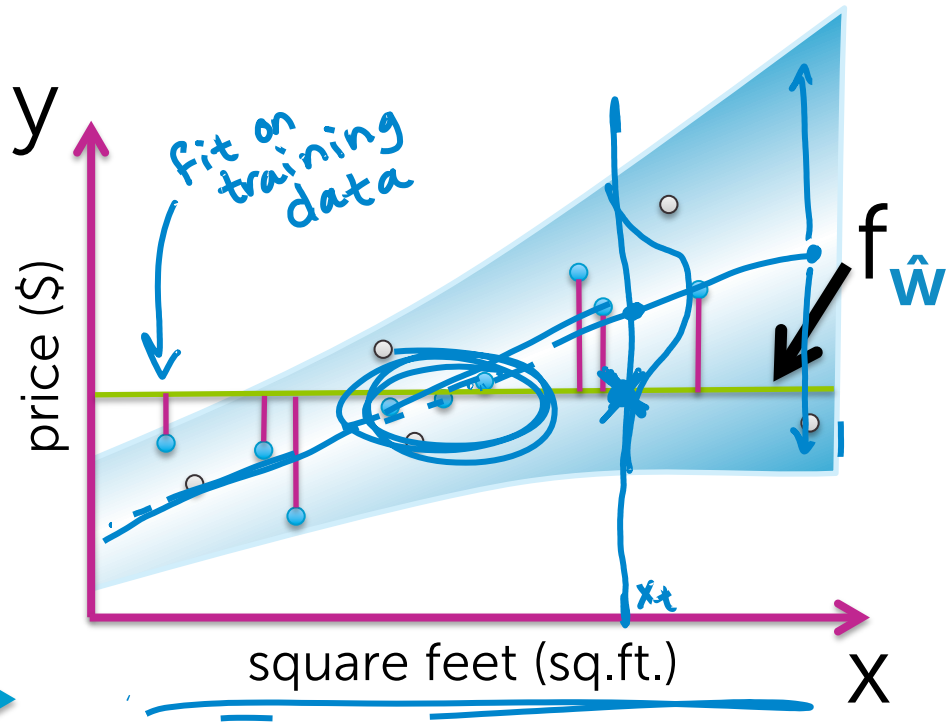
average over all possible (**x**,y) pairs weighted by how likely each is

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

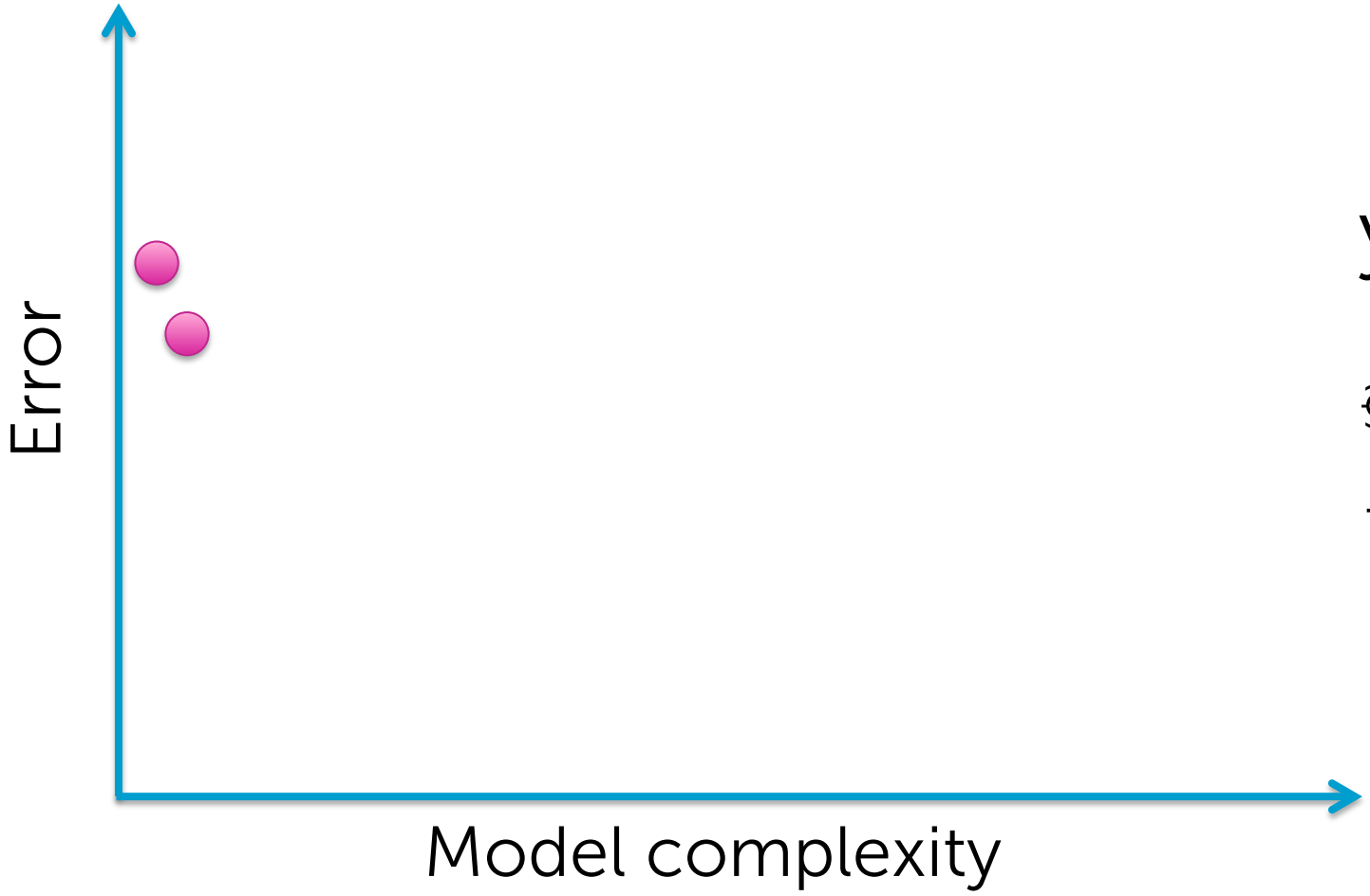
fit using training data

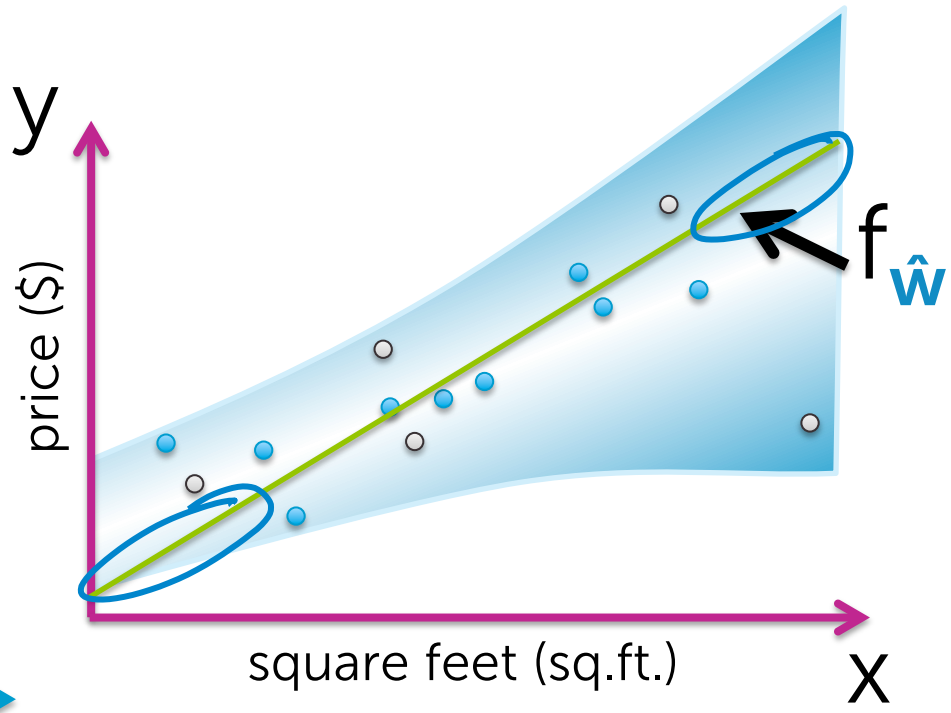
Generalization error vs. model complexity



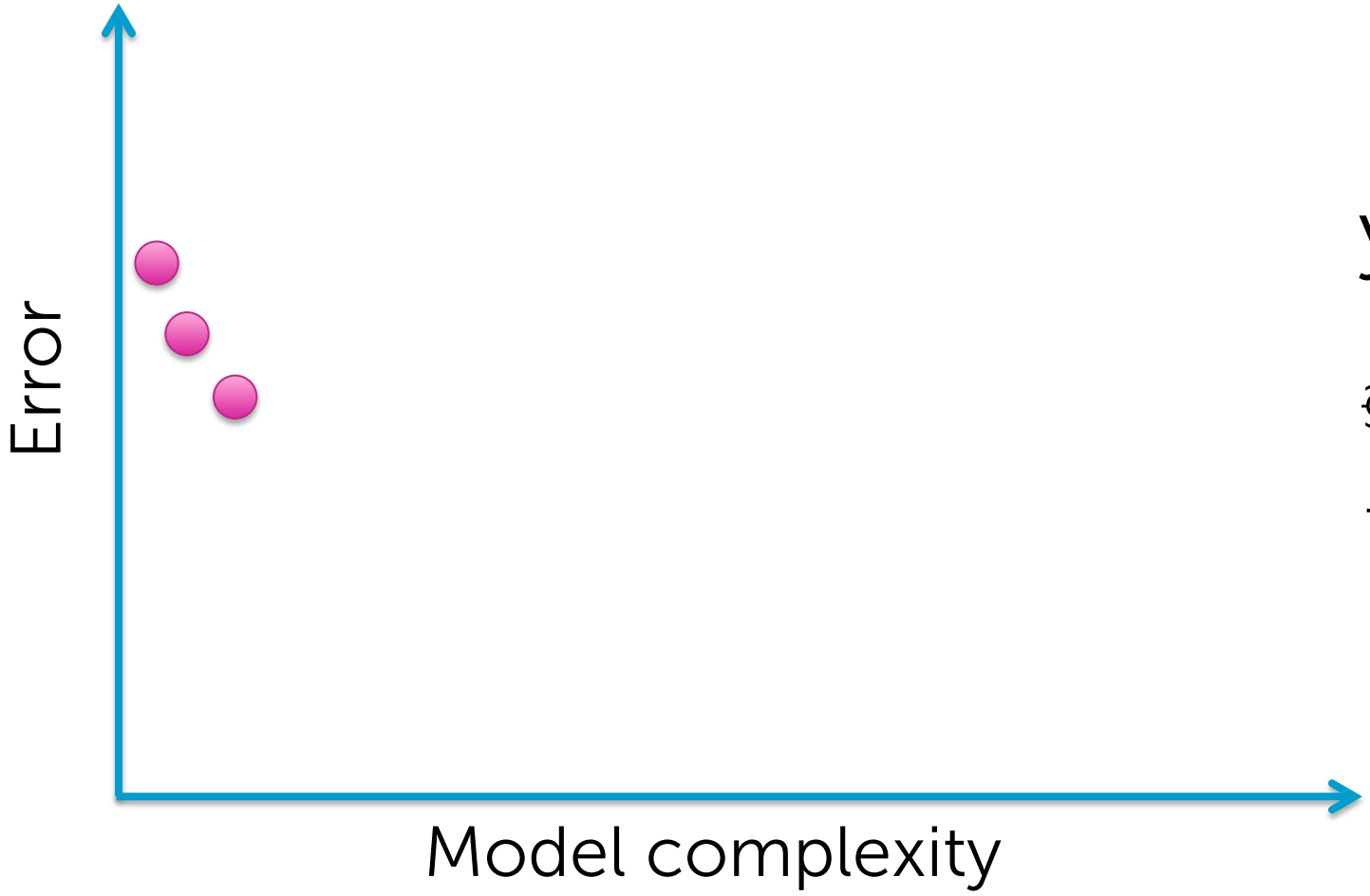


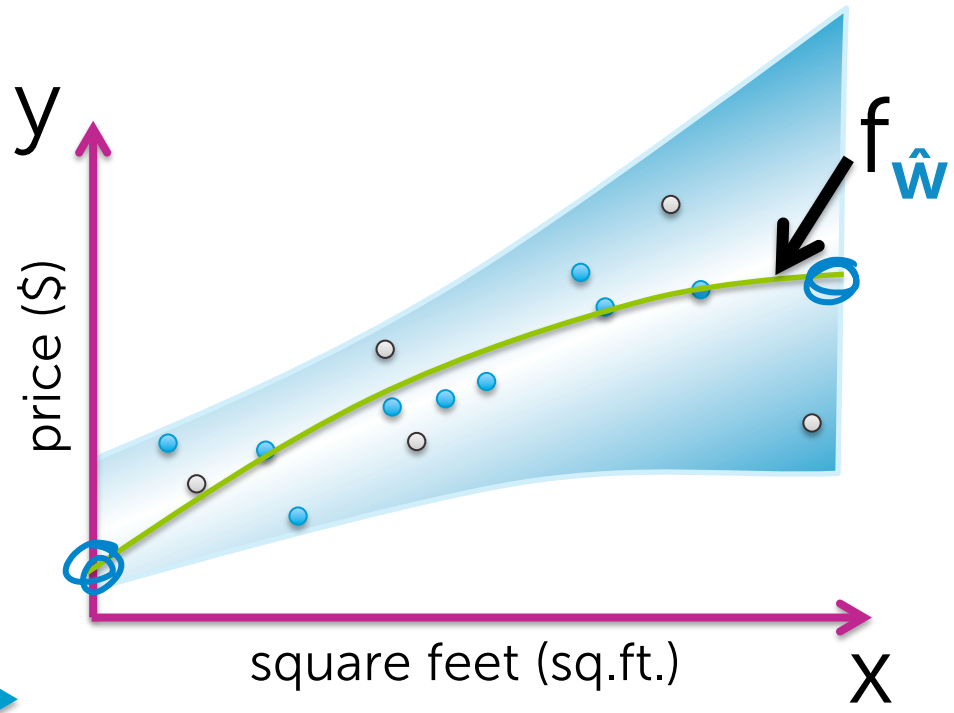
Generalization error vs. model complexity



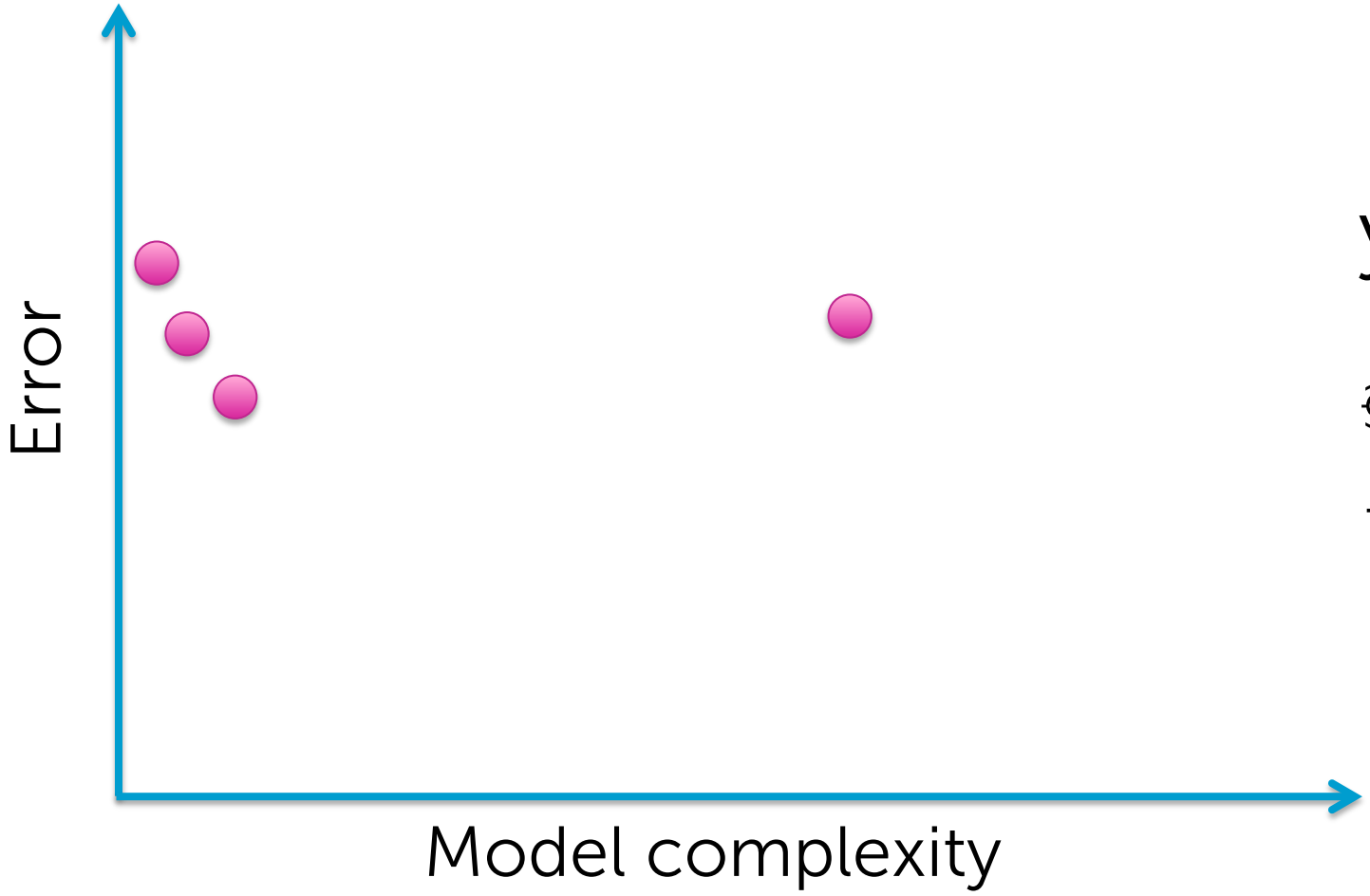


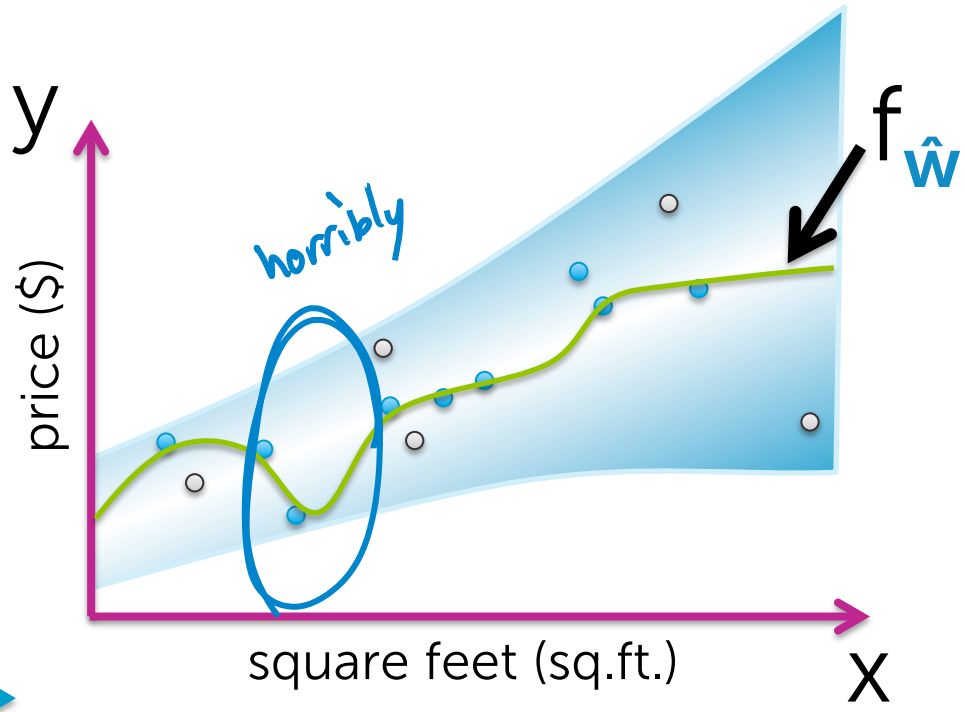
Generalization error vs. model complexity



