Assessing Performance

Measuring loss

Loss function $L(y, f_{\hat{w}}(x))$ — {Cost of using $\hat{w}$ at x when y is true

actual value

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

Examples: (assuming loss for underpredicting = overpredicting)

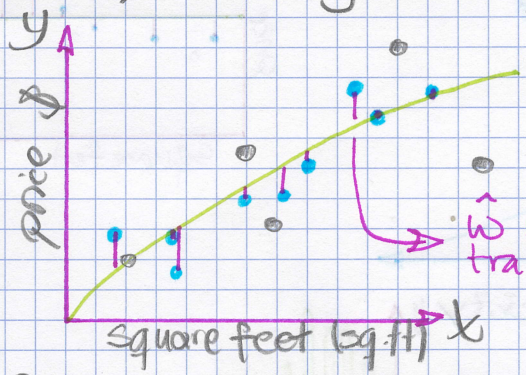
Absolute error: $L(y, f_{\hat{w}}(x)) = |y - f_{\hat{w}}(x)|$

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

Training error: assessing loss on the training set

Assessing the loss

Part 1: Training error



Define training data •

Example: Fit quadratic to minimize RSS

$\hat{w}$ minimizes RSS of training data

Compute training error

1. Define a loss function $L(y, f_{\hat{w}}(x))$
   - Eg squared error, absolute error, ...
2. Training error = avg loss on houses in training set

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

↳ fit using training data

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

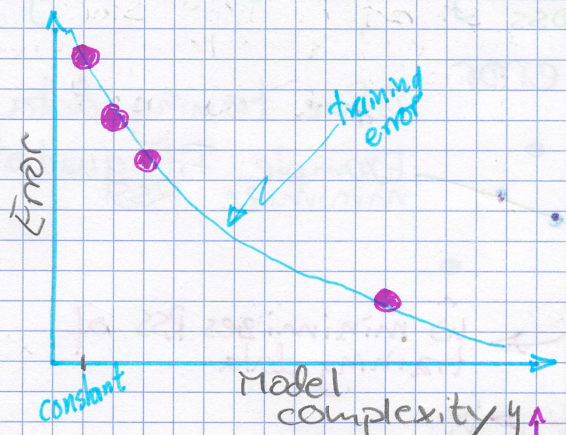
Training error $(\hat{w}) =$

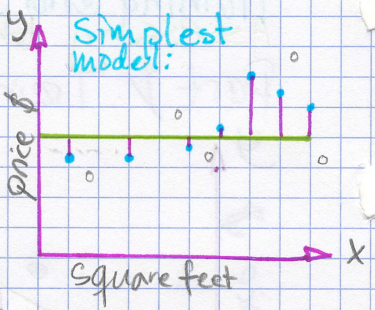
$$= \frac{1}{N} \cdot \left[(\$_{train1} - f_{\hat{w}}(sq.ft._{train1}))^2 + \right.$$
$$+ \$_{train2} - f_{\hat{w}}(sq.ft._{train2}))^2 +$$
$$\left. + \ldots + \text{include all tranning houses} \right]$$

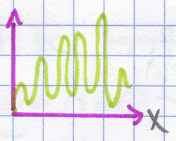
$$= \frac{1}{N} \sum_{i=1}^{N} (y_i - f_{\hat{w}}(x_i))^2$$

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

{ Root mean square error



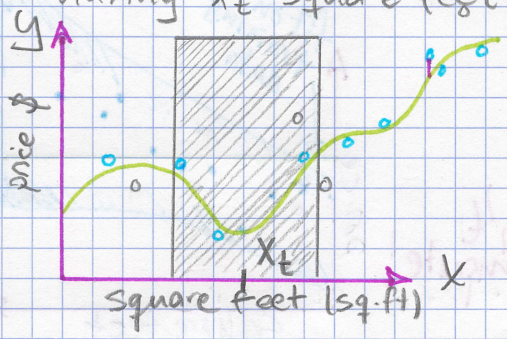




Is training error a good measure of predictive performance?

How do we expect to perform on a new house?

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



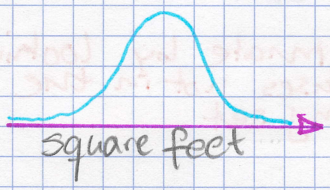
Issue: Training error is overly optimistic because $\hat{w}$ was fit to training data

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

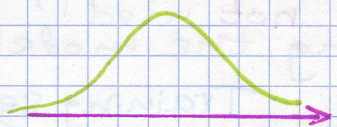
Assessing the loss

Part 2: Generalization (true) error

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?



