# Reminders

- Sign up for GitHub
- Send me your username
- Ask questions on Piazza

# Machine Learning

- 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

# Basic full ML setup

- Polynomial example, 3 algorithms:
  - model: flexible function $\hat{f}(x)$; polynomial linear model

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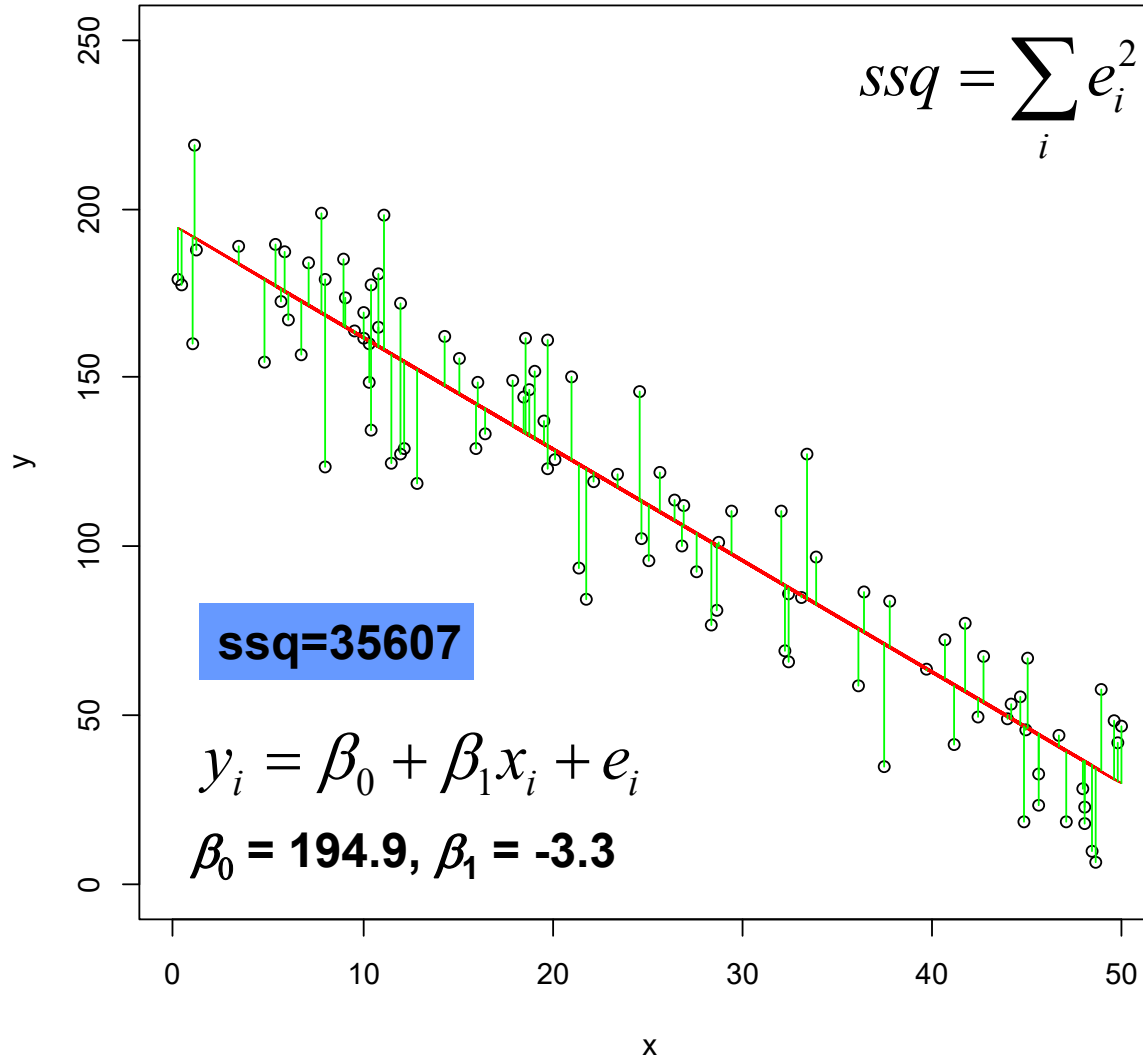
  - training: optimize least squares objective function
  - minimize $SSQ = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ for training data
  - inference: tuning parameter (order of poly); k-fold cross validation $\rightarrow$ prediction error

# Code

- 02_2_ants_cv_polynomial.R

# Least squares optimization



$$ssq = \sum_i e_i^2$$

ssq=35607

$$y_i = \beta_0 + \beta_1 x_i + e_i$$

$\beta_0$ = 194.9, $\beta_1$ = -3.3

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
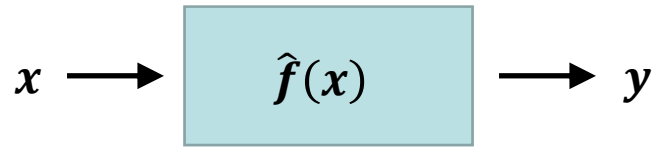
# Linear regression in R uses strategy 4