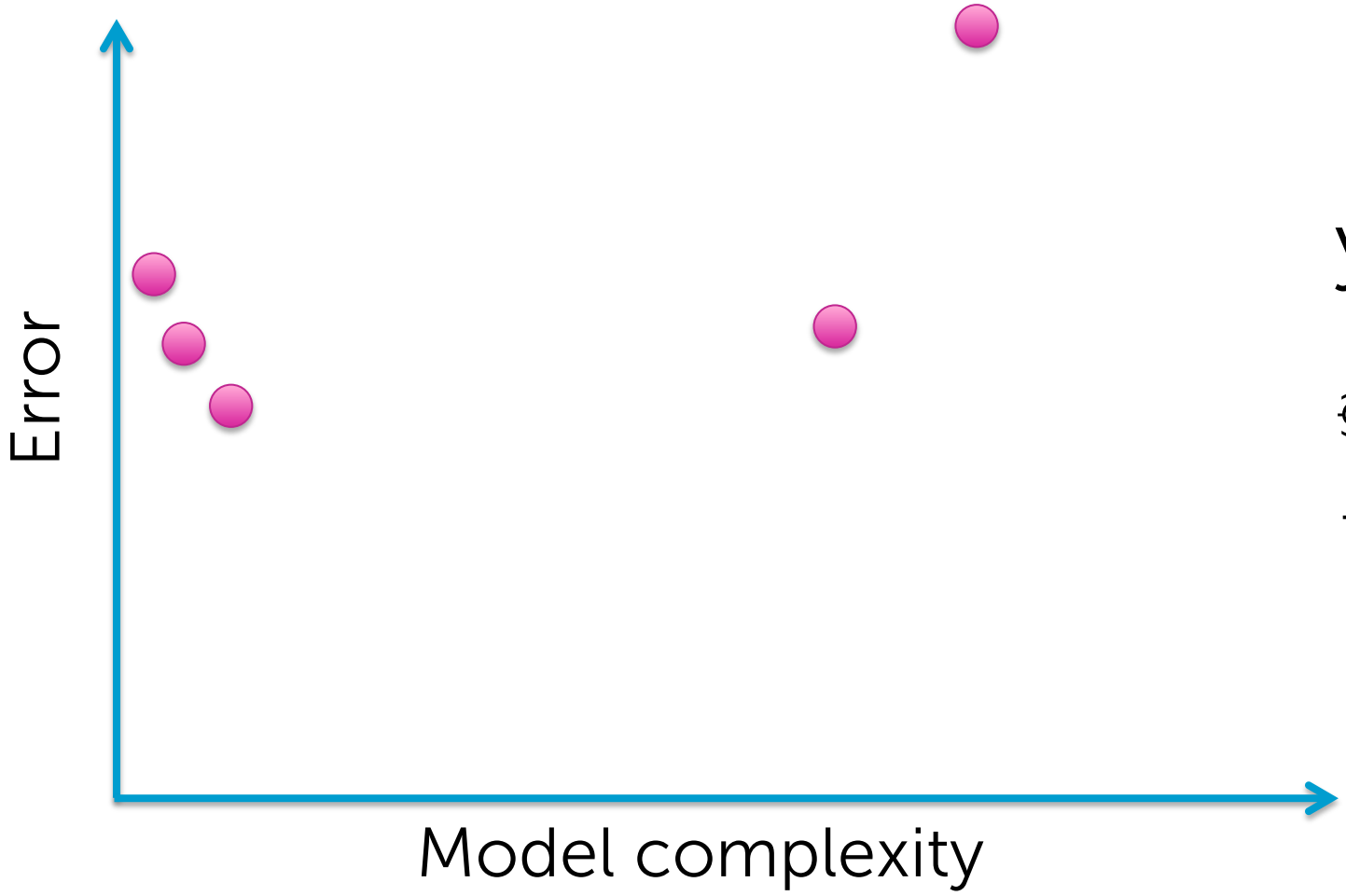


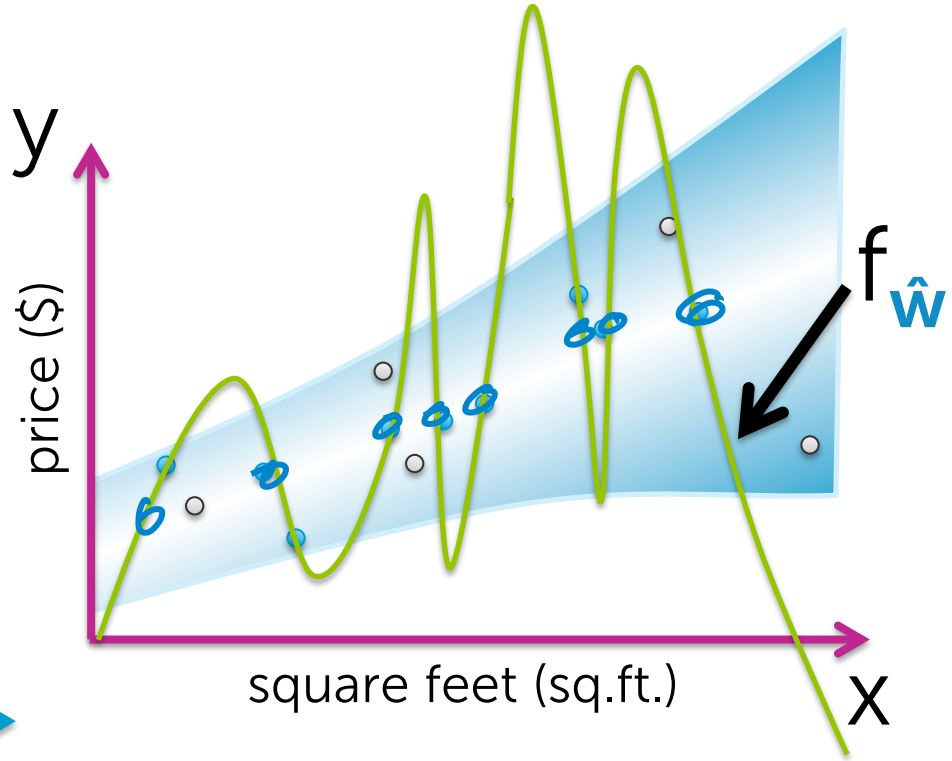
Generalization error vs. model complexity



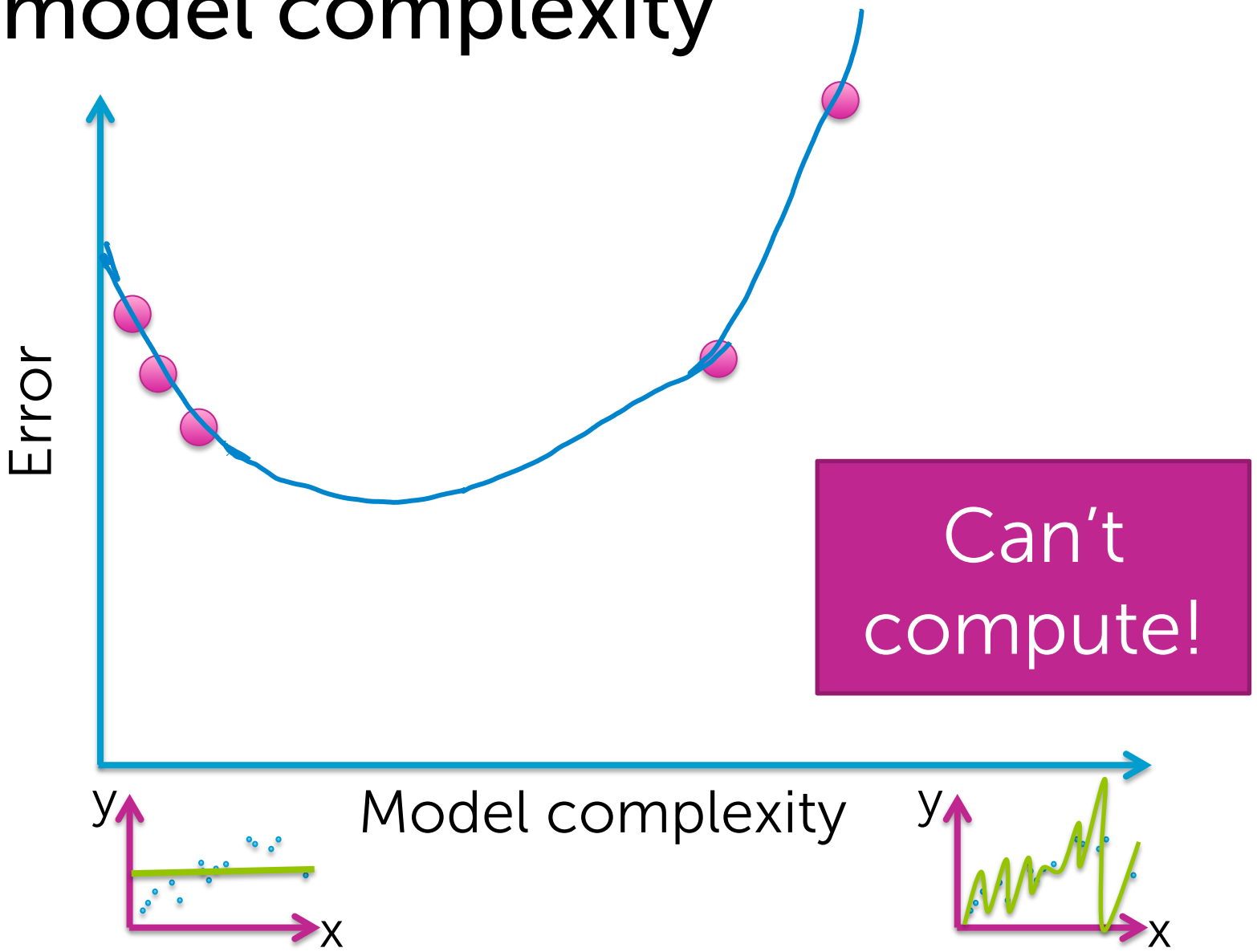


Generalization error vs. model complexity





Generalization error vs. model complexity



Can't compute!

Assessing the loss
Part 3: Test error

Approximating generalization error

Wanted estimate of loss over all possible (🏠,$) 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

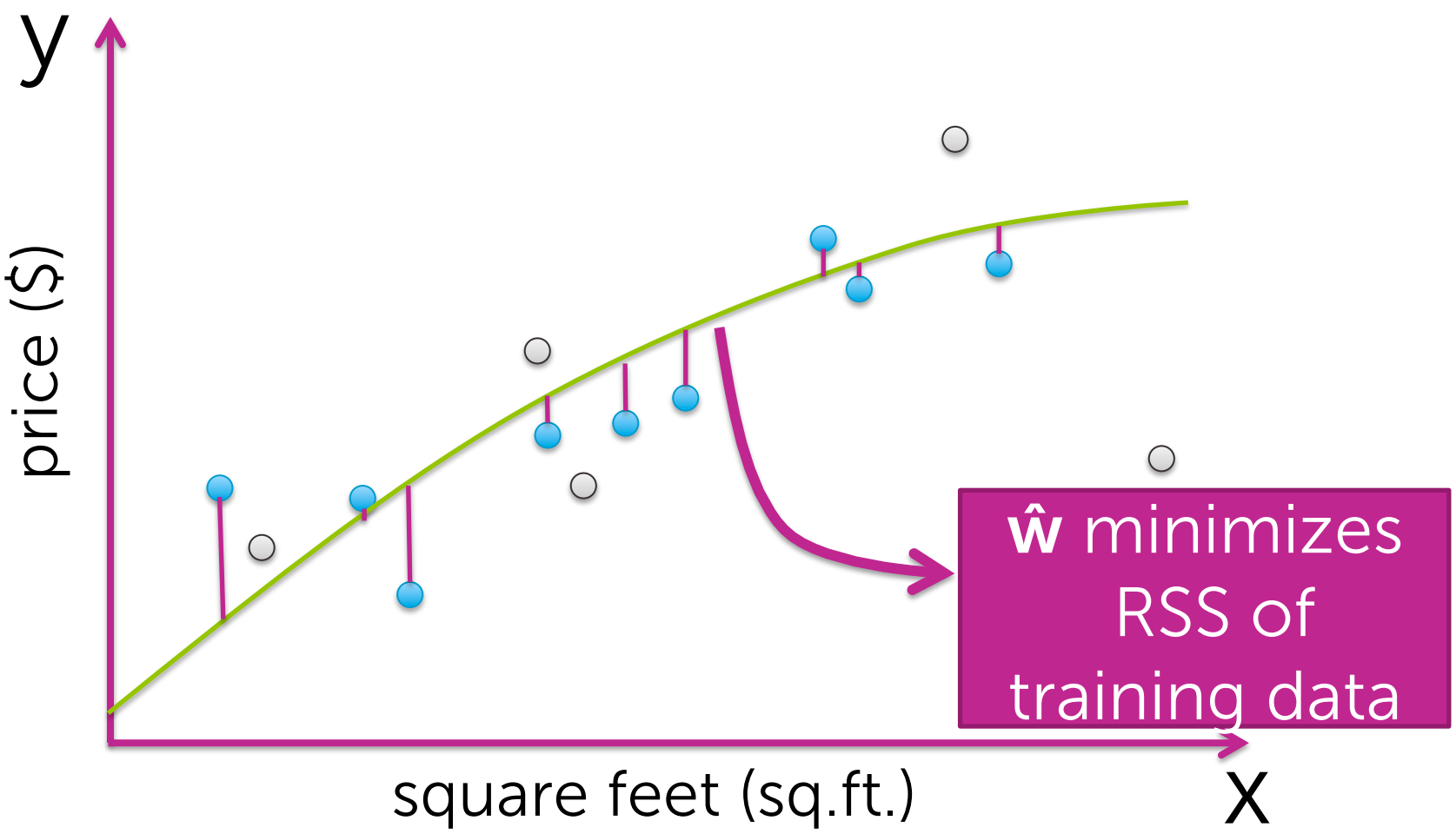
$$= \frac{1}{N_{test}} \sum_{i \text{ in test set}} L(y_i, f_{\hat{w}}(\mathbf{x}_i))$$

test points

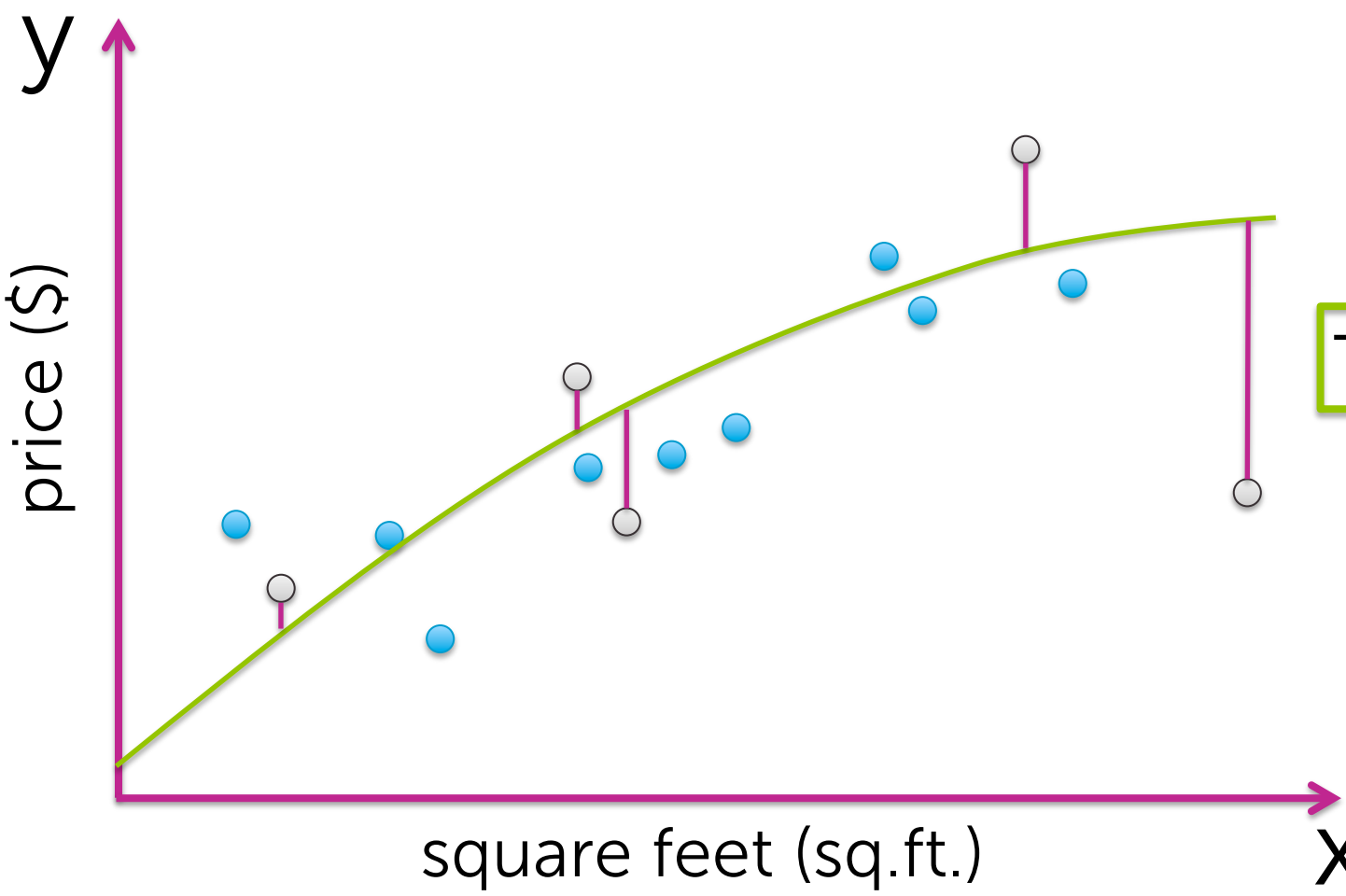
fit using training data

has never seen test data!

Example: As before, fit quadratic to training data



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



Test error ($\hat{w}$) = 1/N *

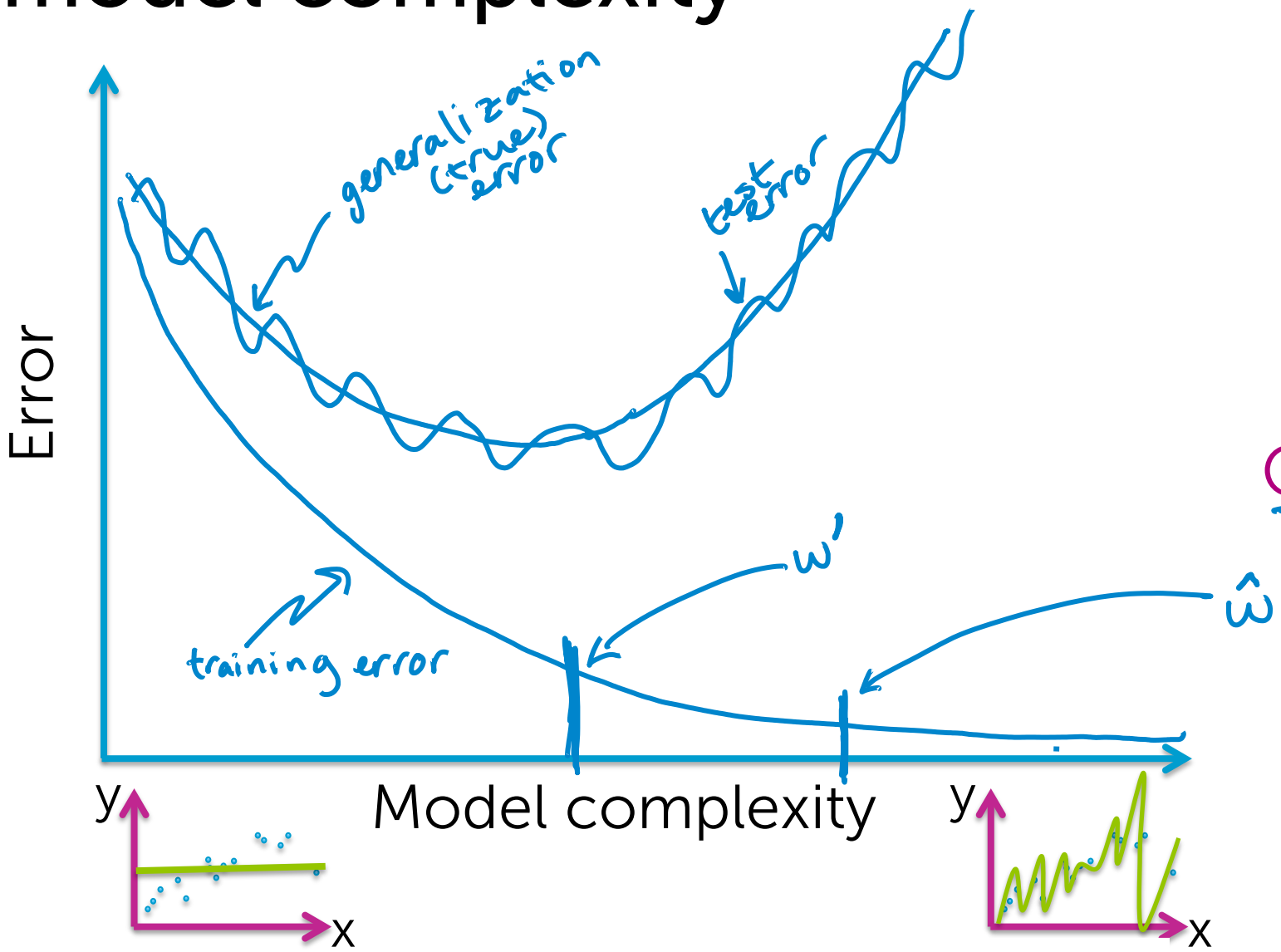
$[(\$_{test\ 1}-f_{\hat{w}}(sq.ft._{test\ 1}))^2$