For fixed #sq.ft

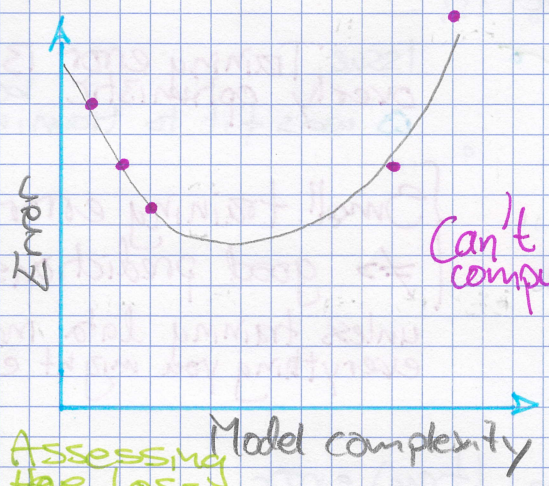
Generalization error definition
Really want estimate of loss over all possible ($\hat{w}$, $) pairs

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

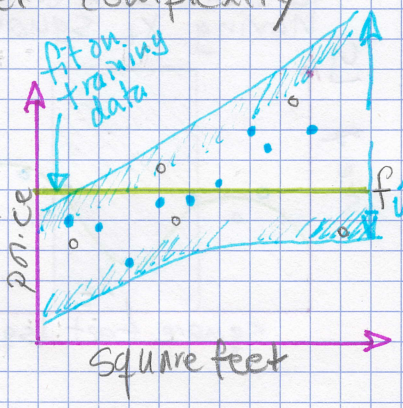
average over all possible (x,y) pairs weighted by how likely each is

fit using training data

Generalization error vs. model complexity



Can't compute!



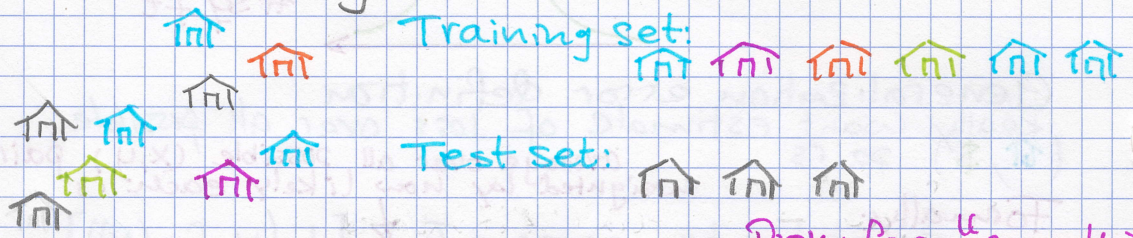
Assessing the loss

Part 3: Test error

Approximating generalization error
Wanted estimate of loss over all possible (🏠, $) pairs

Approximate by looking at houses not in the training set

Forming a test set
Hold out some (🏠, $) that are **not** used for fitting the model



Proxy for "everything you might see"

Compute test error

Test error
= avg loss on houses in test set

$$= \frac{1}{N_{test}} \sum_{\substack{j \text{ in} \\ \text{test set}}} L(y_j, f_{\hat{w}}(x_j))$$

test points

fit using training data

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

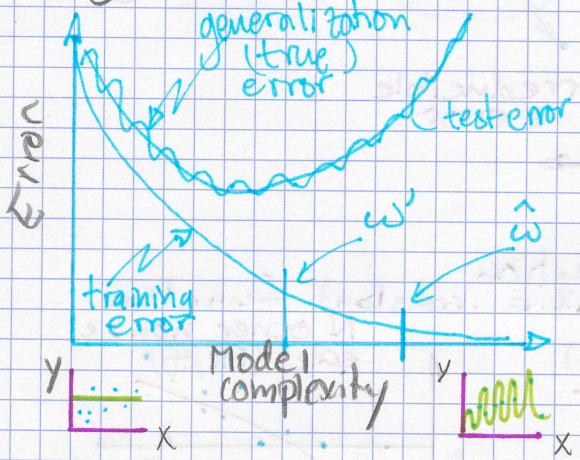
[($\$_{test1}$ - $f_{\hat{w}}$(sq.ft.$_{test1}$))^2 +

+ ($\$_{test2}$ - $f_{\hat{w}}$(sq.ft.$_{test2}$))^2 +

+ ($\$_{test3}$ - $f_{\hat{w}}$(sq.ft.$_{test3}$))^2 +

+ ... include all test houses]

Training, true, & test error vs. model complexity

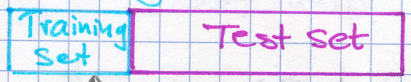


Overfitting if:

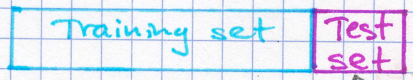
if there exists a model with estimated params w' such that

