Today

- Orthogonal polynomials
- Continue Cross-Validation (CV)
  - inference algorithm
  - algorithm from scratch
  - pseudocode to R code

Basic full ML setup

- 3 algorithms:
  - model: flexible function $\hat{f}(x)$;
    e.g. polynomial linear model

    $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \ldots + \beta_m x^m$ m=order
  - training: optimize objective function
    e.g. least squares
  - minimize $SSQ = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ for training data
  - inference: measure error by cross validation; tuning parameter (order of poly)

Prediction

$$\boldsymbol{x} \longrightarrow \boxed{\hat{\boldsymbol{f}}(\boldsymbol{x})} \longrightarrow \boldsymbol{y}$$

Goal: find function $\hat{f}$ that has good predictive performance

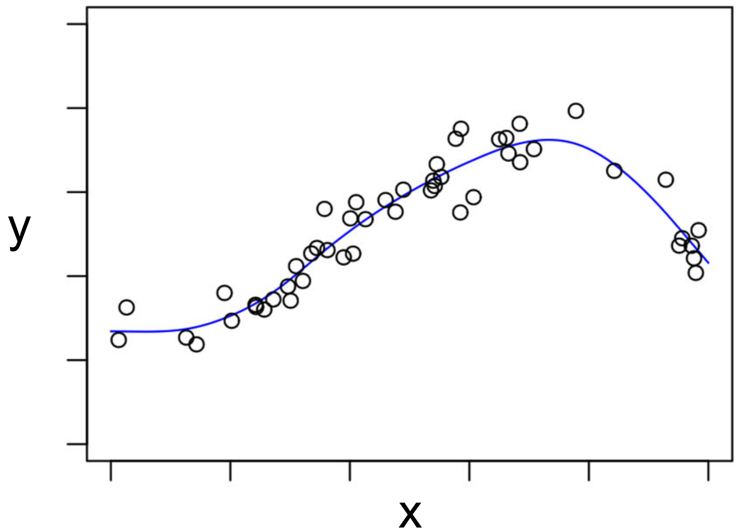
Accurate on new observations of y

(out-of-sample accuracy)

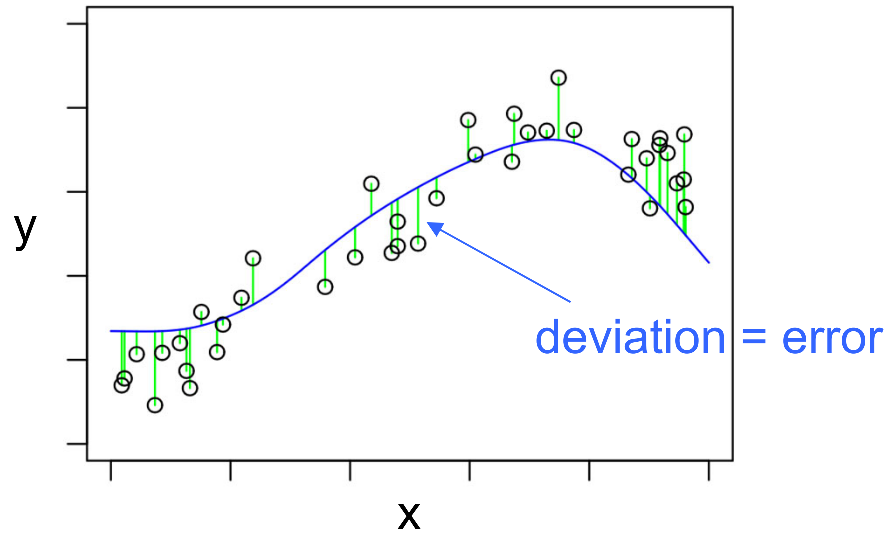
Inference algorithm

Basic algorithm: out-of-sample validation

1. Train model on training dataset



2. Test model on validation dataset



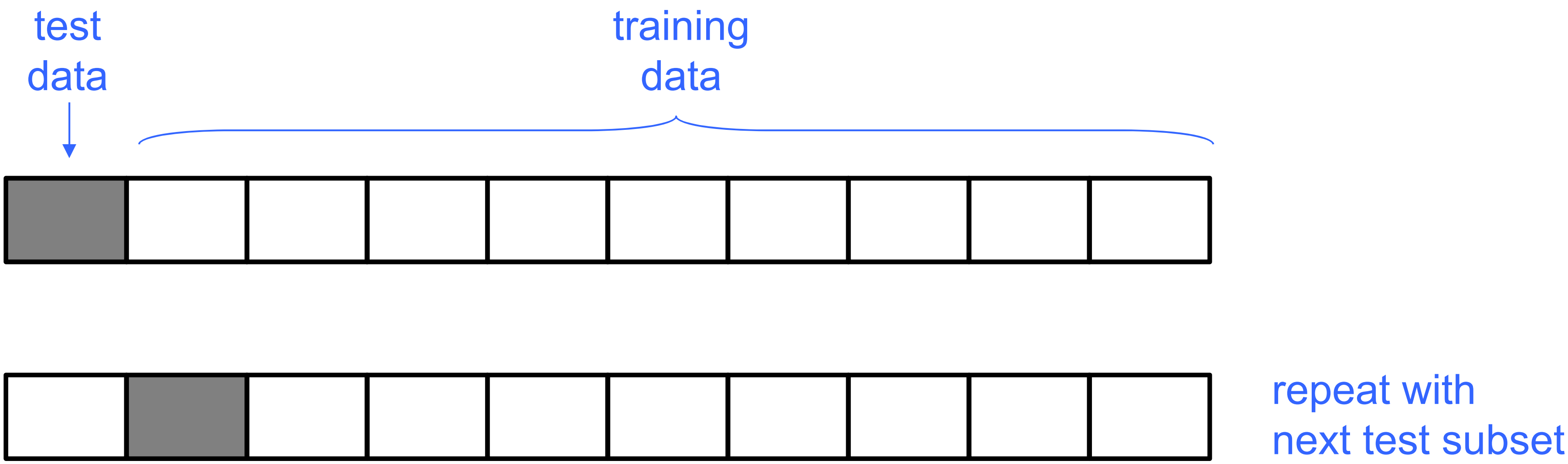
e.g. mean square error (MSE)

Cross validation (CV)

- Some approaches:
- Different datasets for train and test
- Holdout portion of a dataset (e.g. 10%)
  - aka train-test split
  - often used for huge datasets
- Both the above can suffer from bias because we have only one test set
- k-fold CV: replicate test sets

k-fold cross validation (CV)

Divide dataset into k parts (preferably randomly)



... repeat with each test subset

k-fold CV inference algorithm

Algorithm

```
divide dataset into k parts i = 1...k
for each i
        test dataset = part i
        training dataset = remaining data
        find f using training dataset
        use f to predict for test dataset
        e_i = prediction error
CV_error = mean(e)
```

Typical values for k: 5, 10, n

Tuning parameters

- Order of polynomial
- Different values of tuning parameters give different models
- Use CV inference algorithm to choose model with best predictive performance

Code

- 02_2_ants_cv_polynomial.R