$+ (\$_{test\ 2}-f_{\hat{w}}(sq.ft._{test\ 2}))^2$

$+ (\$_{test\ 3}-f_{\hat{w}}(sq.ft._{test\ 3}))^2$

+ … include all test houses]

Training, true, & test error vs. model complexity



Overfitting if:

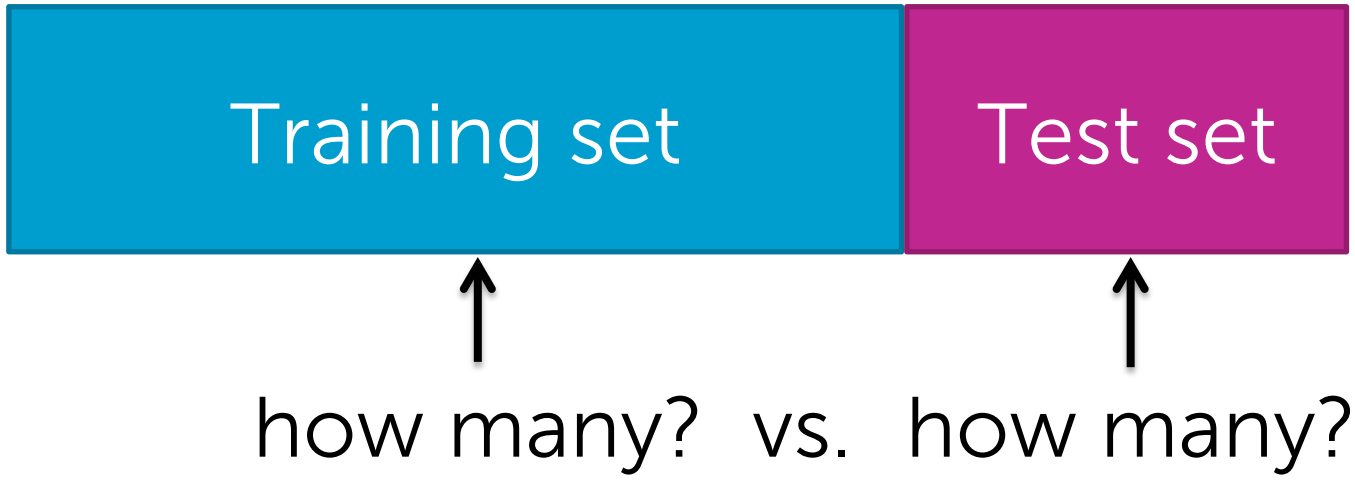
If there exists a model with estimated params w' such that

① training error($\hat{w}$) < training error(w')

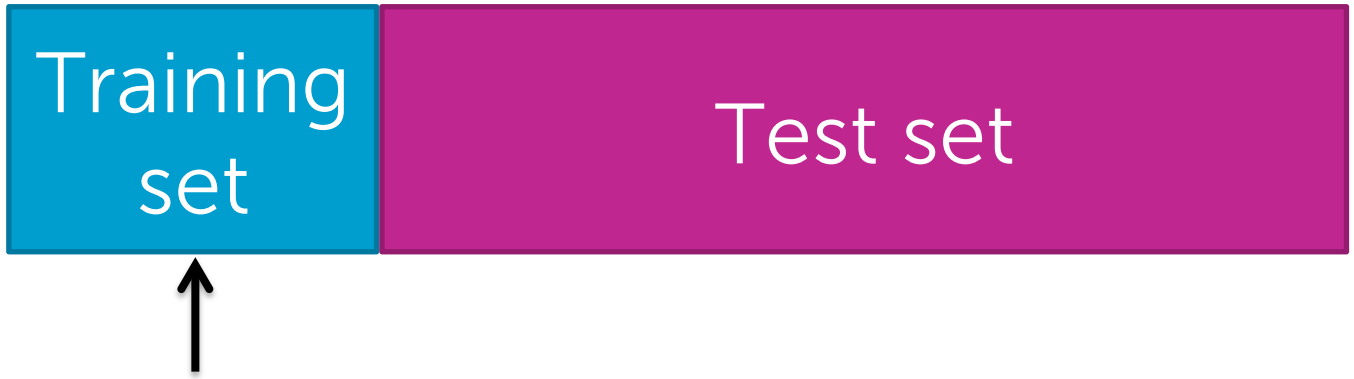
② true error($\hat{w}$) > true error(w')

Training/test split

Training/test splits



Training/test splits



Too few → $\hat{w}$ 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…)

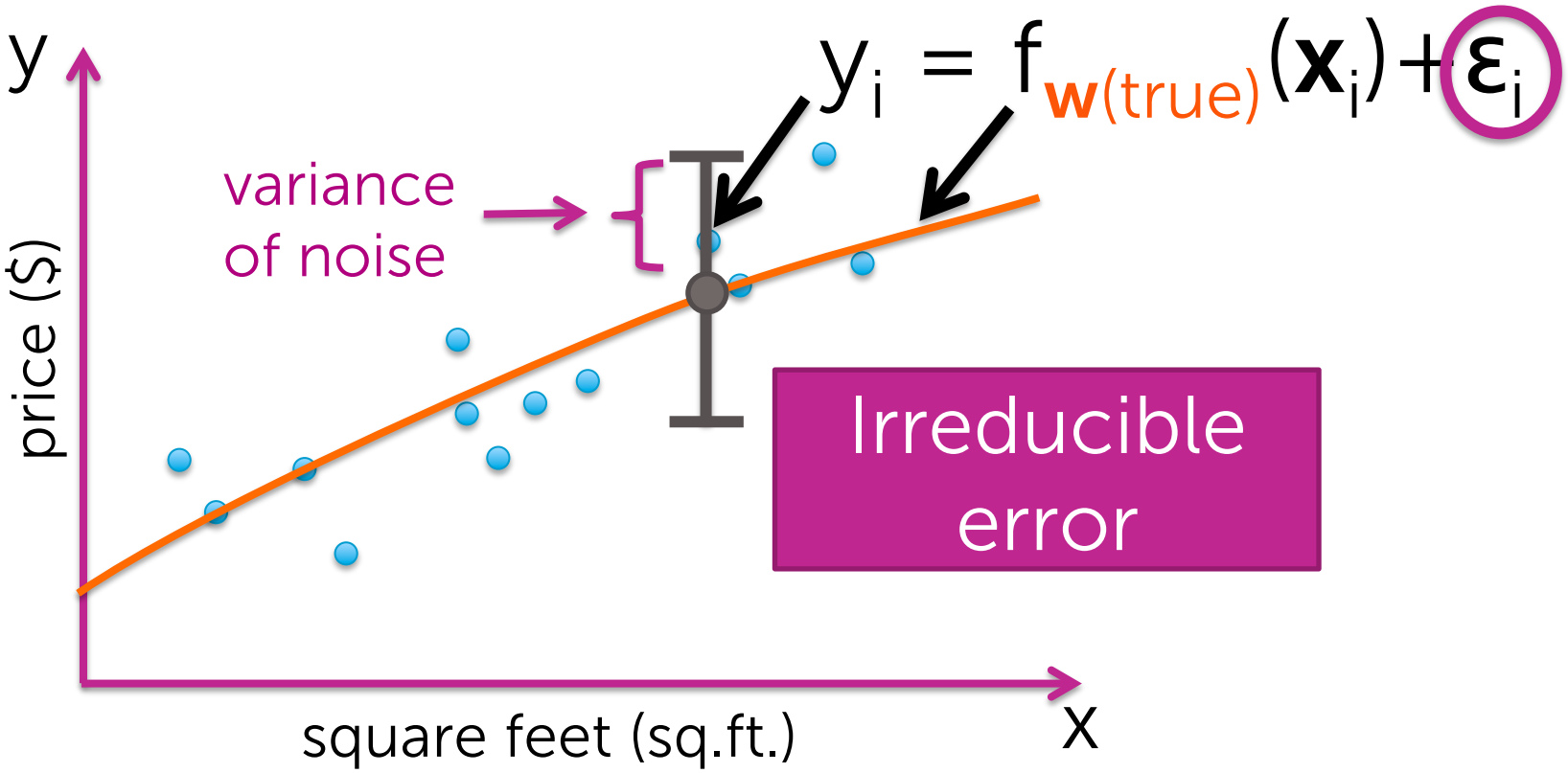
3 sources of error + the bias-variance tradeoff

3 sources of error

In forming predictions, there are 3 sources of error:

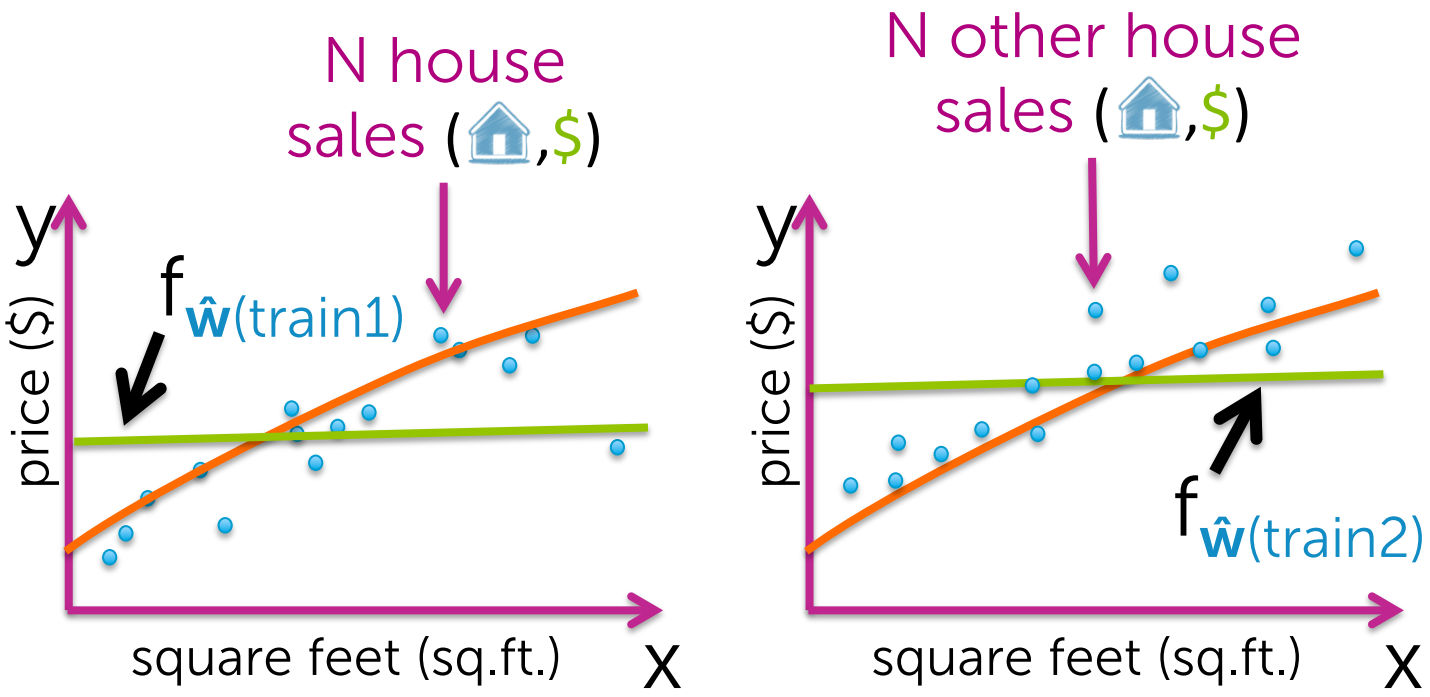
1. Noise
2. Bias
3. Variance

Data inherently noisy



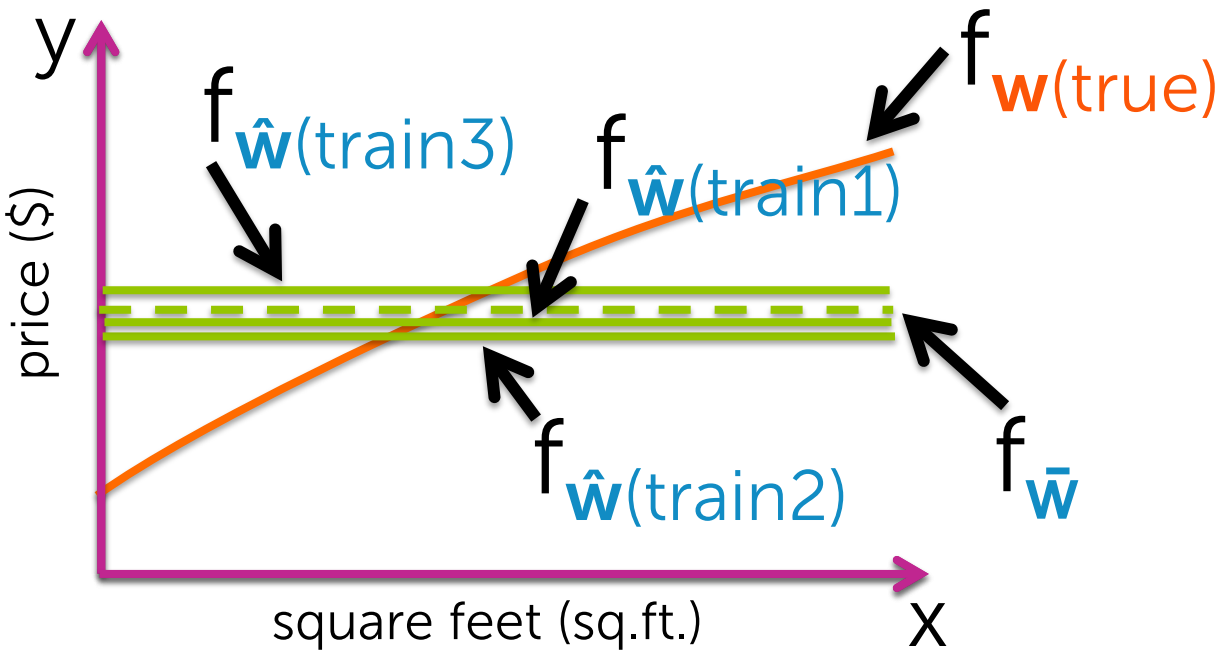
Bias contribution

Assume we fit a constant function



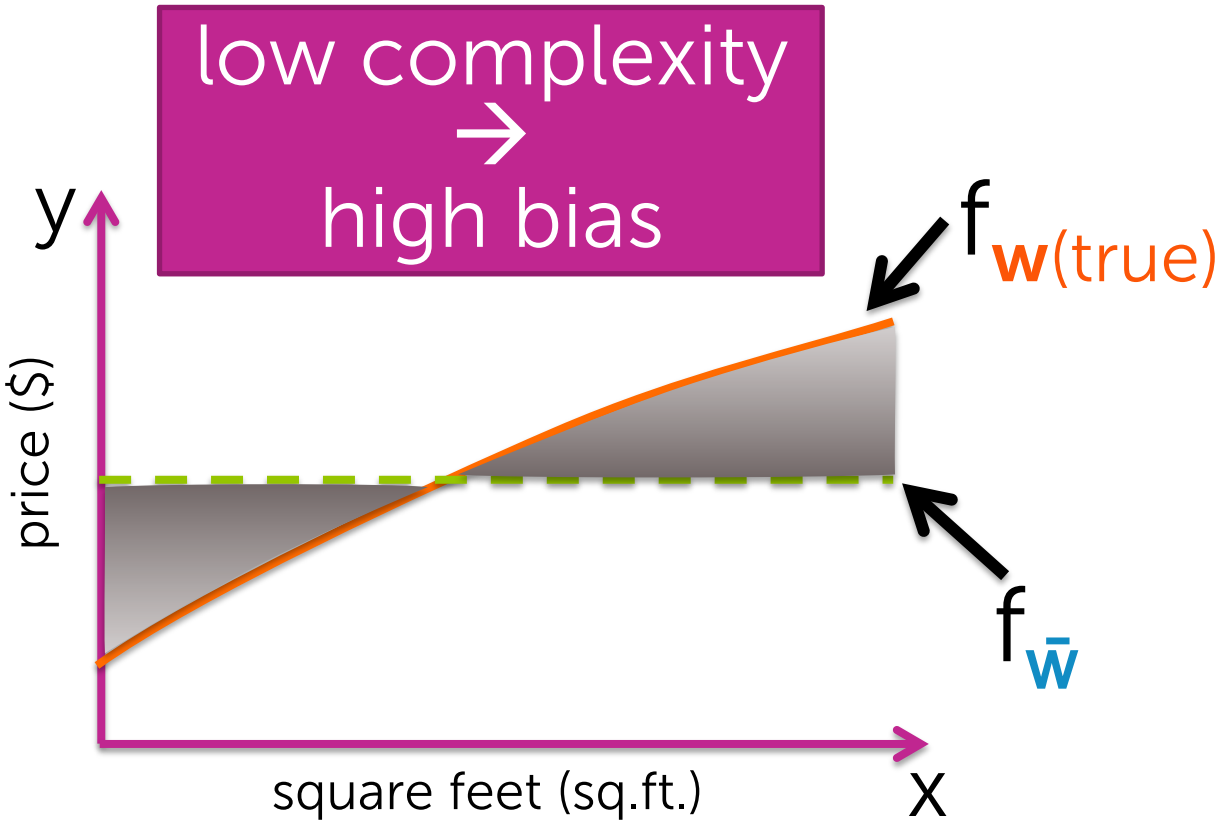
Bias contribution

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



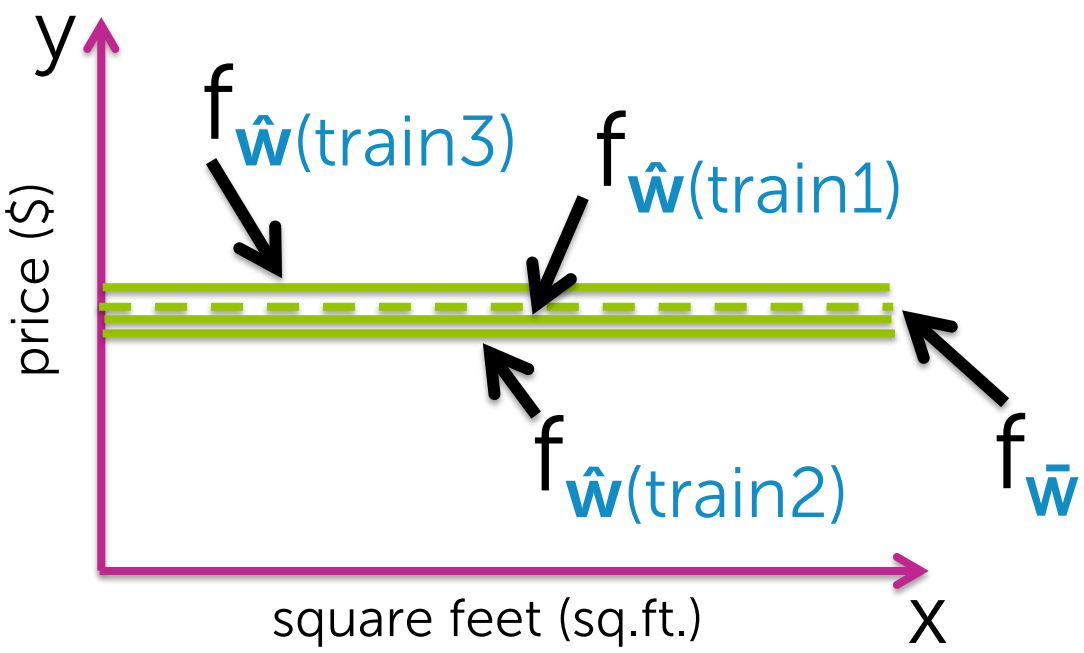
Bias contribution

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



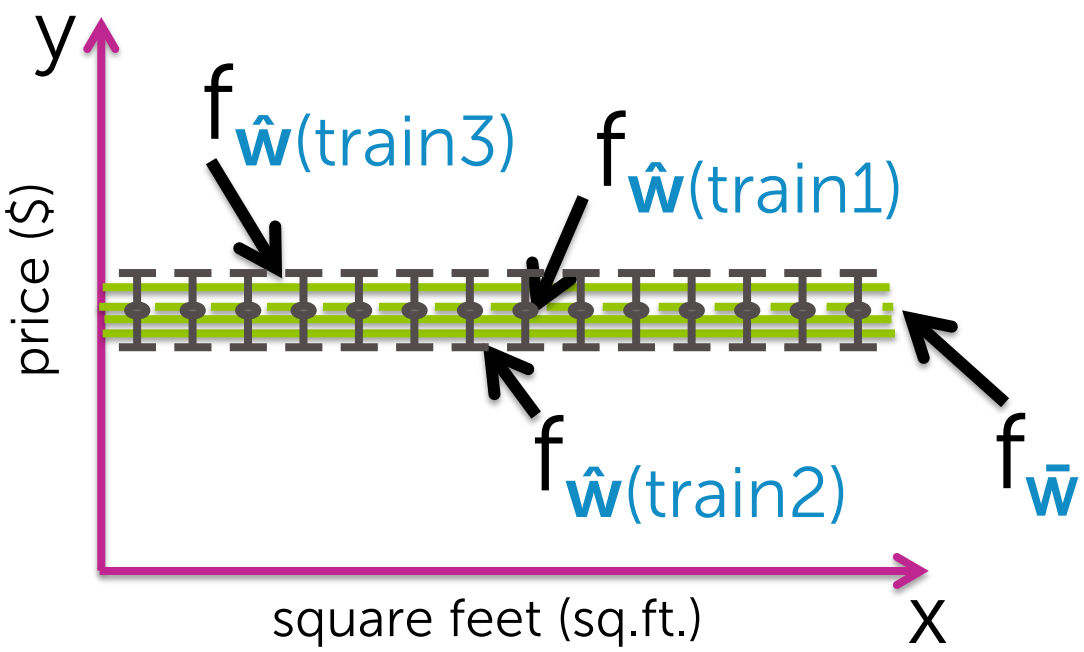
Variance contribution

How much do specific fits vary from the expected fit?