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

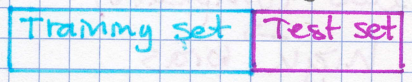
Training/test split



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

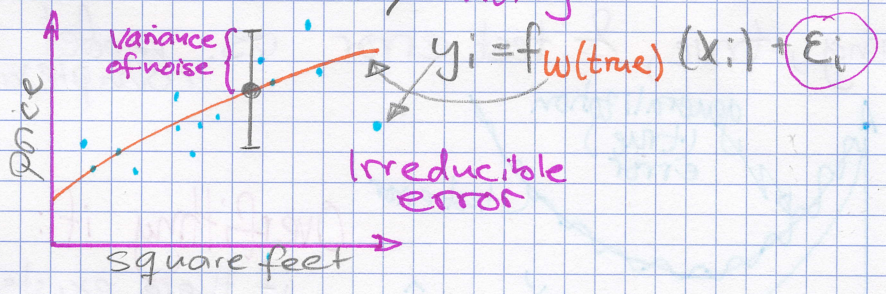
Irreducible error and bias

3 sources of error

In forming predictions, there are 3 sources of error

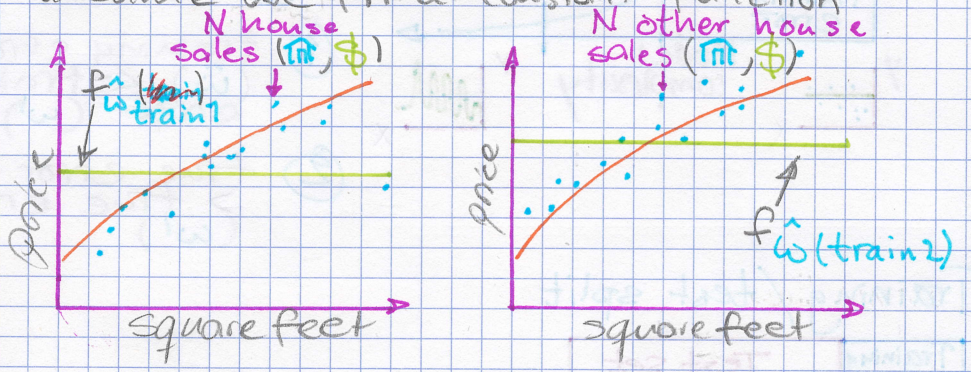
1. Noise
2. Bias
3. Variance

Data inherently noisy

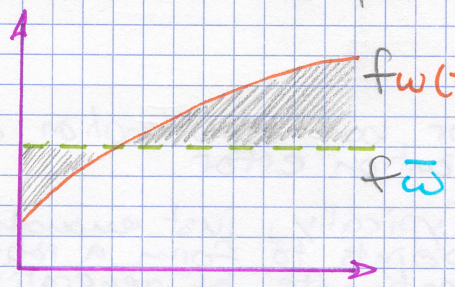


Bias contribution

assume we fit a constant function



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

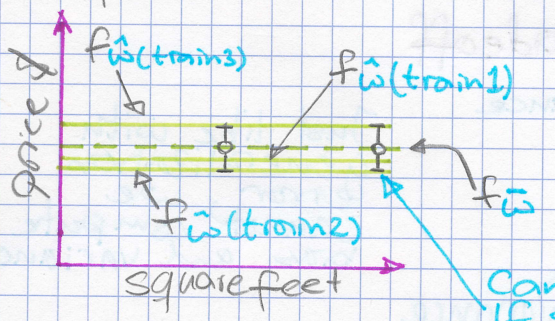


{ low complexity → high bias

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

Variance contribution

How much do specific fits vary from the expected fit

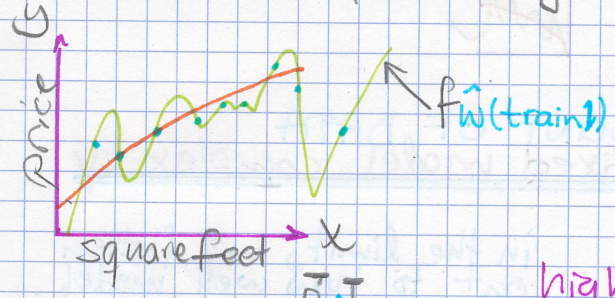


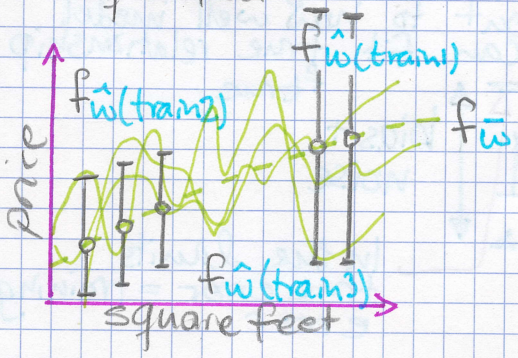
low complexity → low variance

Can specific fits vary widely? If so, erratic predictions

Variance of high-complexity models

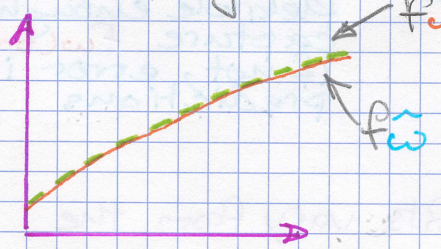
assume we fit a high order polynomial





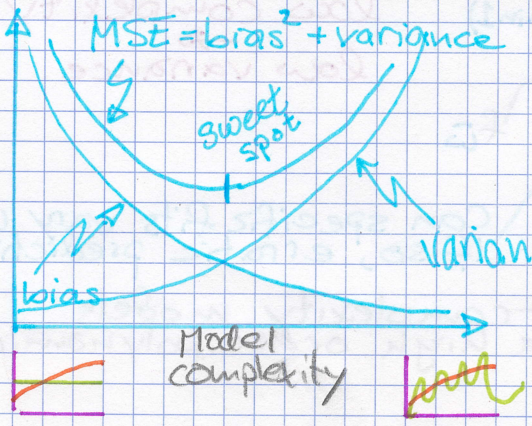
high complexity → high variance

Bias of high-complexity models



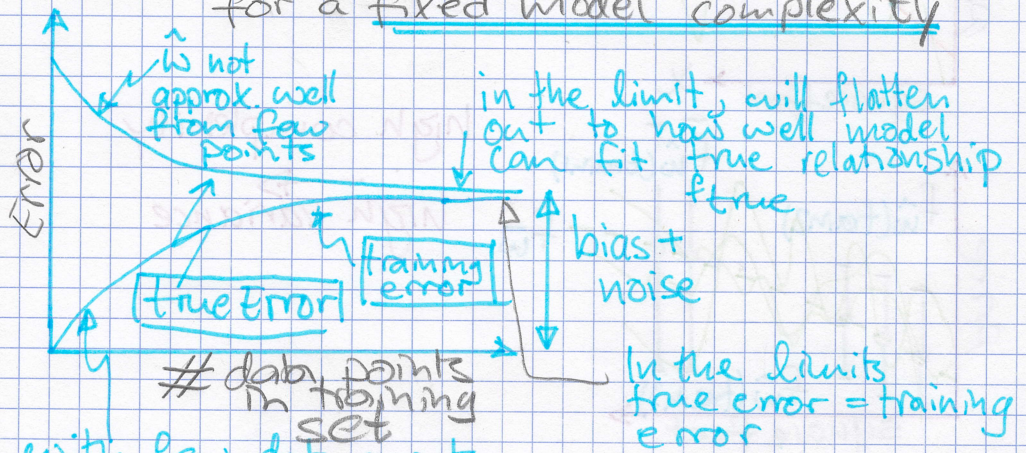