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



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

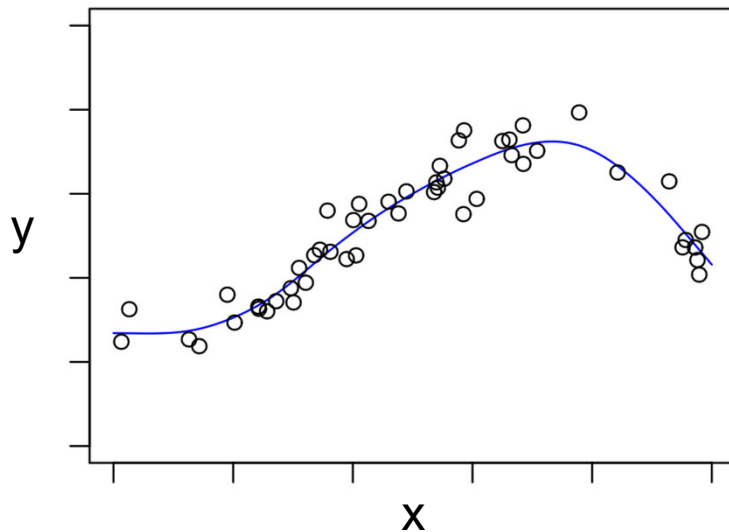Accurate on new observations of y

(out-of-sample accuracy)

# Tuning parameters

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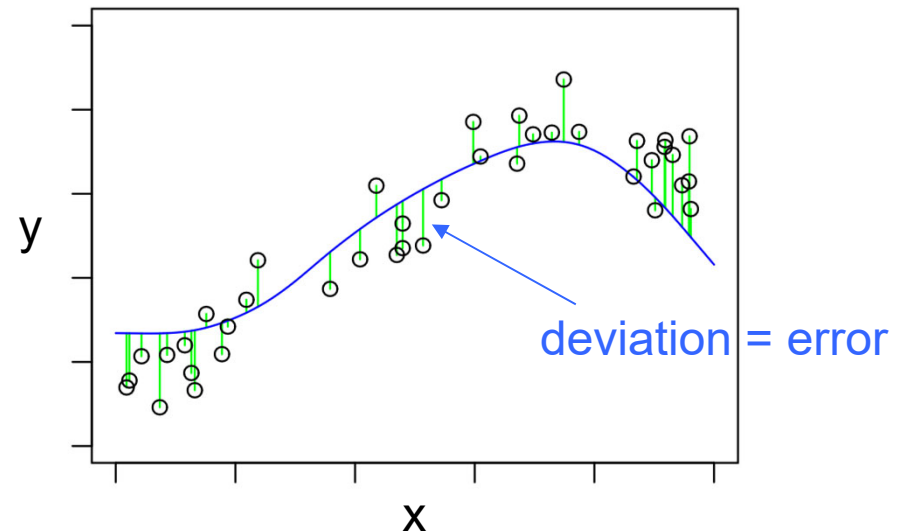- Use CV inference algorithm to choose model with best predictive performance

# Inference algorithm

Basic idea: out-of-sample validation
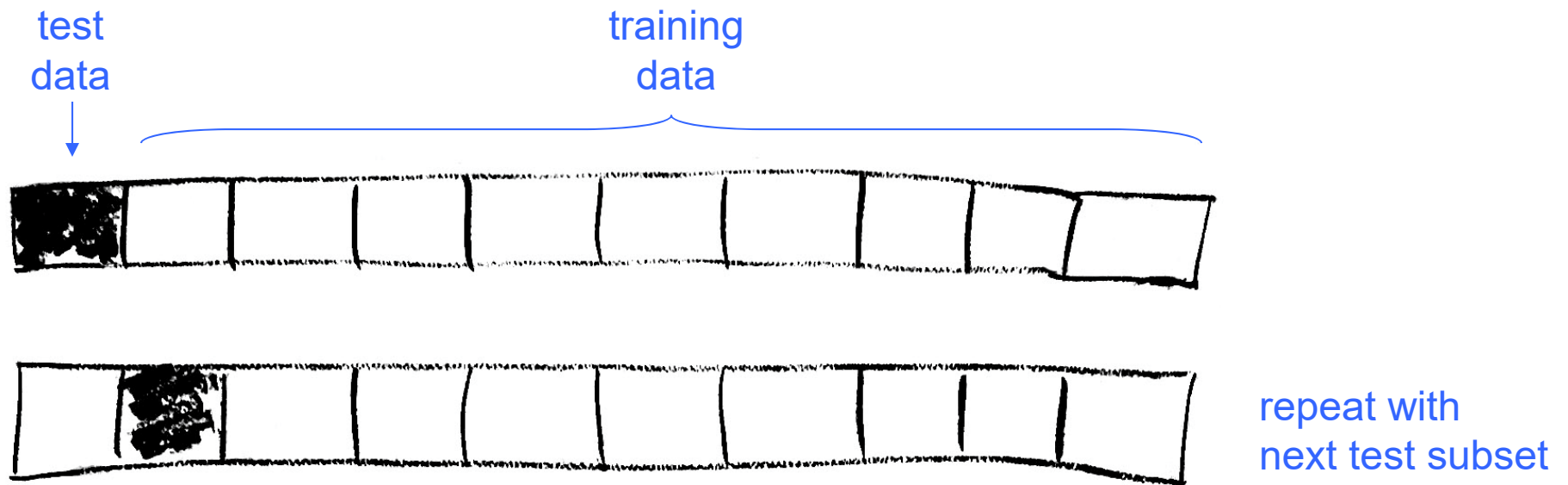
Fit model to training dataset

Test model on validation dataset



deviation = error

e.g. mean square error (MSE)

# k-fold cross validation (CV)

Divide dataset into k parts (preferably randomly)



test
data

training
data

repeat with
next test subset

... 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