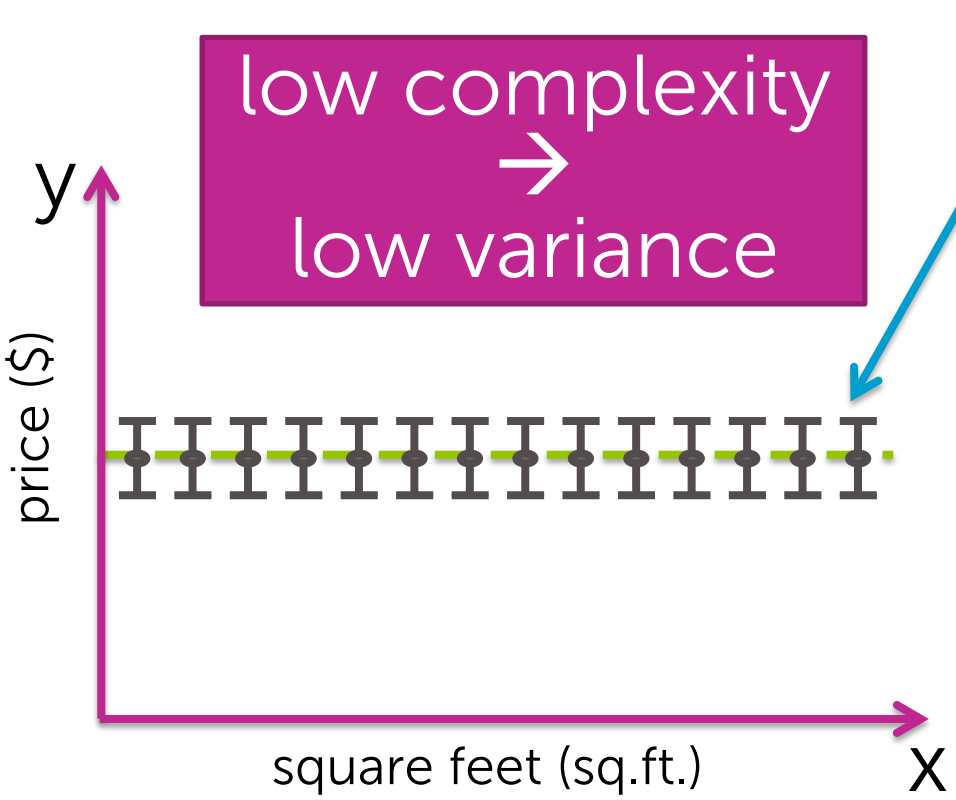
Variance contribution

How much do specific fits vary from the expected fit?



Variance contribution

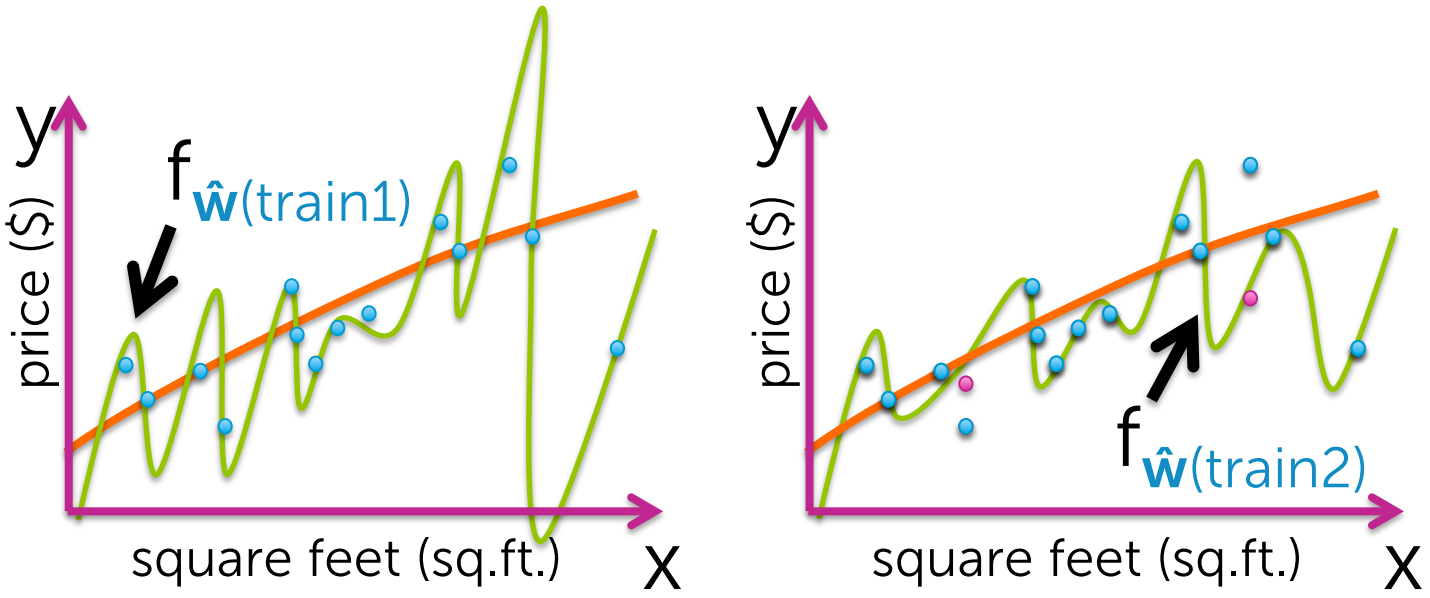
How much do specific fits vary from the expected fit?



