Evaluating models fairly

Yingming Li
yingming@zju.edu.cn

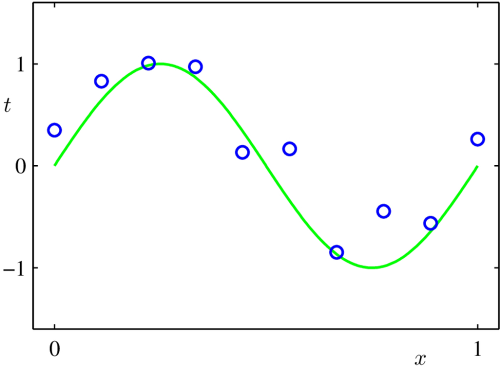
Data Science & Engineering Research Center, ZJU

20th March 2018

Adapted from slides provided by Prof. Michael Mandel.

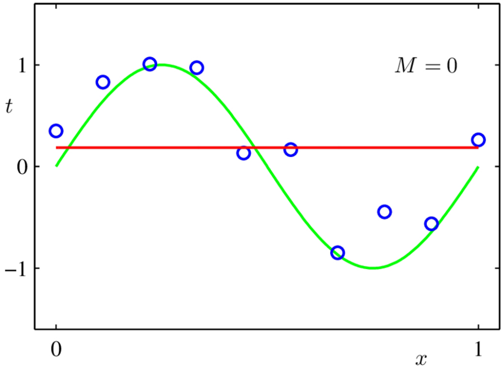
Model complexity

- More complex models can fit more complex data



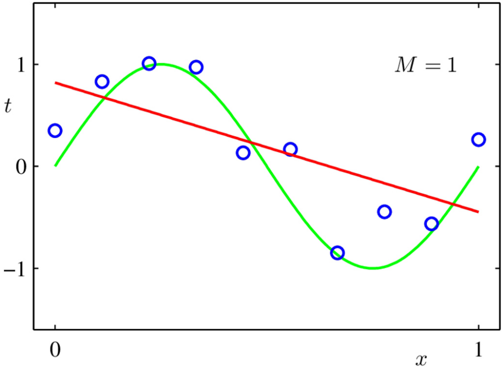
Model complexity

- Fit a polynomial of order 0



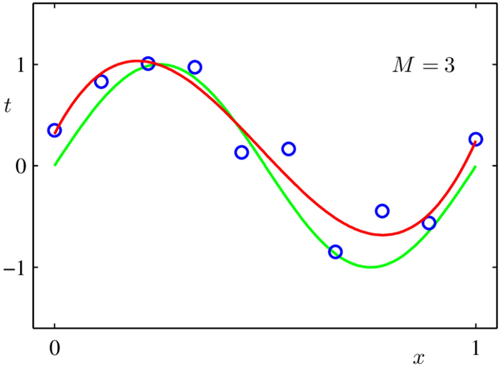
Model complexity

- Fit a polynomial of order 1



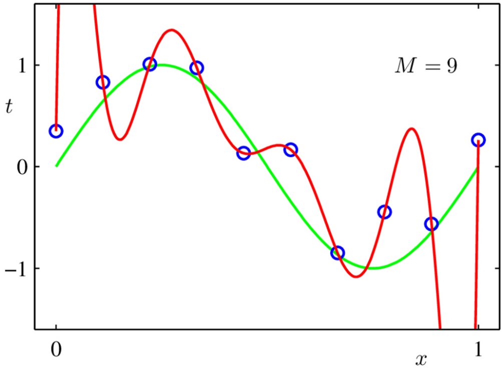
Model complexity

- Fit a polynomial of order 3



Model complexity

- Fit a polynomial of order 9

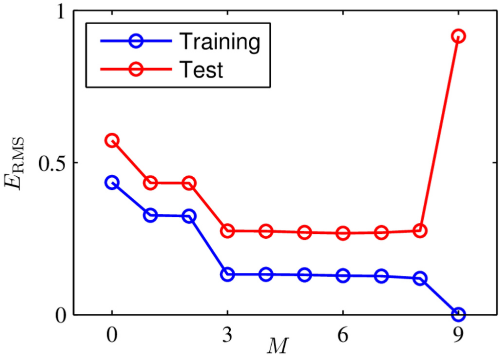


Model complexity

- Goal of training a machine learning model is **generalization**
    - After training on a given set of data
    - How good will the predictions be on new, unseen data?
- Create a separate set of data, unseen at training time
    - The test set
    - Measure performance on it

