Reminders

- Take charge of your GitHub repo
- `git clone ...`

Machine Learning

- Supervised learning
  - labeled response data
  - compare prediction to labeled (“known”) response
  - this semester
- Unsupervised learning
  - unlabeled response data, discover patterns
  - aka traditional topic: “multivariate analysis”
  - clustering, ordination etc

Machine Learning algorithms

- Model algorithm
- Training algorithm
- Inference algorithm

Today

- Machine learning with ants data
  - polynomial model algorithm
  - least squares training algorithm
- Explore Cross-Validation (CV)
  - inference algorithm
  - pseudocode to R code

Machine learning workflow

Overall algorithm:

1. Create model algorithm(s) for $\hat{f}(x)$
2. Use a training algorithm to find parameter values of $\hat{f}(x)$
3. Use an inference algorithm to measure prediction error and compare predictive skill among models (model families, tuning parameters, x sets, etc).

Basic full ML setup

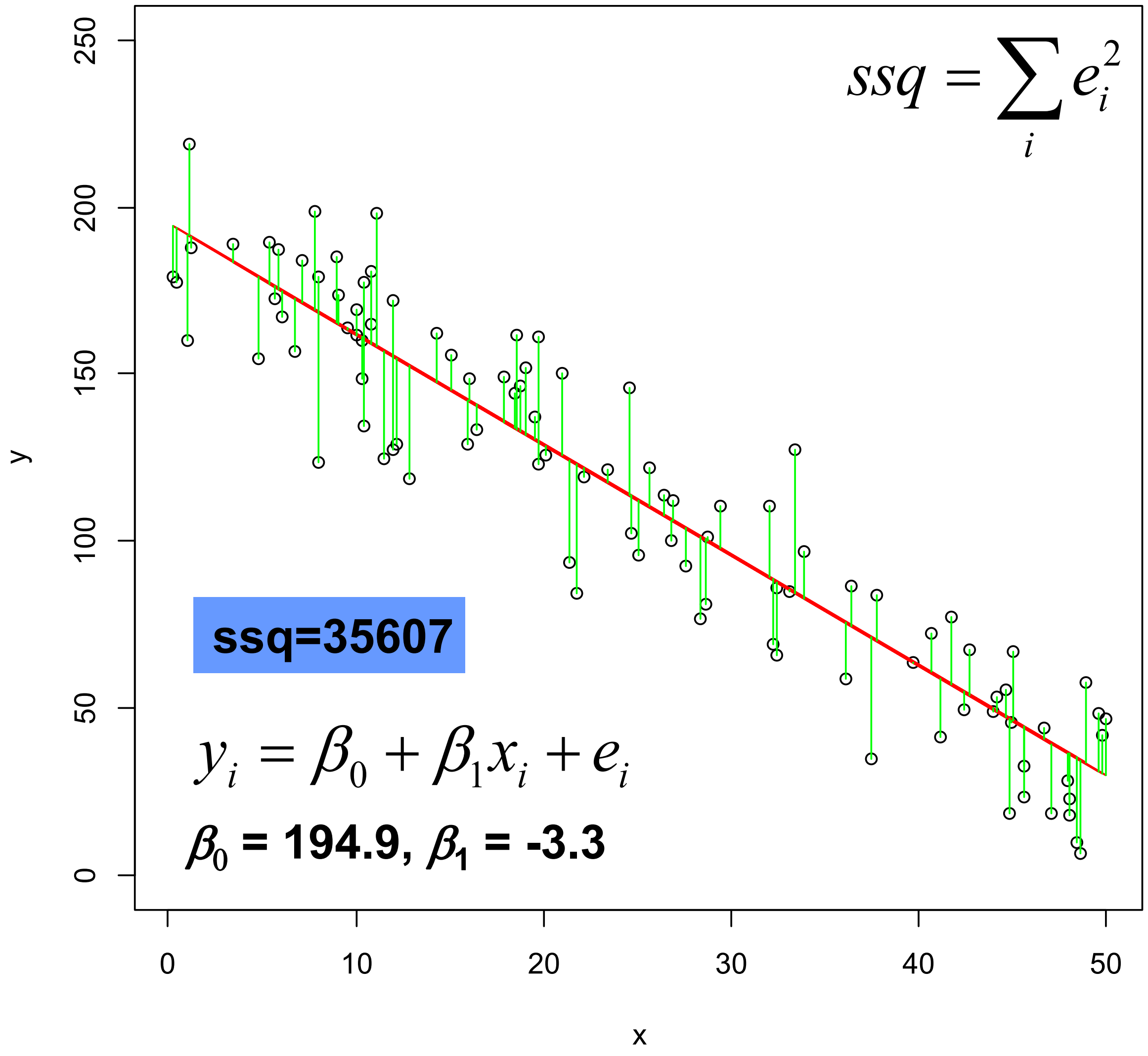
- 3 algorithms:
  - model: flexible function $\hat{f}(x)$;
    e.g. polynomial linear model