Can specific fits vary widely?
If so, erratic predictions

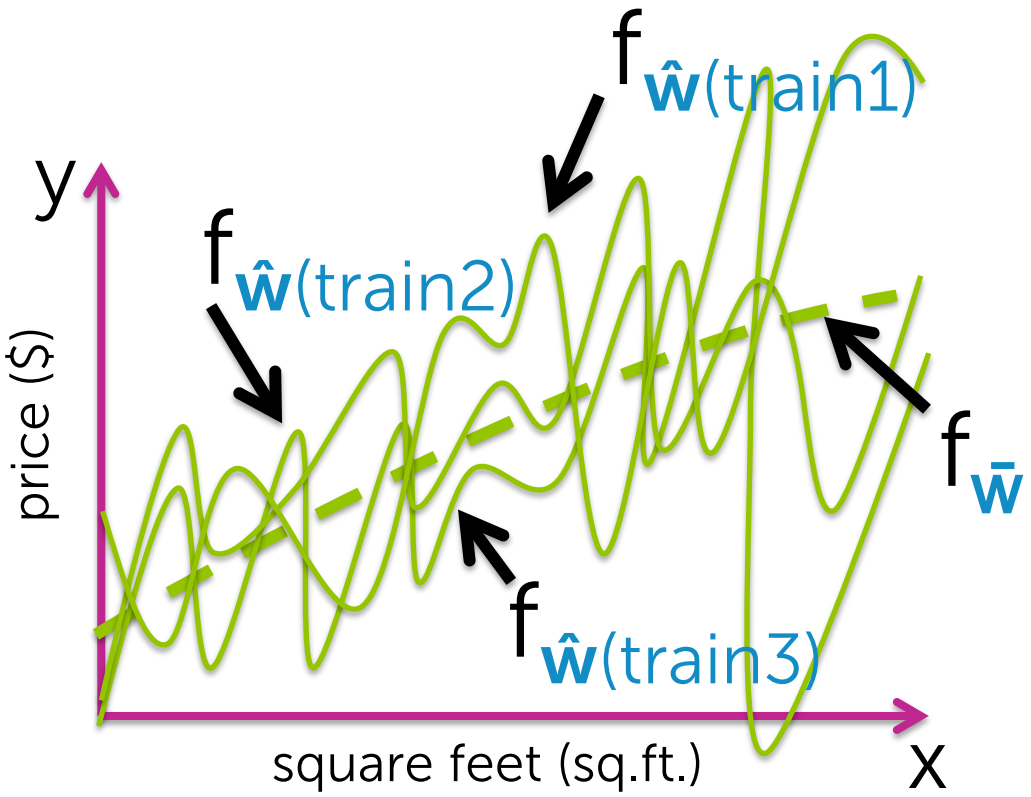
Variance of high-complexity models

Assume we fit a high-order polynomial

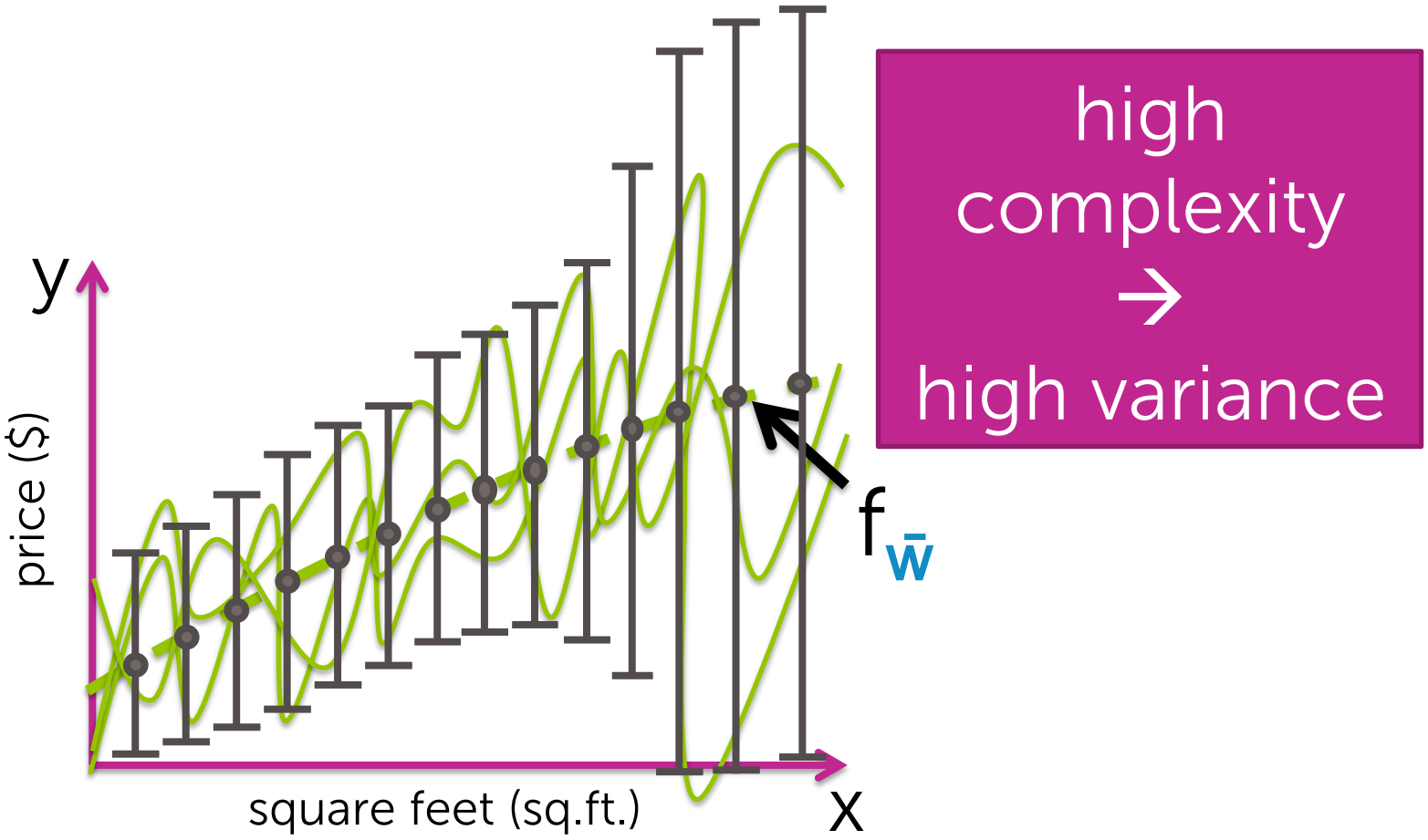


Variance of high-complexity models

Assume we fit a high-order polynomial

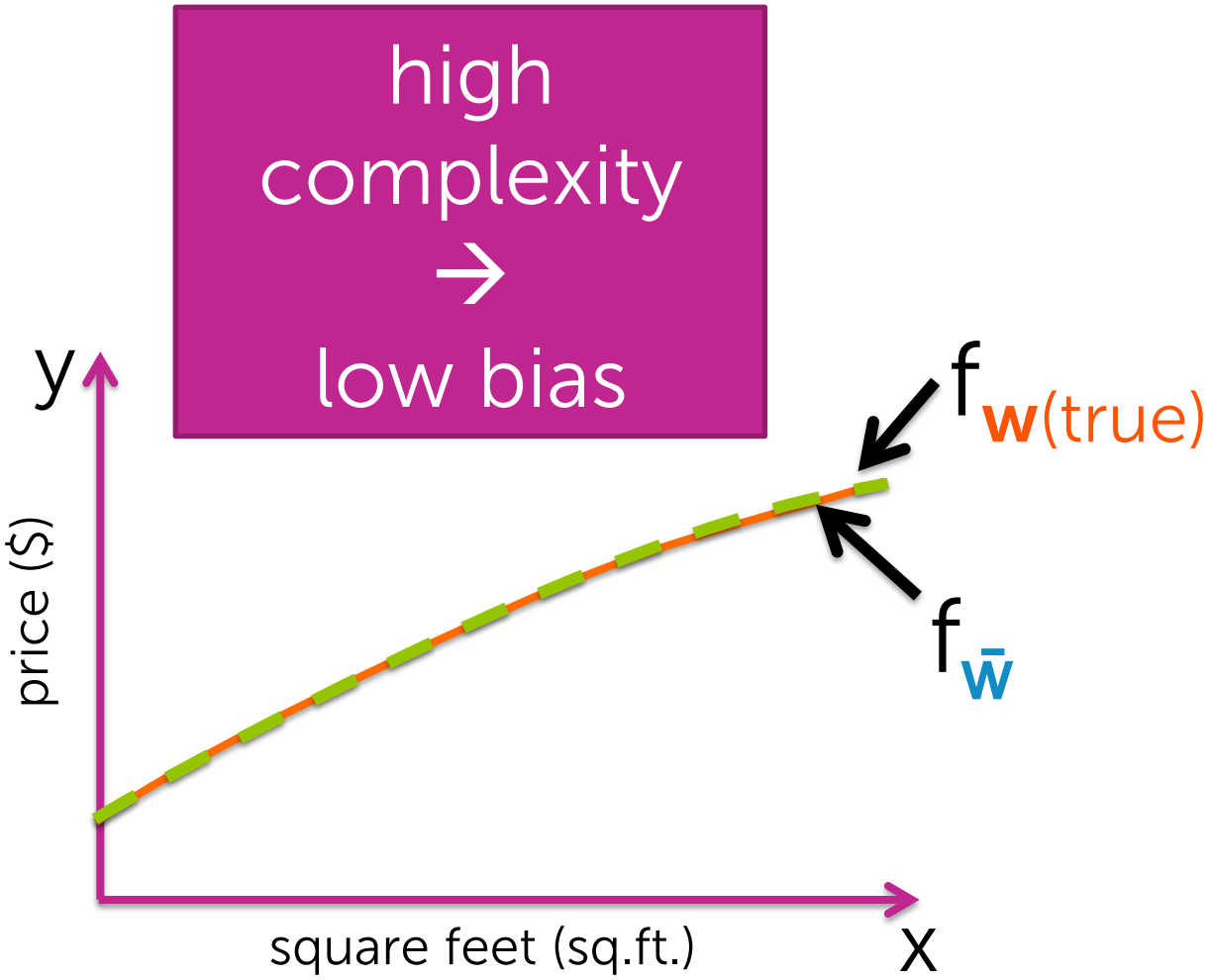


Variance of high-complexity models

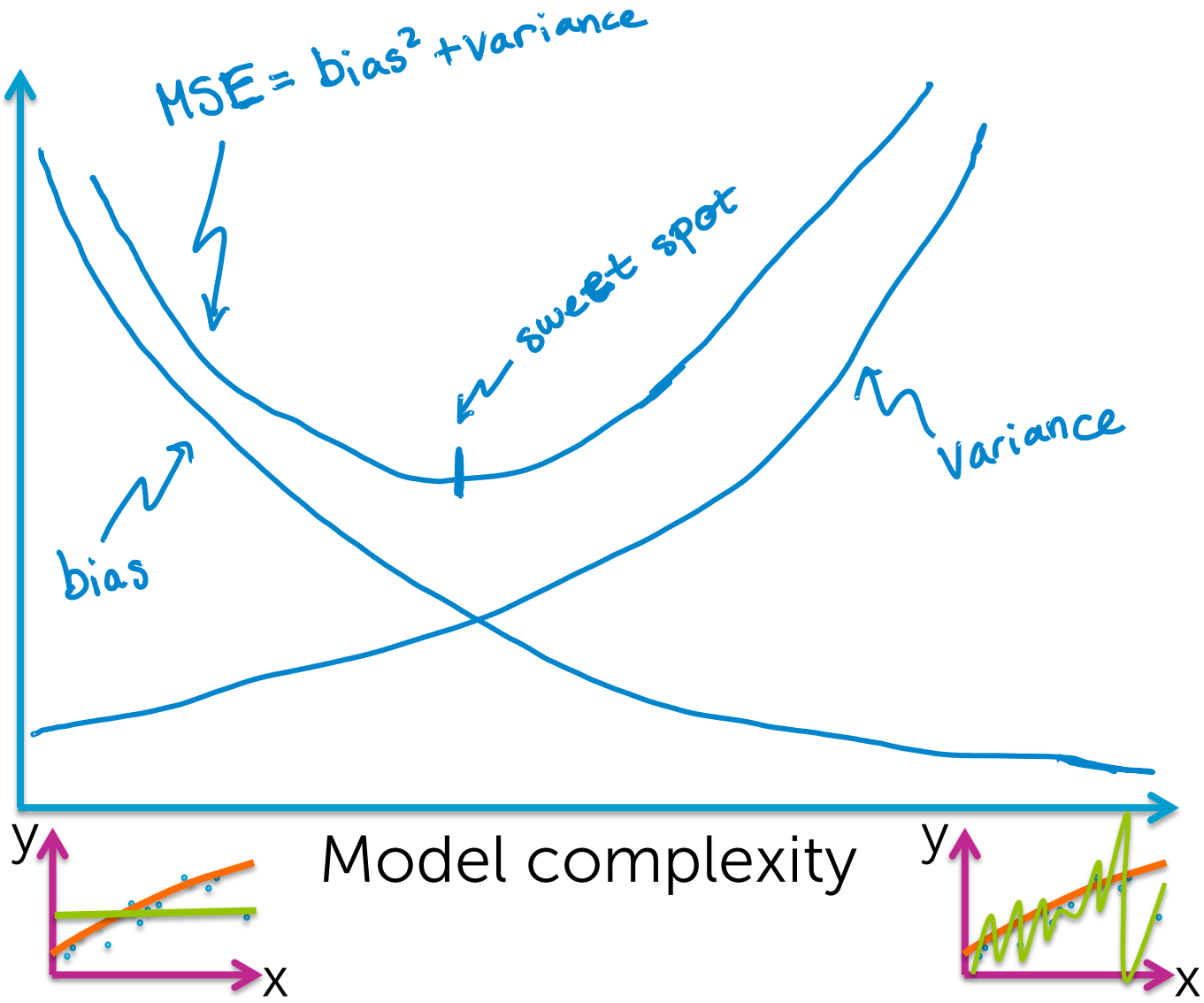


high complexity → high variance

Bias of high-complexity models

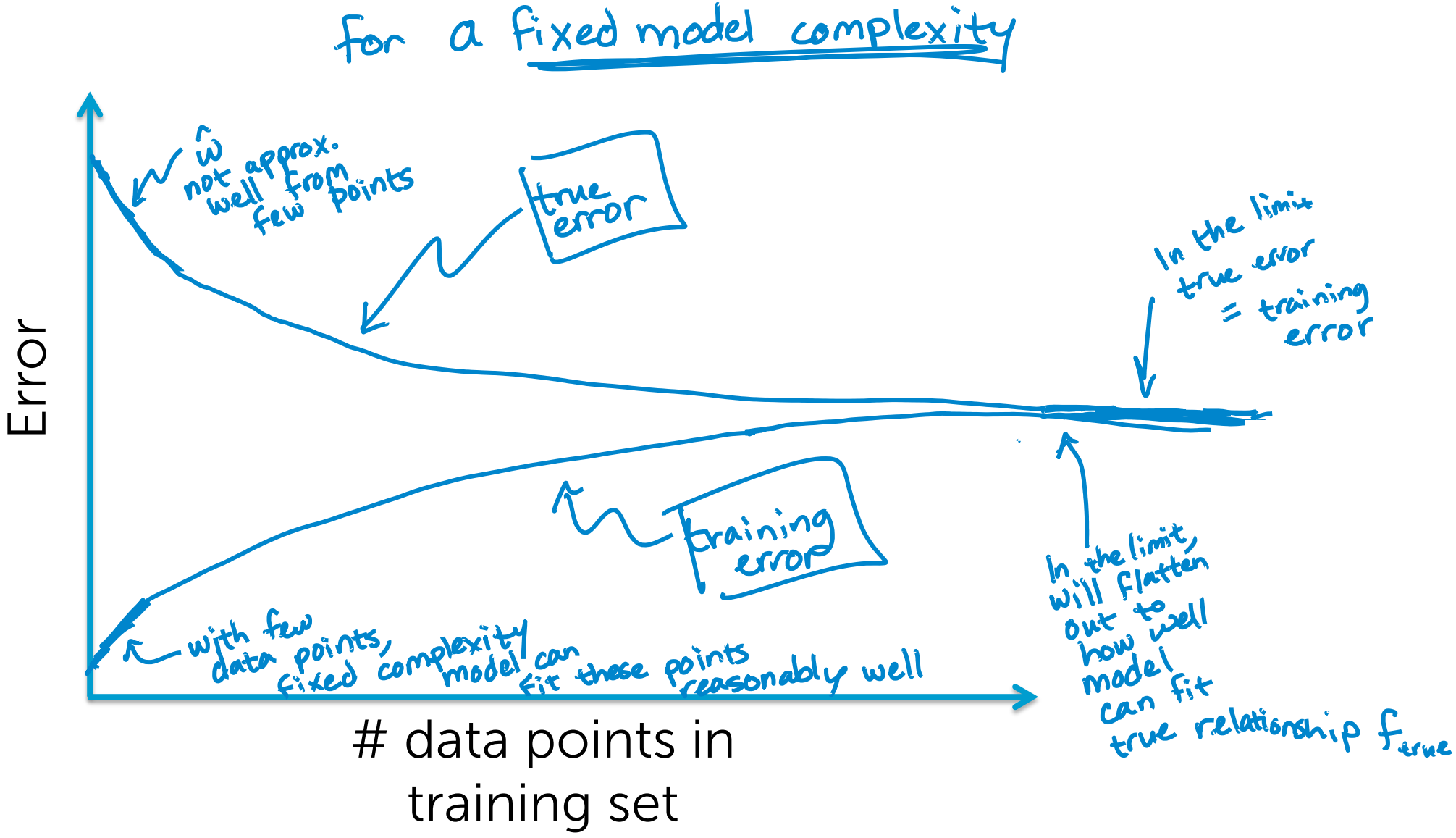


Bias-variance tradeoff



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

Error vs. amount of data



Summary of tasks

The regression/ML workflow

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

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

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

Overly optimistic!

2. **Model assessment**

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

Hypothetical implementation

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

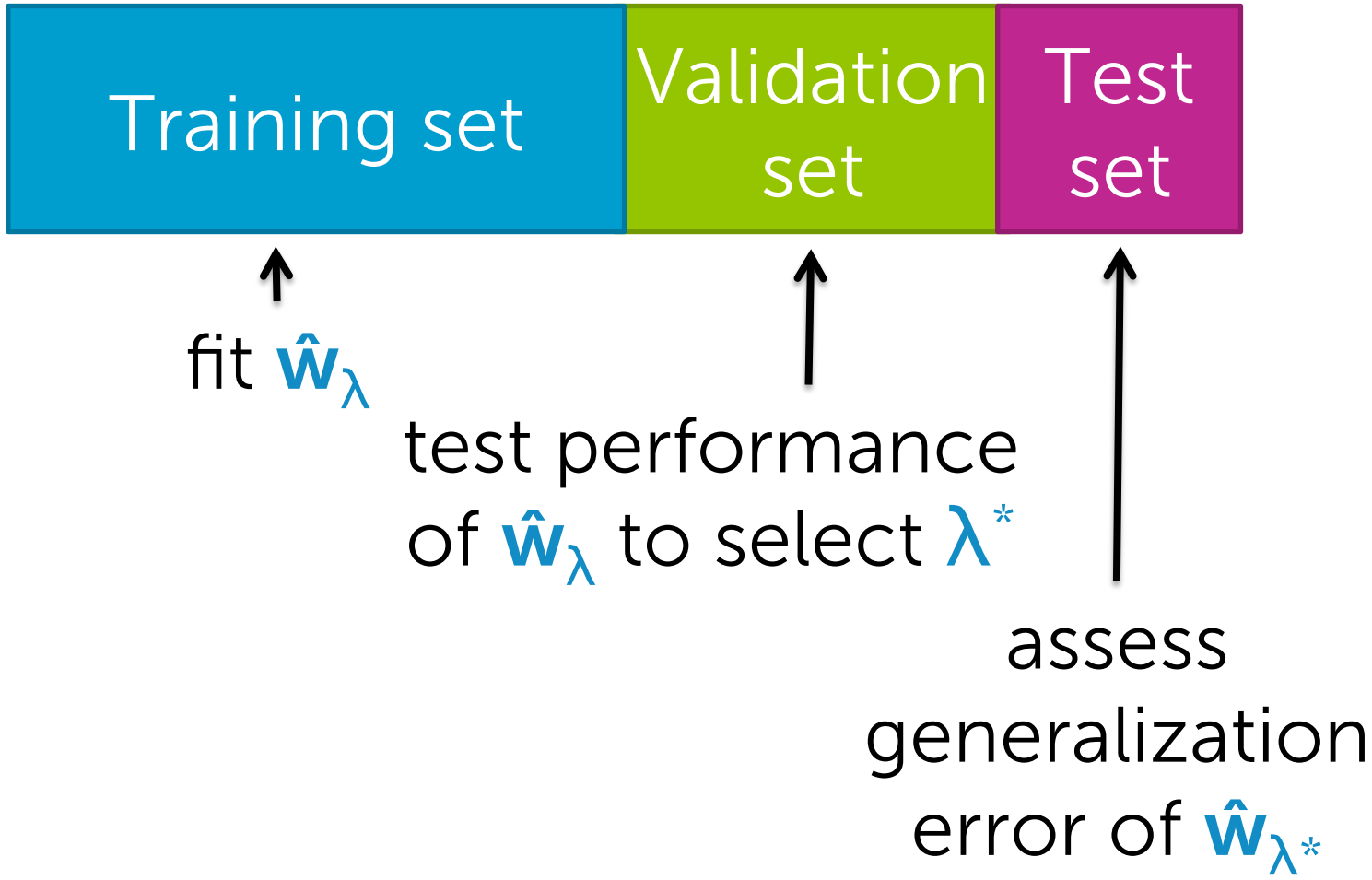
- λ^* was selected to minimize test error (i.e., λ^* was fit on test data)
- If test data is not representative of the whole world, then $\hat{\mathbf{w}}_{\lambda^*}$ will typically perform worse than test error indicates

Practical implementation

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



Typical splits

Training set	Validation set	Test set
80%	10%	10%
50%	25%	25%

Evaluating classifiers:

Precision & Recall

Using reviews to promote my restaurant

Goal: increase # guests by 30%

Reviews

Need an automated, "authentic" marketing campaign

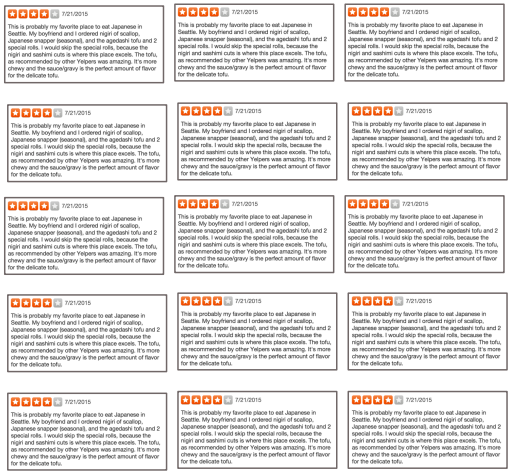
Great quotes

"Easily best sushi in city."

Great spokespeople

How do I find sentences with positive sentiment?

All reviews
for my restaurant



Intelligent restaurant review system

All reviews for restaurant

Break all reviews into sentences

The seaweed salad was just OK, vegetable salad was just ordinary.

I like the interior decoration and the blackboard menu on the wall.

All the sushi was delicious.

My wife tried their ramen and it was pretty forgettable.

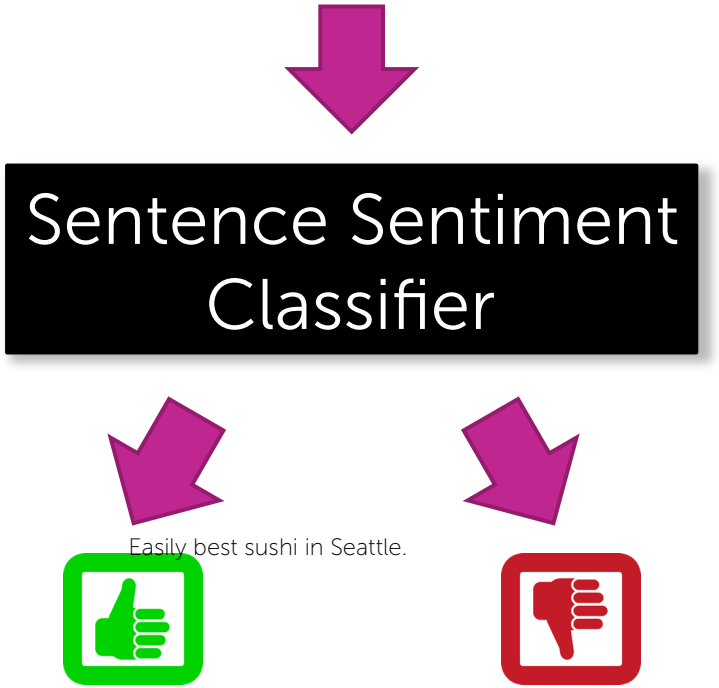
The sushi was amazing, and the rice is just outstanding.

The service is somewhat hectic.

Easily best sushi in Seattle.

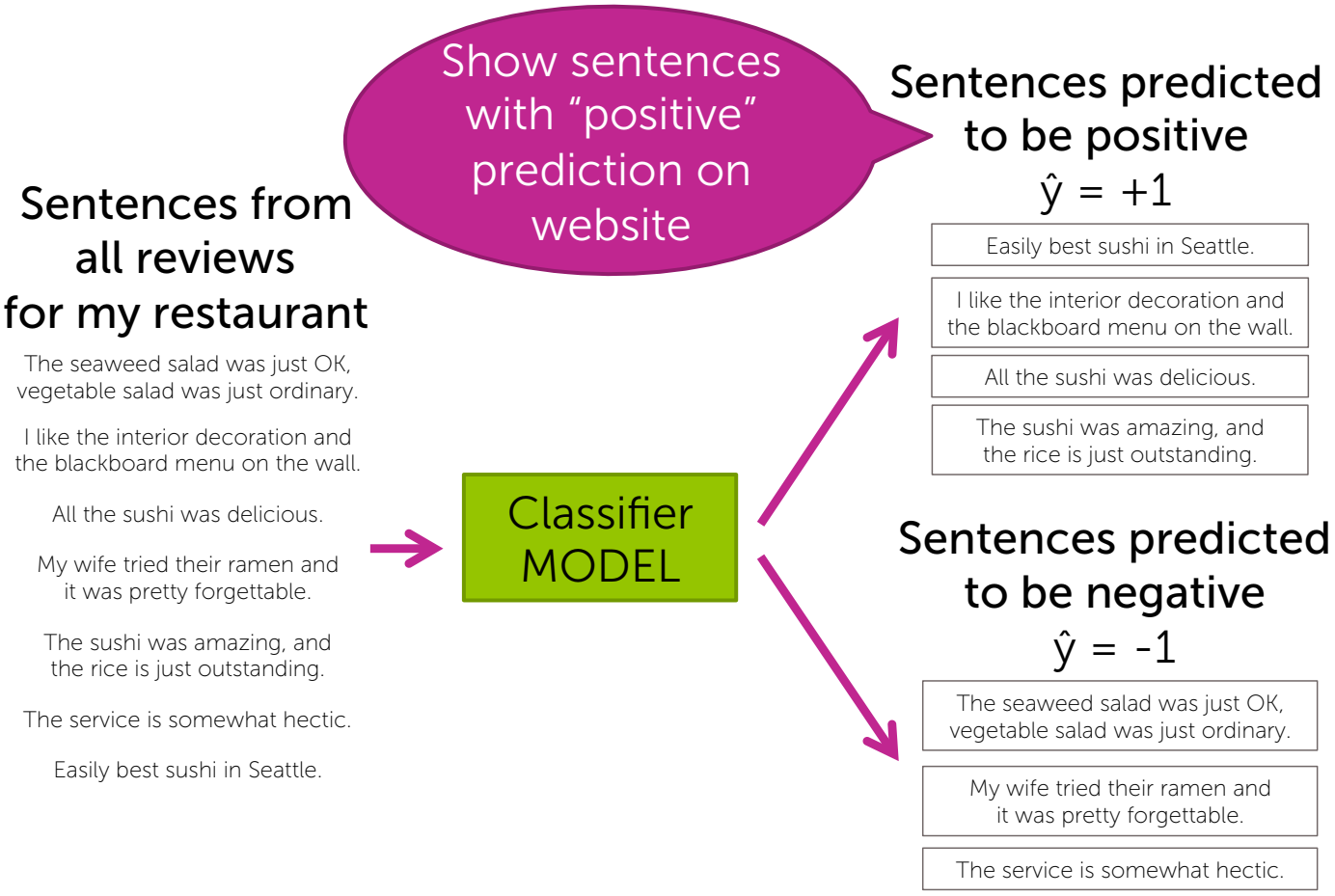
Sentiment classifier

Input $\mathbf{x}_i$: Easily best sushi in Seattle.



Output: $\hat{y}_i$
Predicted sentiment

Use the sentiment classifier model!



What does it mean for a classifier to be good?

Previously, we asked the question:

"What is good accuracy?"

We explored accuracy of random classifier as baseline

- For binary classification:
  - Half the time, you'll get it right! (on average)
    - ➔ classification error = 0.5
- For k classes, error= 1 - 1/k
  - error = 0.666 for 3 classes, 0.75 for 4 classes,…

At the very, very, very least,
you should healthily beat random…
Otherwise, it's (usually) pointless…

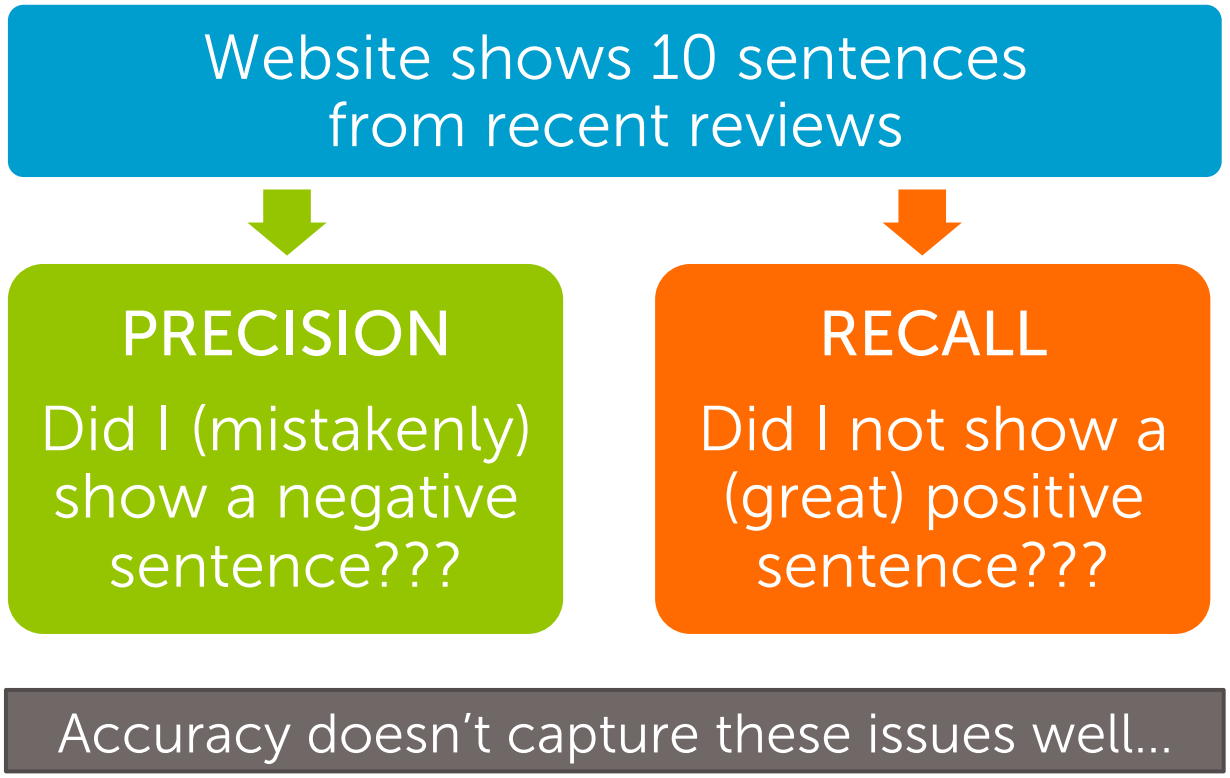
We explored the pitfalls of imbalanced problems: *Is 90% accuracy good? Depends …*

90% of sentences are negative!

90% accuracy by predicting every sentence is negative!!!

Amazing "performance" but not useful for me right now!

Automated marketing campaign cares about something else…

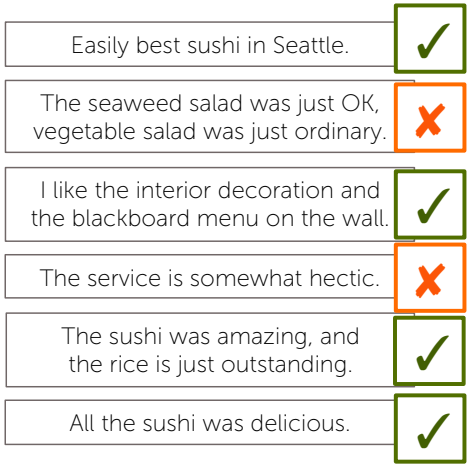


Accuracy doesn't capture these issues well…

Precision:
Fraction of positive predictions
that are actually positive

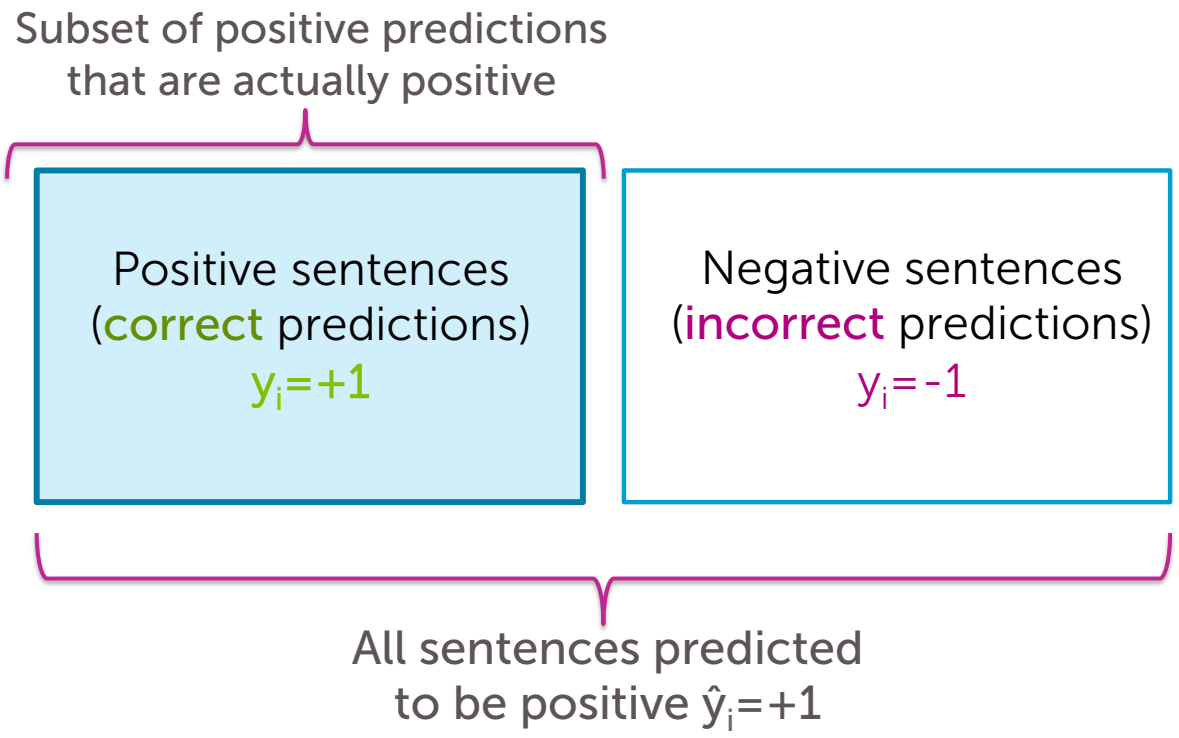
What fraction of the positive predictions are correct?