{ high complexity → low bias

Bias - variance tradeoff



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

ERROR VS. AMOUNT OF DATA for a fixed model complexity

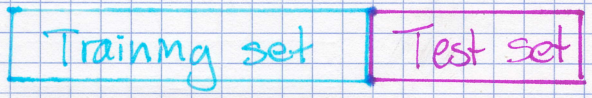


with few data points, fixed complexity model can fit these points reasonably well

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



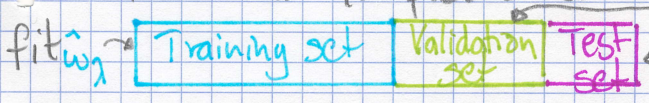
1. Model selection
   For each considered model complexity λ:
   i. Estimate parameters $\hat{w}_\lambda$ on training data
   ii. Assess performance of $\hat{w}_\lambda$ on test data
   iii. Choose λ^* to be λ with lowest test error
2. Model assessment
   Compute test error of $\hat{w}_{\lambda^*}$ (fitted model for selected complexity λ^*) to approx generalization error

Overly optimistic!

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

- λ^* was selected to minimize test error (ie., λ^* 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 λ^* such that $\hat{w}_{\lambda^*}$ minimizes error on validation set

2. Approximate generalization error of $\hat{w}_{\lambda^*}$ using test set

Typical splits

	Training set	Validation set	Test set
1)	80%	10%	10%
2)	50%	25%	25%