    $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \ldots + \beta_m x^m$ 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)

Code

- 02_2_ants_cv_polynomial.R
- 02_2_ants_cv_polynomial.py

Least squares optimization



General algorithmic idea:

Vary model parameters until we find the parameter values that minimize the distance of the model from the data

Optimization algorithms

Strategies

1. Systematically try all combinations of parameters - Grid search algorithms
2. Narrowing in: keep changing parameters in the direction that leads to lower SSQ - Descent algorithms
3. Try random values for parameter combinations - Monte Carlo algorithms
4. Solve for parameters using math - Analytical or numerical algorithms

Linear regression in R uses strategy 4

lm(y ~ x) solves a system of linear equations using linear algebra

Mathematical theory shows what to do (QR decomposition)

Numerical algorithm is needed to do it (householder algorithm)

Fast, specialized, guaranteed to find the minimum SSQ. Only works for SSQ: **limited to ordinary linear regression**.

Prediction

$$\begin{tikzcd} \boldsymbol{x} \arrow[r] & \boxed{\hat{\boldsymbol{f}}(\boldsymbol{x})} \arrow[r] & \boldsymbol{y} \end{tikzcd}$$

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

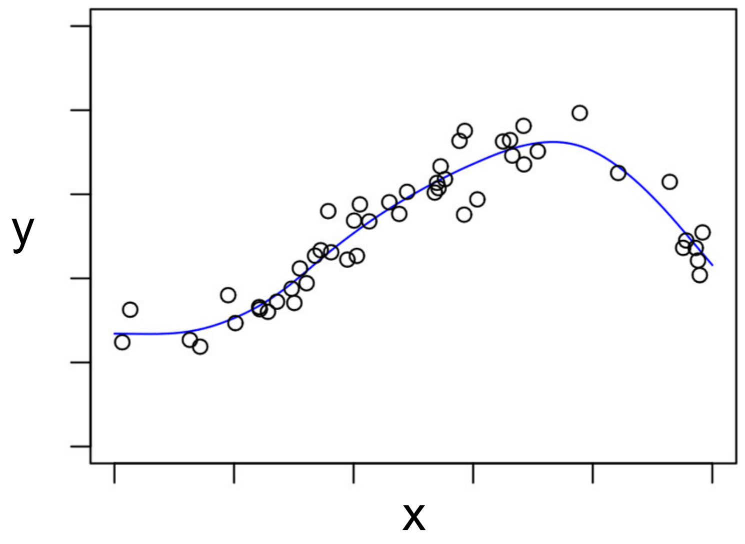
Accurate on new observations of y

(out-of-sample accuracy)