Sentences predicted to be positive: $\hat{y}_i = +1$

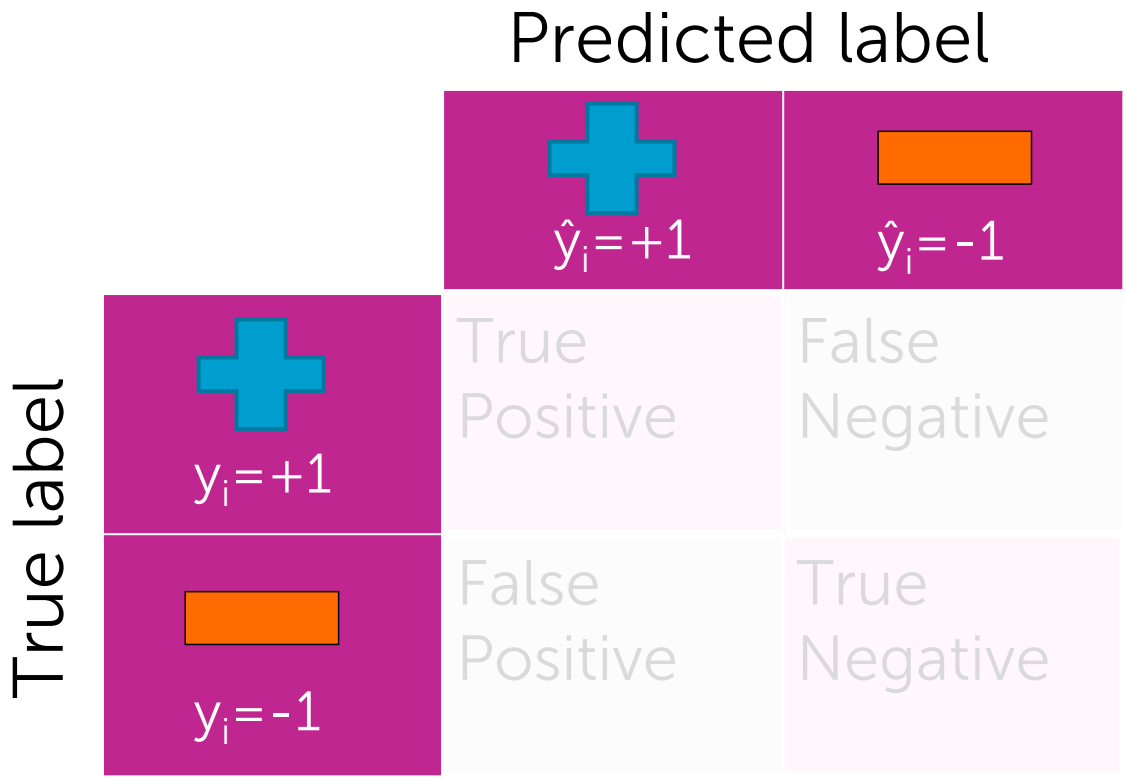


Only 4 out of 6 sentences predicted to be positive are actually positive

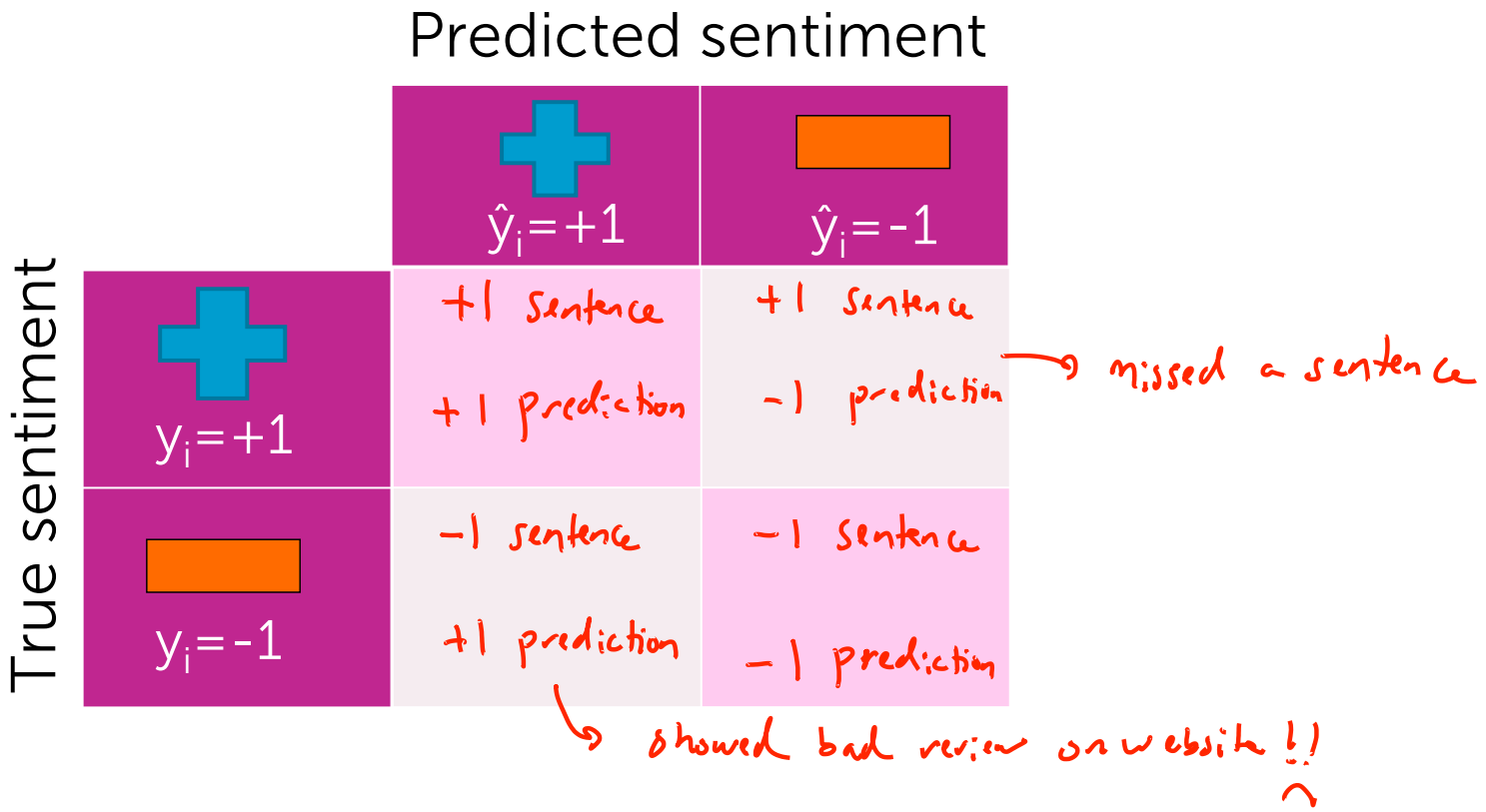
Precision: Fraction of positive predictions that are actually positive



Types of error: *Review*



Confusion matrix for sentiment analysis



Precision - Formula

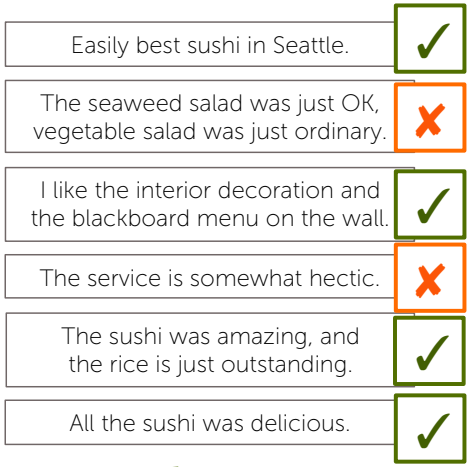
- Fraction of positive predictions that are correct

$$\text{precision} = \frac{\text{\# true positives}}{\text{\# true positives + \# false positives}}$$

  - Best possible value : 1.0
  - Worst possible value : 0.0

Example: Calculating precision

Sentences predicted to be positive: $\hat{y}_i = +1$



4 correct

2 mistakes

$$\text{precision} = \frac{4}{4+2} = \frac{2}{3}$$

Why precision is important

Shown on website

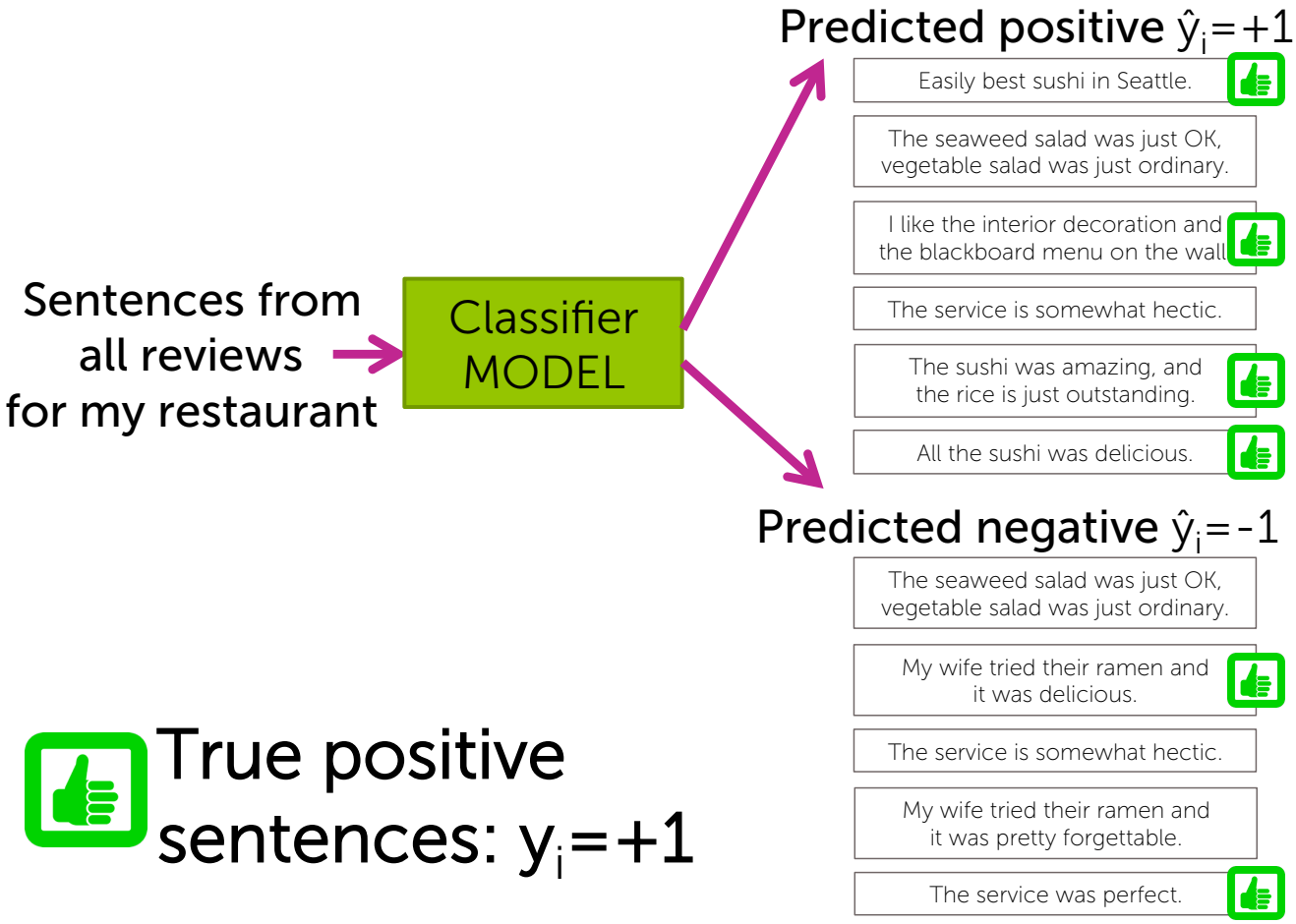
Sentences predicted to be positive: $\hat{y}_i = +1$

2 negative sentences shown to potential customers... ☹

High precision means positive predictions actually likely to be positive!

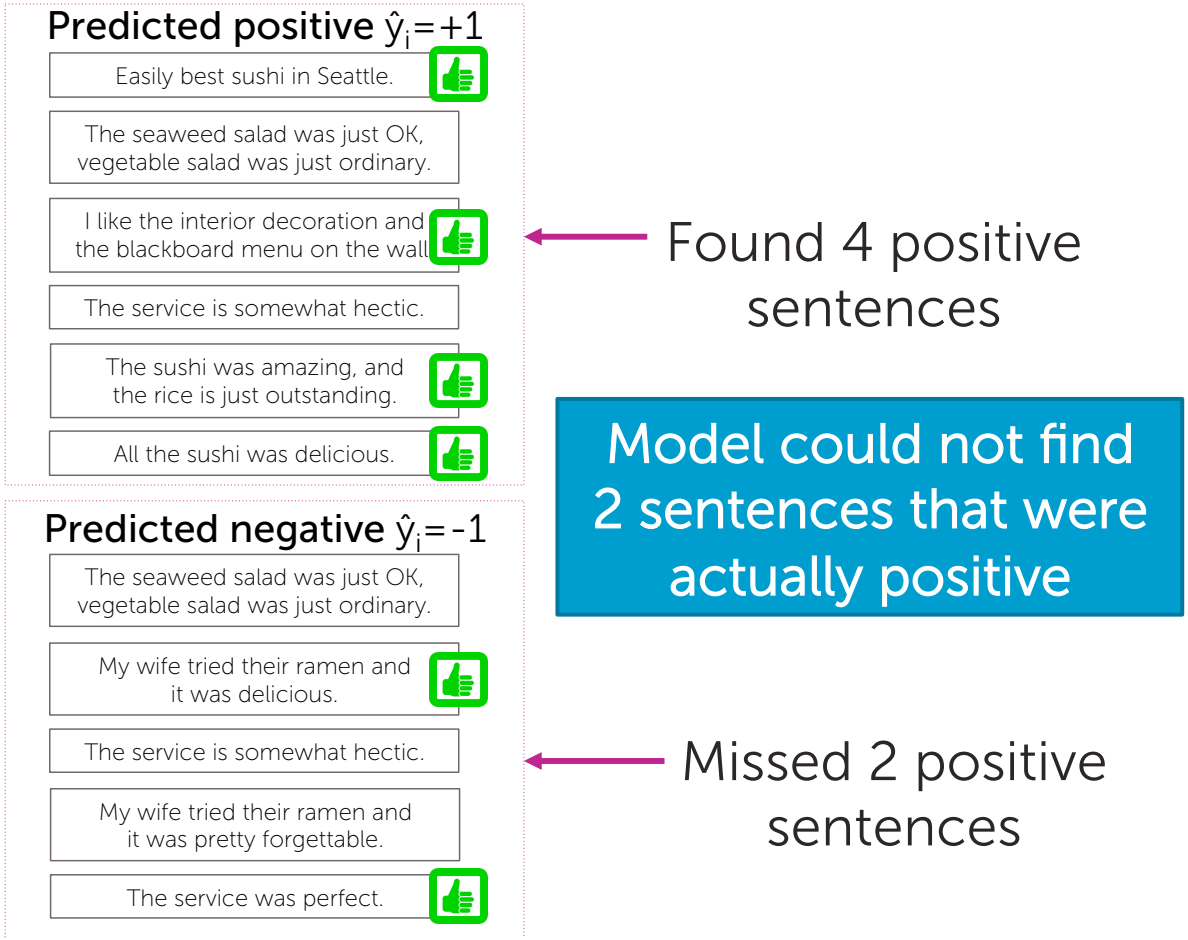
Recall:
Fraction of positive data predicted to be positive

Did I find all the positive sentences?

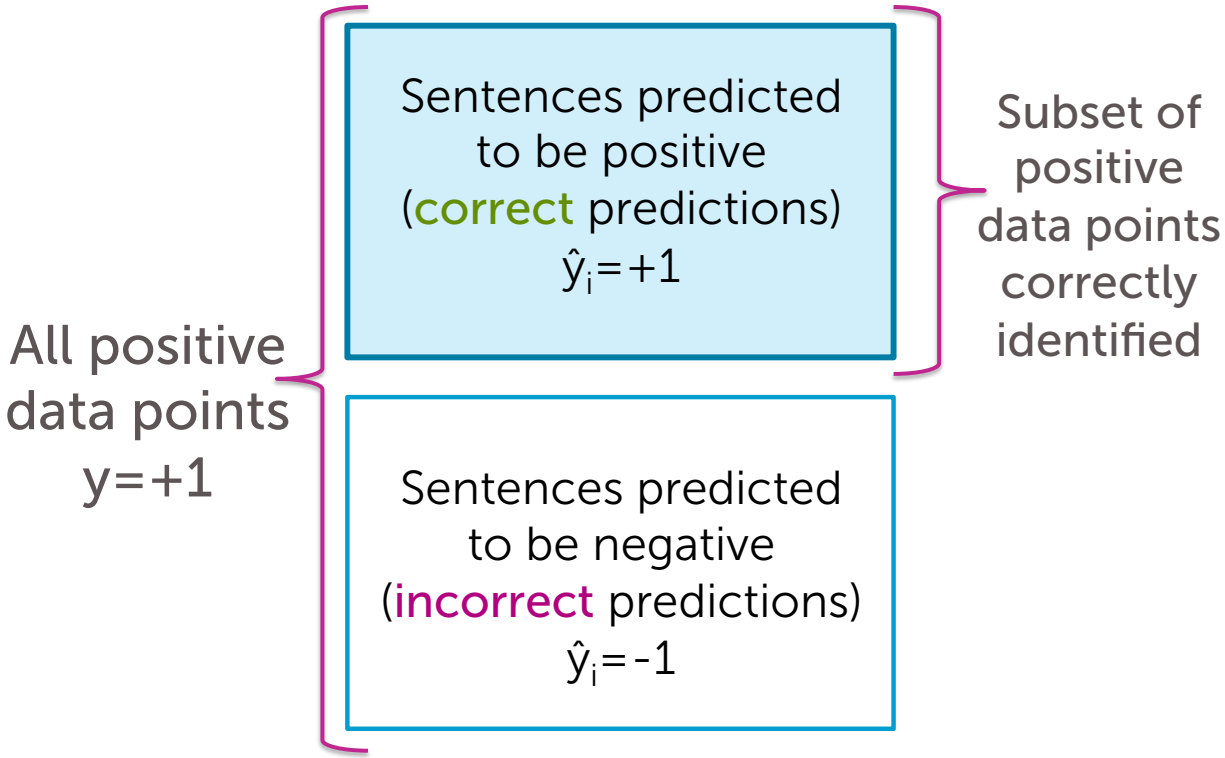


True positive sentences: $y_i=+1$

What fraction of positive sentences were missed out?



Recall: Fraction of positive data predicted to be positive



Recall - Formula

- Fraction of positive data points correctly classified

$$\text{Recall} = \frac{\text{\# true positives}}{\text{\# true positives + \# false negatives}}$$

  - Best possible value : 1.0
  - Worst possible value : 0.0

Why is recall important?

Predicted positive $\hat{y}_i = +1$

- Easily best sushi in Seattle.
- The seaweed salad was just OK, vegetable salad was just ordinary.
- I like the interior decoration and the blackboard menu on the wall
- The service is somewhat hectic.
- The sushi was amazing, and the rice is just outstanding.
- All the sushi was delicious.

Predicted negative $\hat{y}_i = -1$

- The seaweed salad was just OK, vegetable salad was just ordinary.
- My wife tried their ramen and it was delicious.
- The service is somewhat hectic.
- My wife tried their ramen and it was pretty forgettable.
- The service was perfect.