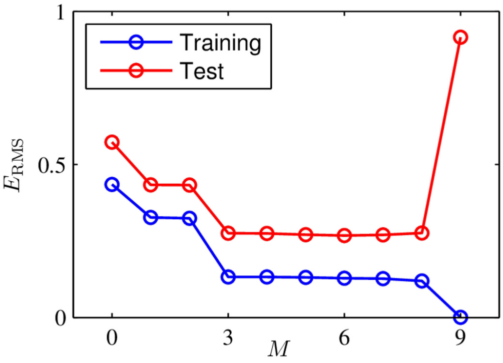
Model complexity

- Prediction error on training and test sets



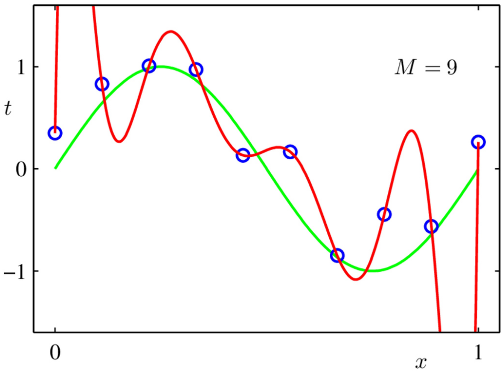
Over- vs under-fitting

	Under-fit	Good fit	Over-fit
Training error	High	Low	Low
Testing error	High	Low	High



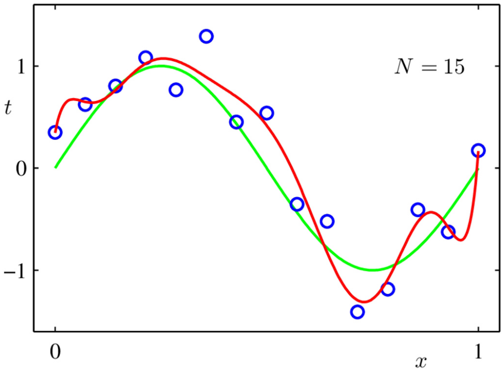
Fit depends on amount of data

- Fit a polynomial of order 9, 10 points



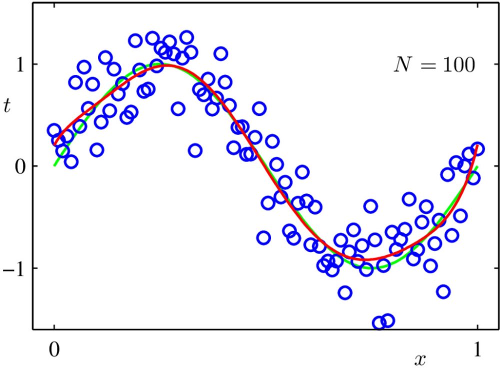
Fit depends on amount of data

- Fit a polynomial of order 9, 15 points



Fit depends on amount of data

- Fit a polynomial of order 9, 100 points



Parameter tuning

- What if your model has parameters that need to be tuned?
- Need to compare different parameter settings on unseen data
- Then need to measure the final selected model on *new* unseen data

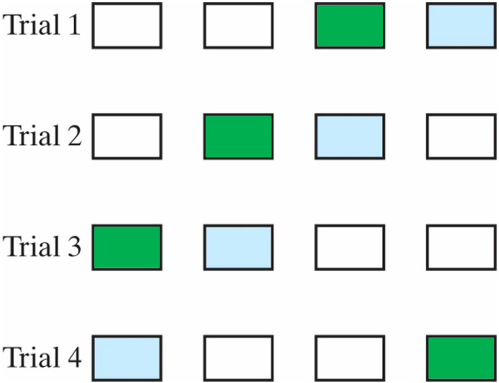
Parameter tuning

- 3-way division of data: training, validation, and test
  - Train models with different parameters on training set
  - Measure their performance on validation set
- Makes fair comparison of models' abilities to generalize beyond the training set
- Select the best-performing model as the final model
  - Measure *only* its performance on the test set
  - Gives fair estimate of whole system's ability to generalize to new data (i.e., beyond the training and validation sets)

Selecting model parameters: (cross-)validation

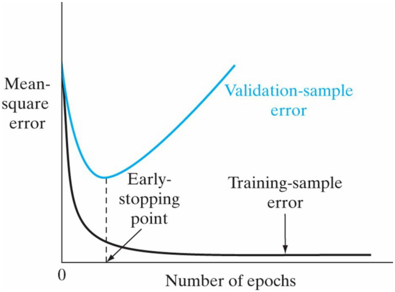
- When lots of data is available, use dedicated sets
- When data is scarce, use k -fold cross-validation
  - Partition N data points into k sets
  - Designate one set as the test set
  - Designate one of the remaining sets as the validation set
  - Train on the rest, select model on validation
  - Test on test set
  - Rotate through the data so that each set is tested on once
- Provides unbiased estimate of performance when training on $Nk - 1/k$ points and testing on N points

Cross validation illustration



Early stopping

- Now back to MLPs
- Measure performance on the validation set of models trained for different numbers of epochs
- Keep the model with the best validation performance
    - And stop training when it looks like a better one isn't coming



Thank you!