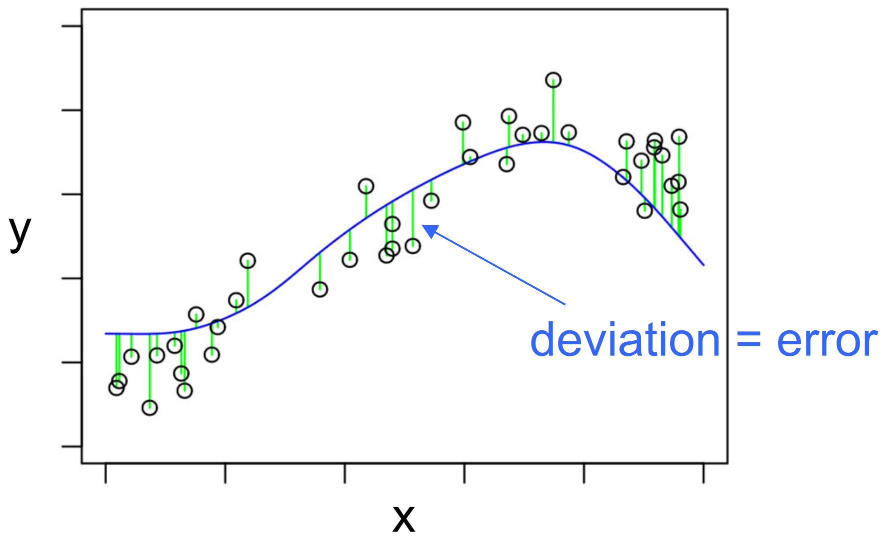
Inference algorithm

Basic idea: out-of-sample validation

Fit model to training dataset



Test model on validation dataset



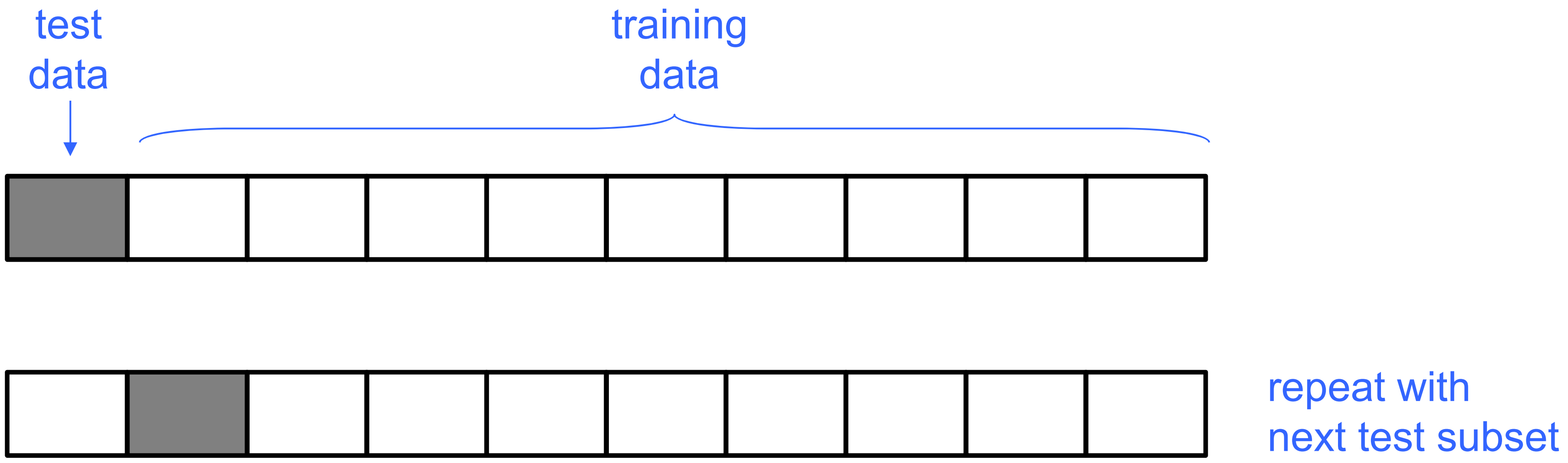
e.g. mean square error (MSE)

Some CV 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