Want to show positive sentences on website

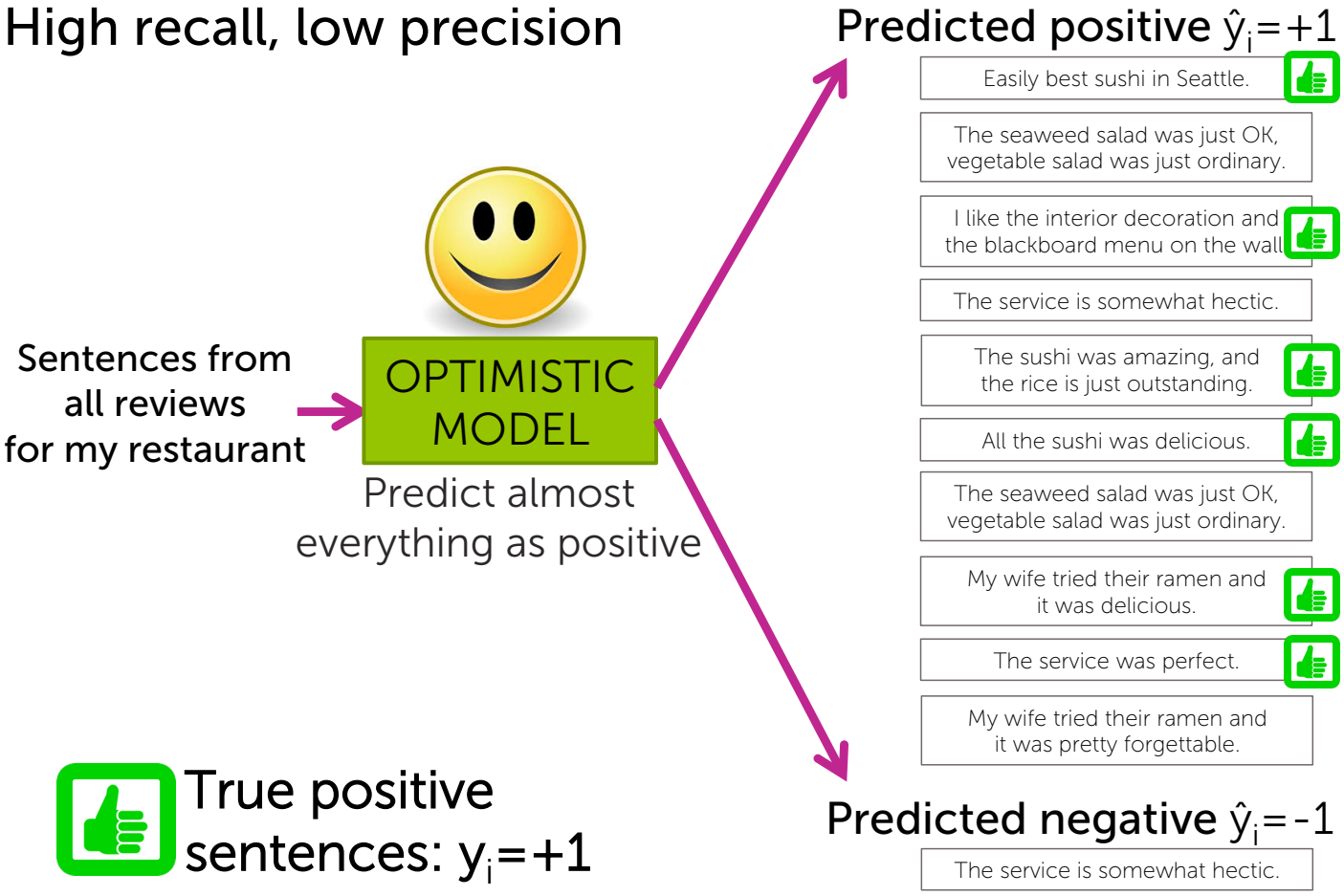
2 positive sentences not shown to potential customers... ☹

High recall means positive data points are very likely to be discovered!

Precision-recall extremes

Optimistic model:

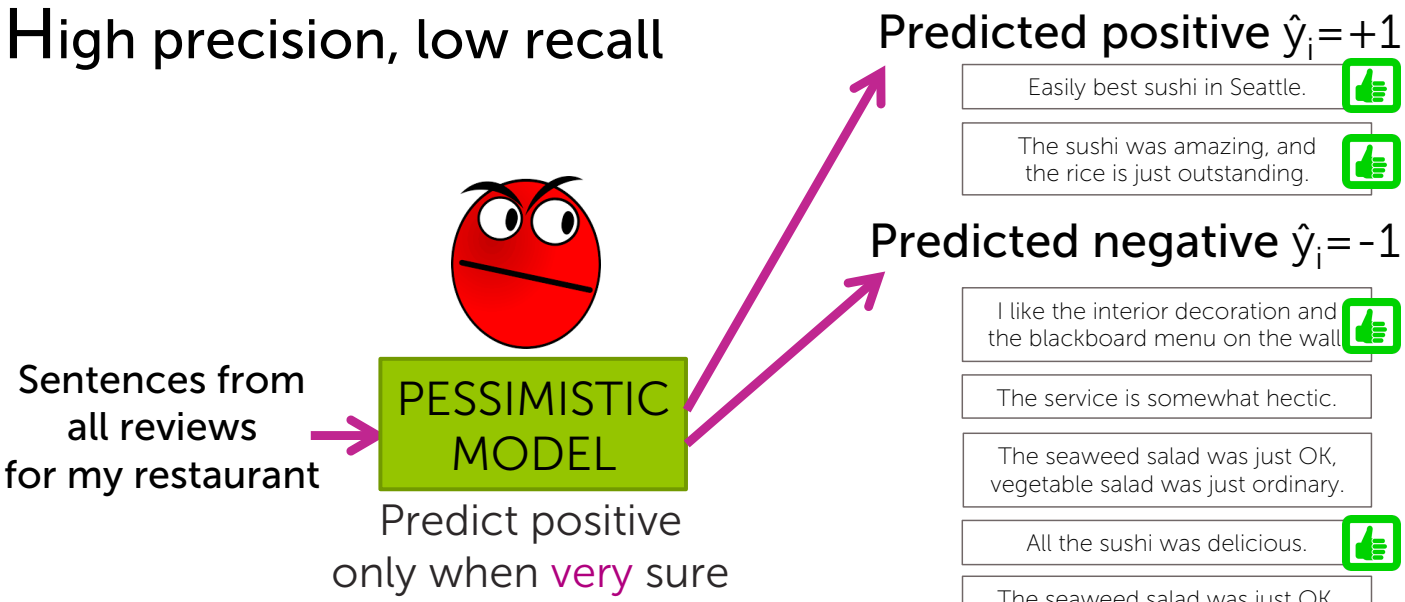
High recall, low precision



True positive sentences: $y_i=+1$

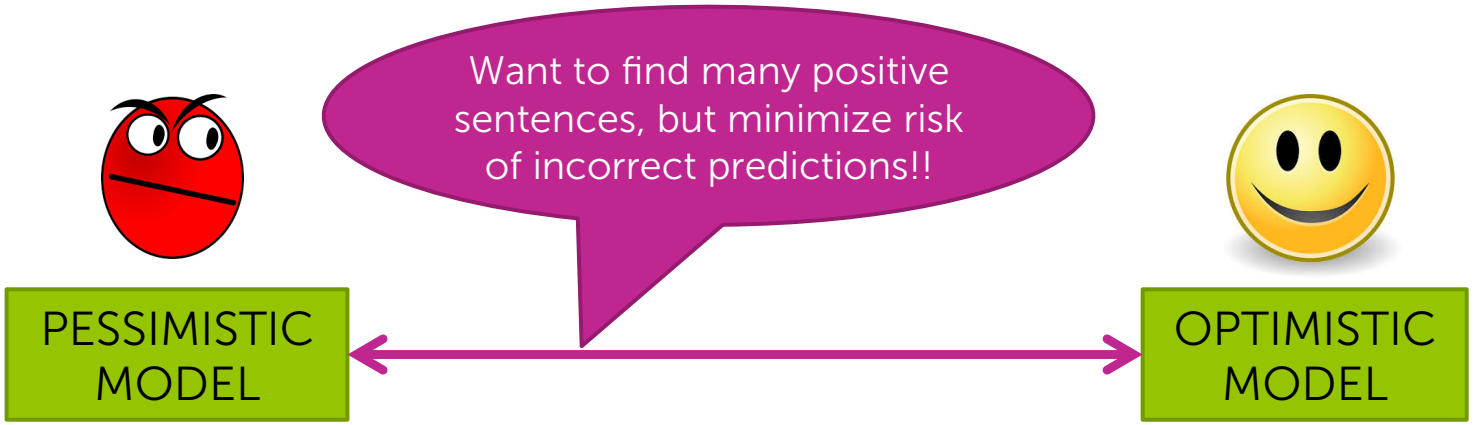
Pessimistic model:

High precision, low recall



True positive sentences: $y_i=+1$

Balancing precision & recall



Tradeoff precision and recall

Can we tradeoff precision & recall?

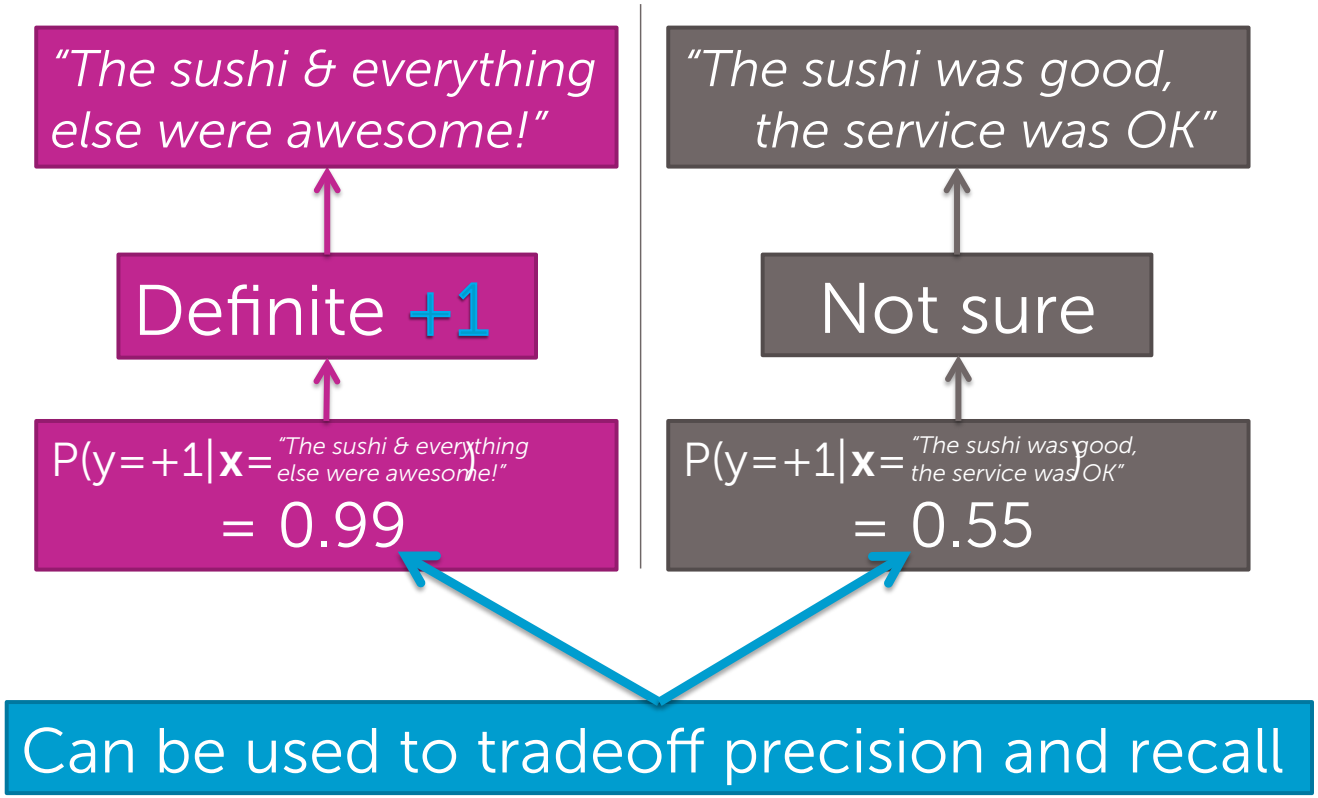
Low precision, high recall

High precision, low recall

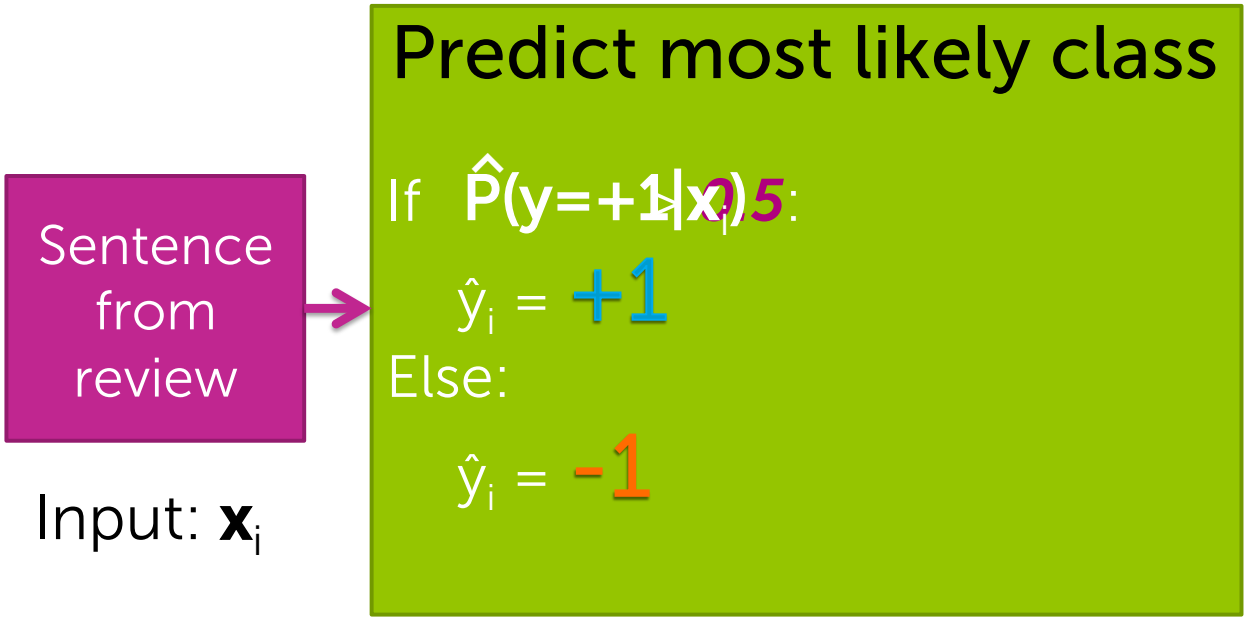
Optimistic Model
Predict almost everything as positive

Pessimistic Model
Predict positive only when very sure

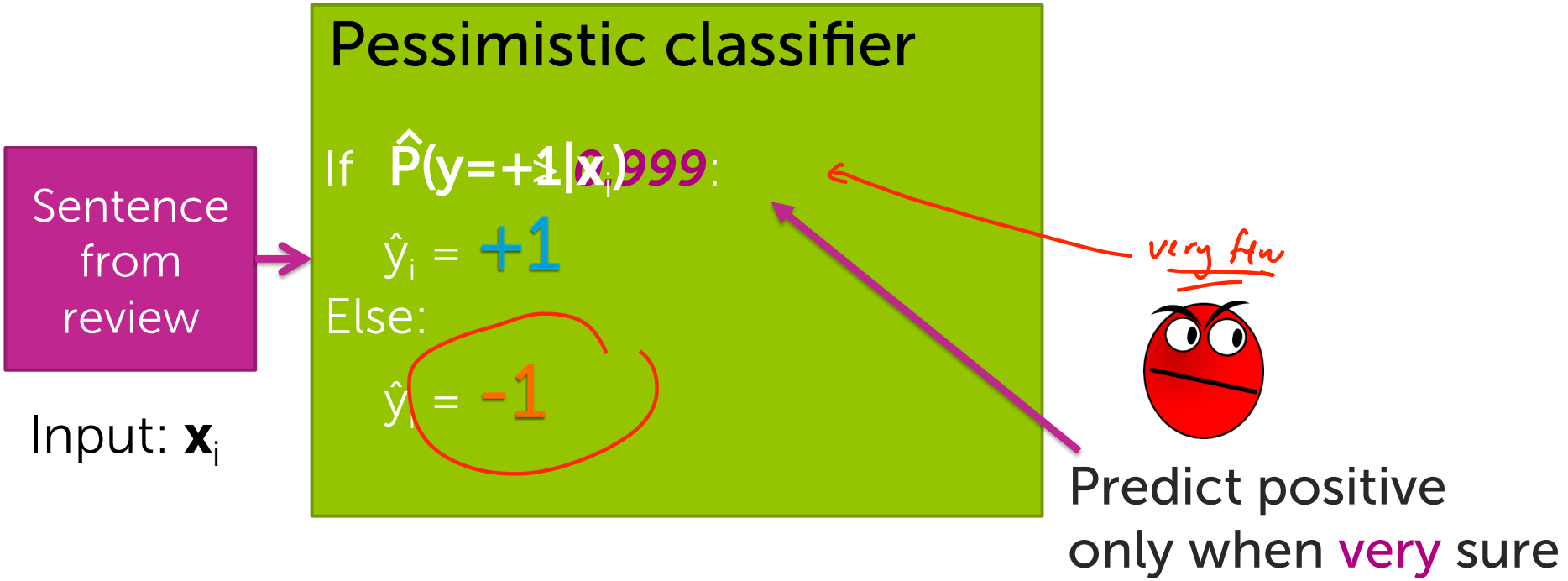
How confident is your prediction?



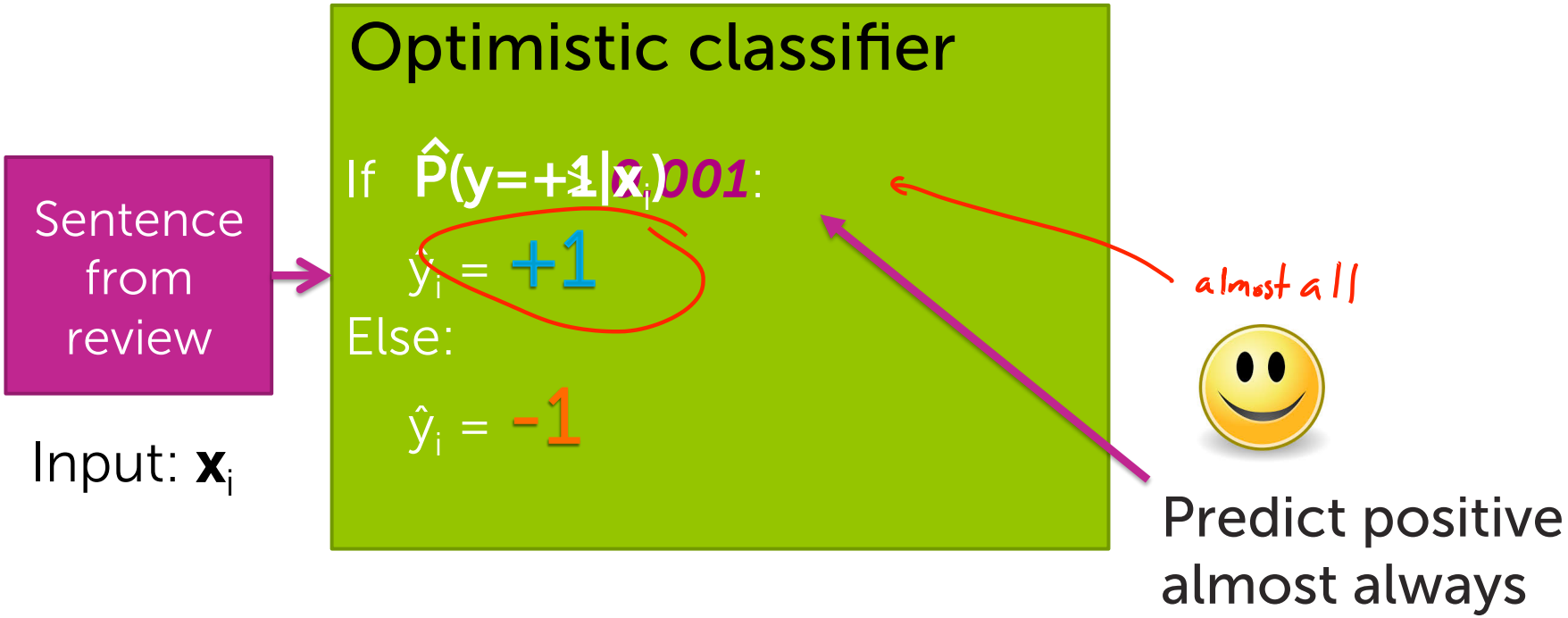
Basic classifier



Pessimistic: High precision, low recall

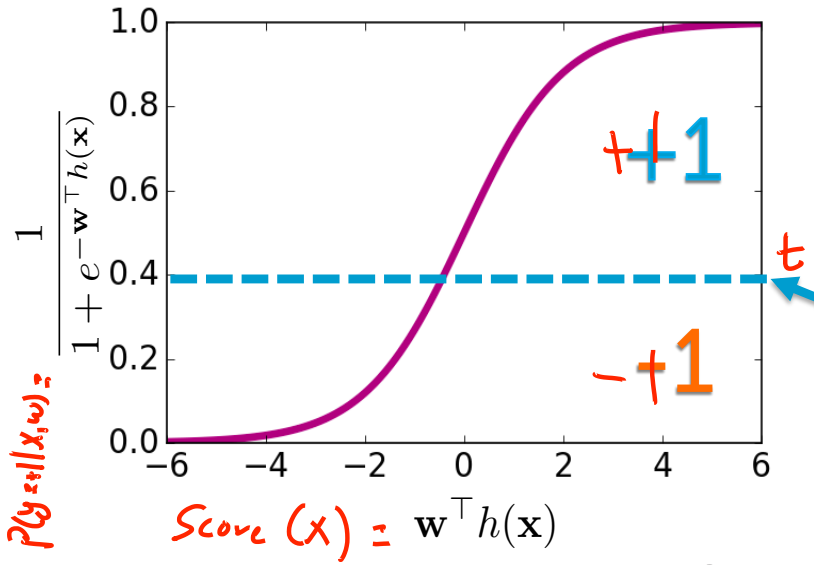


Optimistic: Low precision, high recall



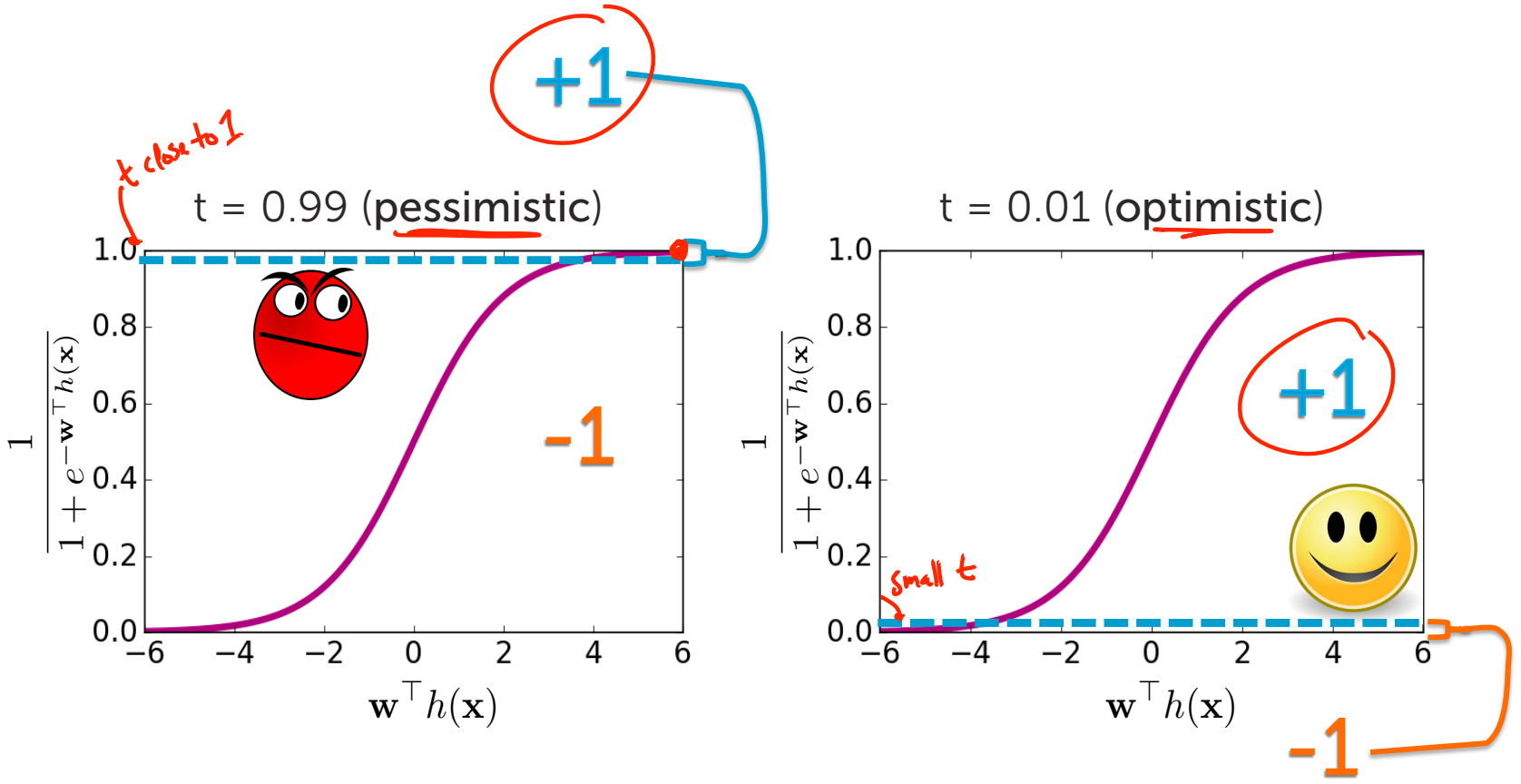
Prediction probability threshold

Probability t above which model predicts true

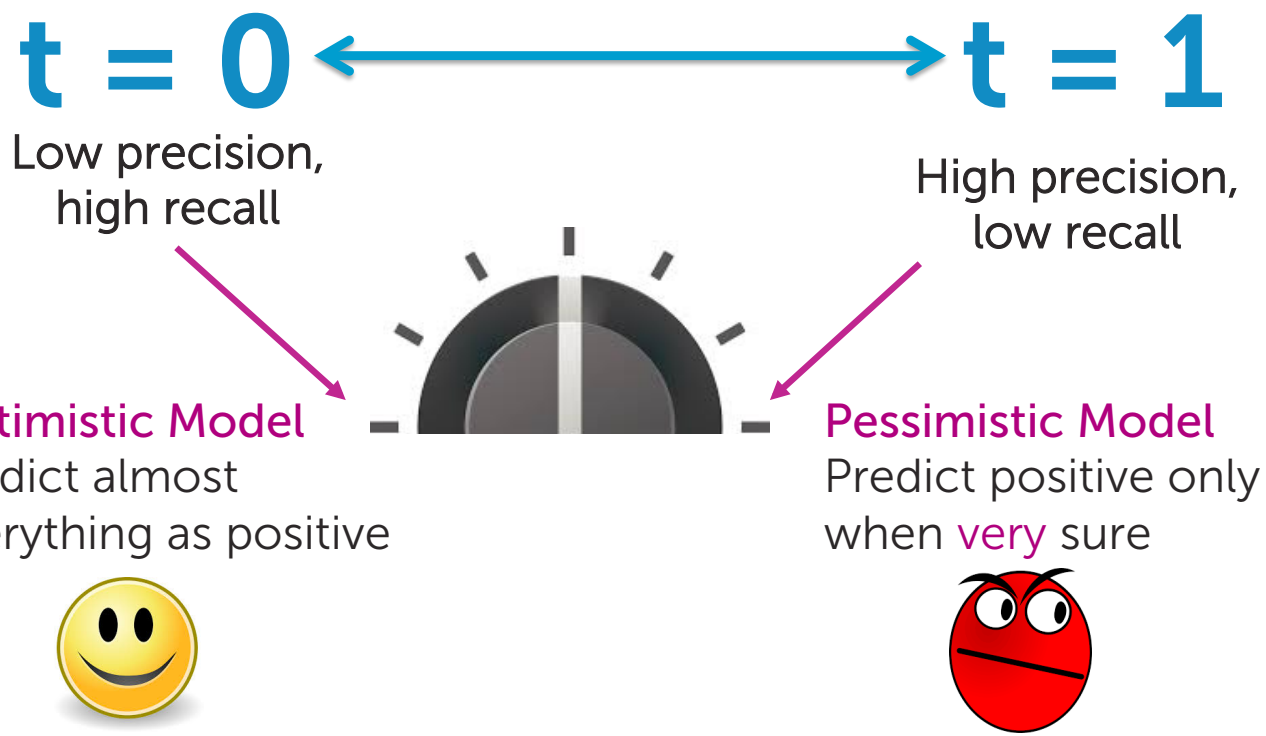


Set $\hat{y} = +1$ if $\hat{P}(y|x) >= t$

Example threshold values

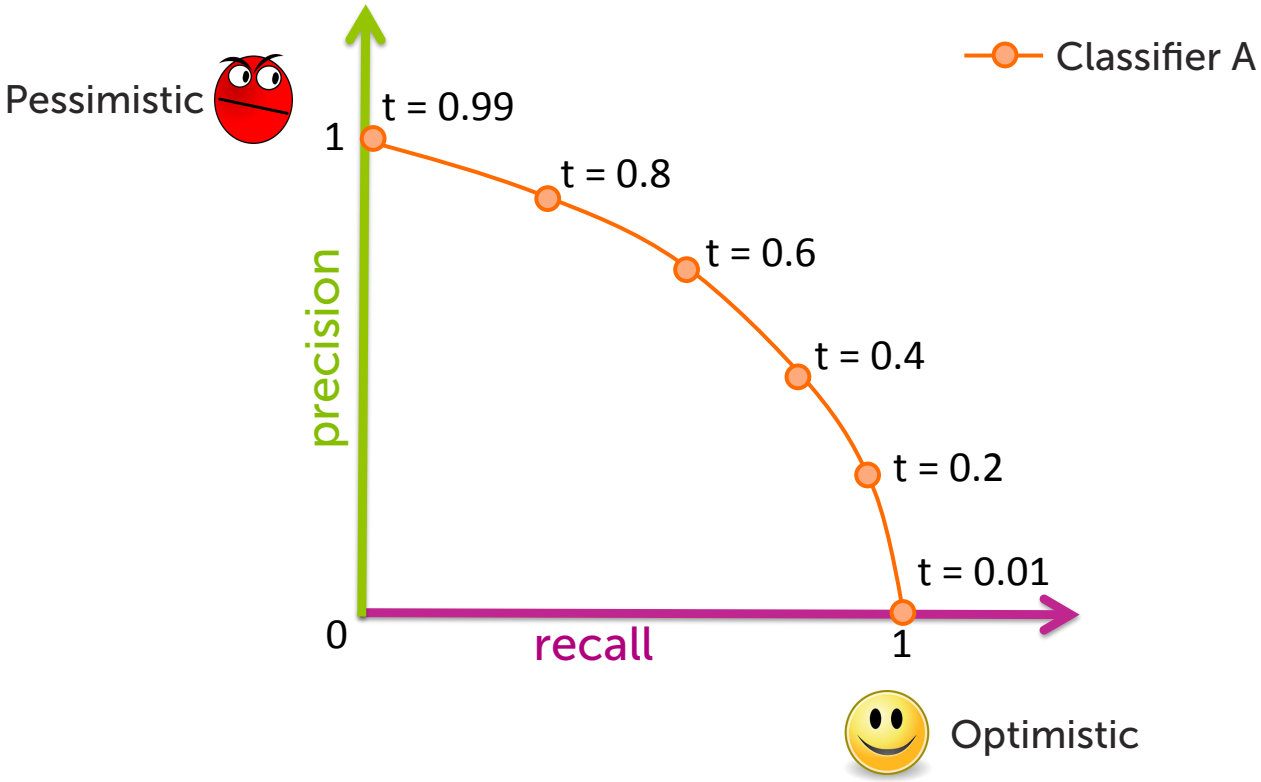


Tradeoff precision & recall with threshold

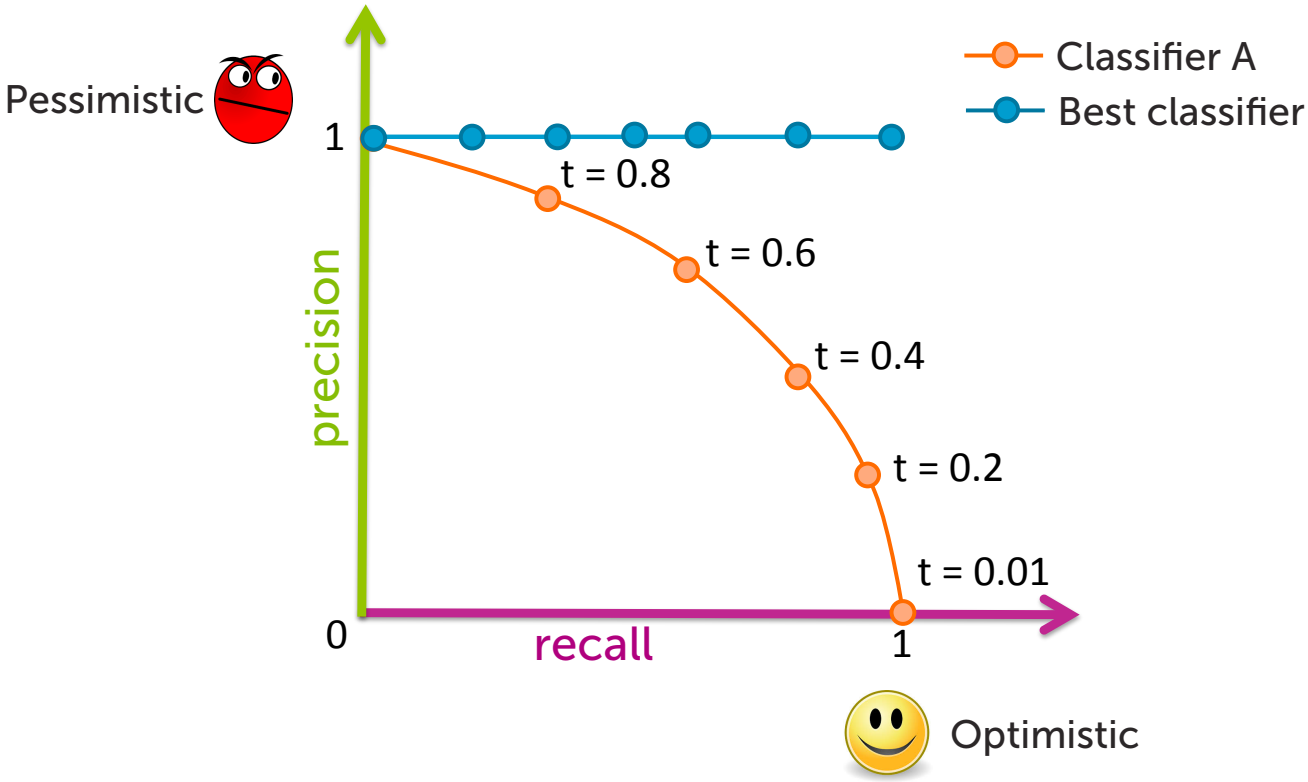


Precision-recall curve

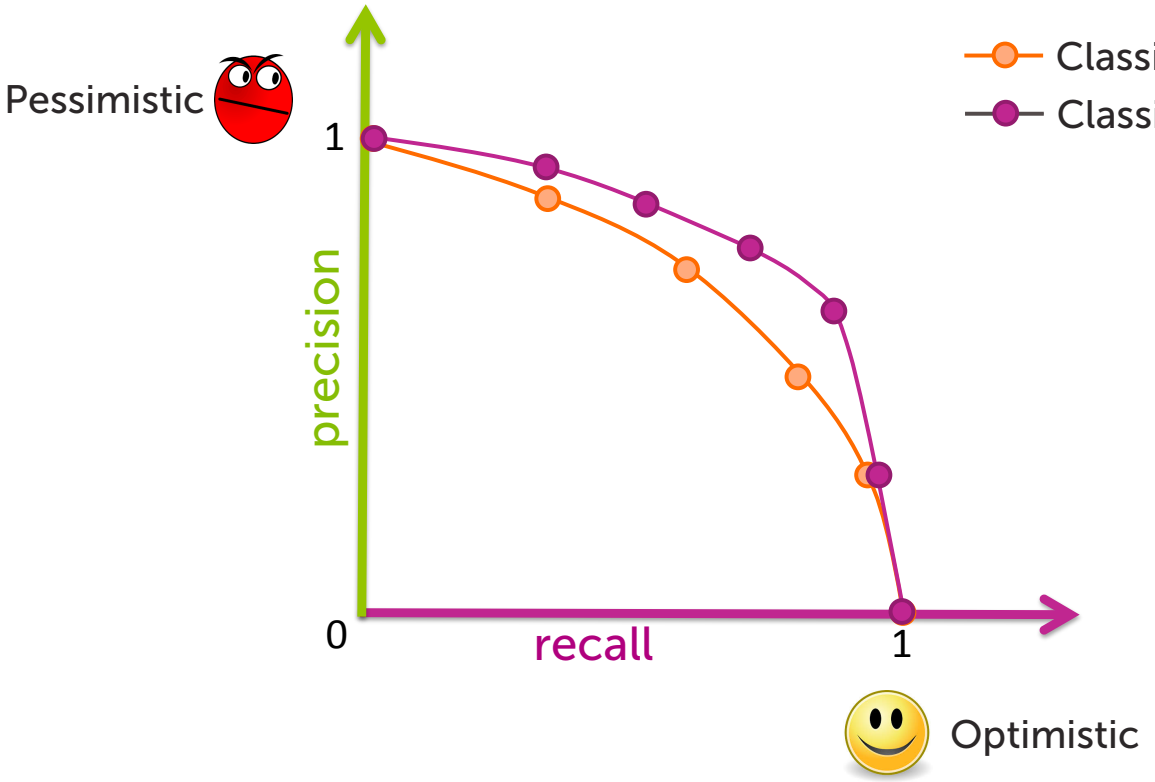
The precision-recall curve



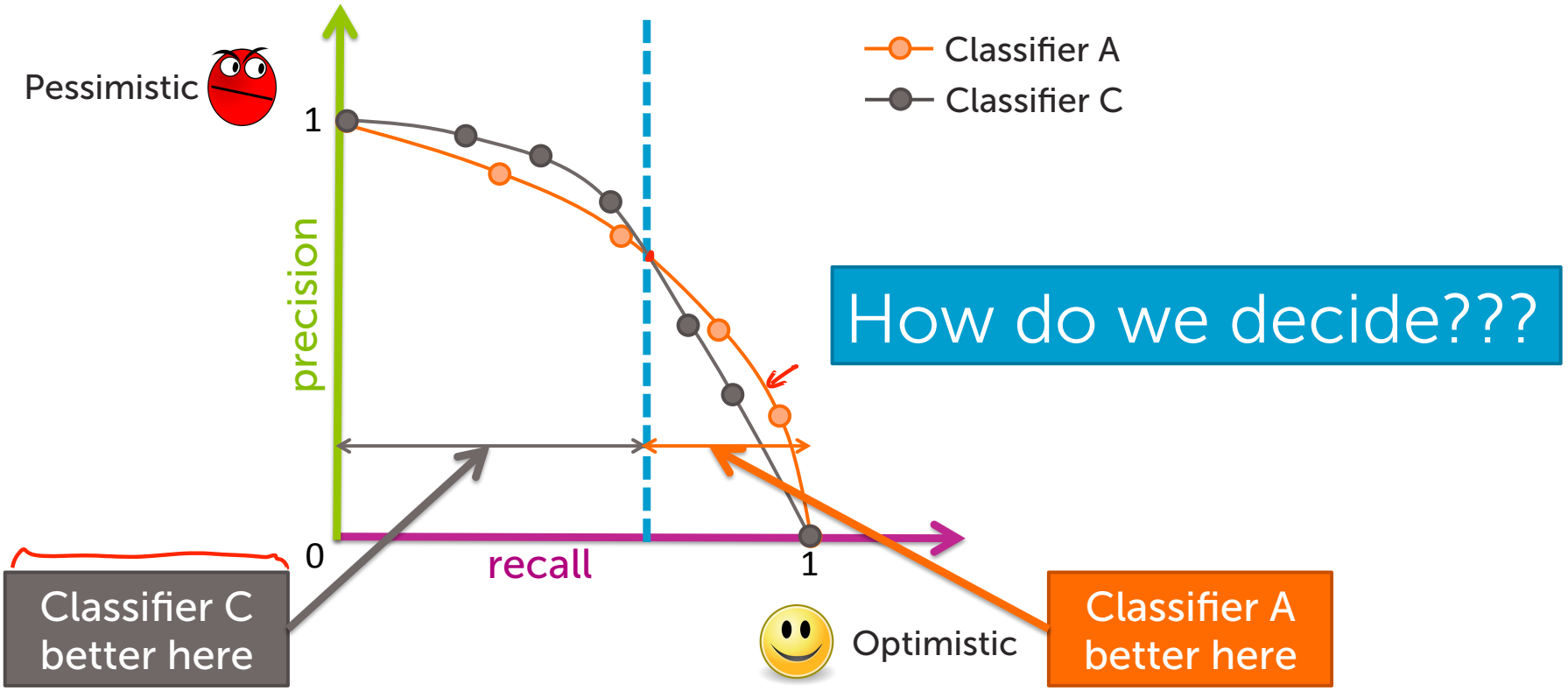
What does the perfect algorithm look like?



Which classifier is better? A or B?



Which classifier is better? A or C?



Compare algorithms

- Often, reduce precision-recall to single number to compare algorithms
  - F1 measure, area-under-the-curve (AUC),...

Precision at k