Machine Learning Techniques

DATASCI 420

Lesson 02-1 Pitfalls in Machine Learning

Generalization

Training set (labels known)

Test set (labels unknown)

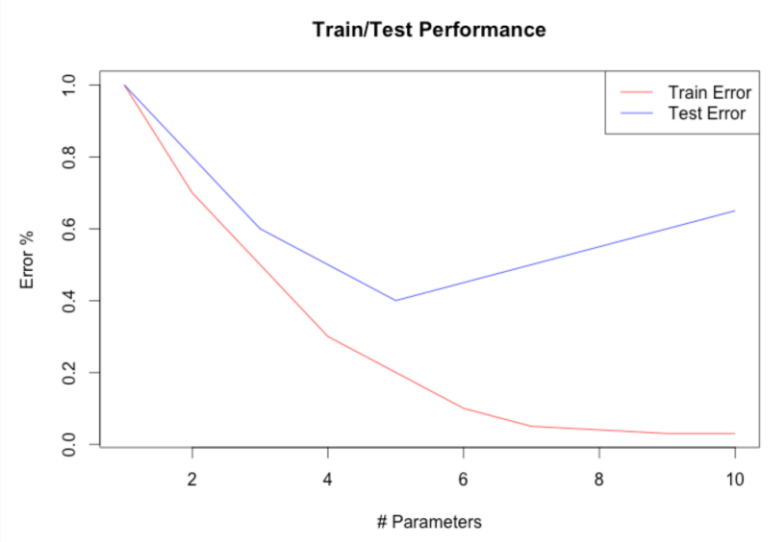
How well does a learned model generalize from the data it was trained on to a new test set?

Generalization

- What does the model generalization mean?
  - We say a model generalizes well, meaning the model achieve similar performance on the training and validation data
  - We need to split the original dataset into training and validation, in order to test the generalization of models. Usually 70-80% in training, and remainder in validation.
- **Underfitting:** model is too "simple" to represent all the relevant class characteristics
  - High training error and high validation error
- **Overfitting:** model is too "complex" and fits irrelevant characteristics (noise) in the data
  - Low training error and high validation error (big gap between training and validation performance)

Common Pitfalls in Machine Learning

- Overfitting

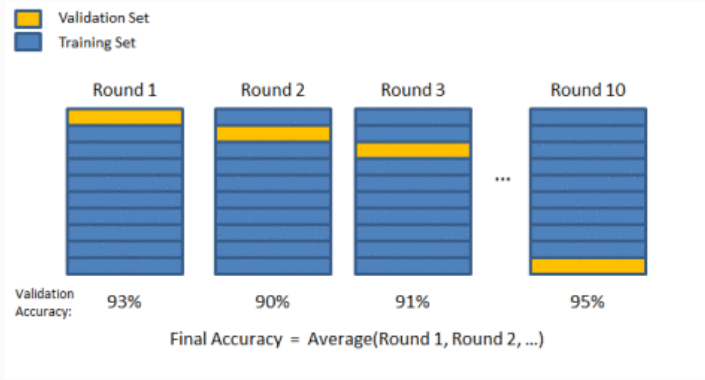


- Target leaking

- Model has good performance on validation, but not applicable
  - Have to think about when the model is in production, whether you have data available for the variables of this model when prediction is made

Common Pitfalls in Machine Learning

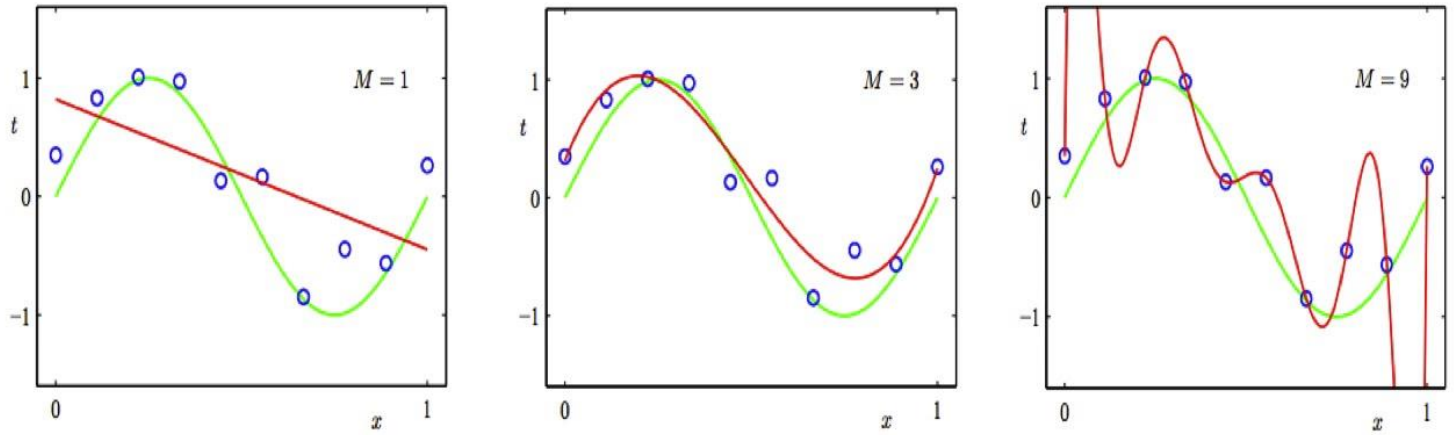
- Overfitting
  - Split the data into training and validation, and only care about the performance on validation
  - Cross validation.



- Target leakage:
  - Predicting readmission. You have one binary variable "readmission", which is your target column. You also have columns "readmission time", "readmission location", "readmission reason".
- Model has good performance on validation, but not applicable

Under- and Over-fitting examples

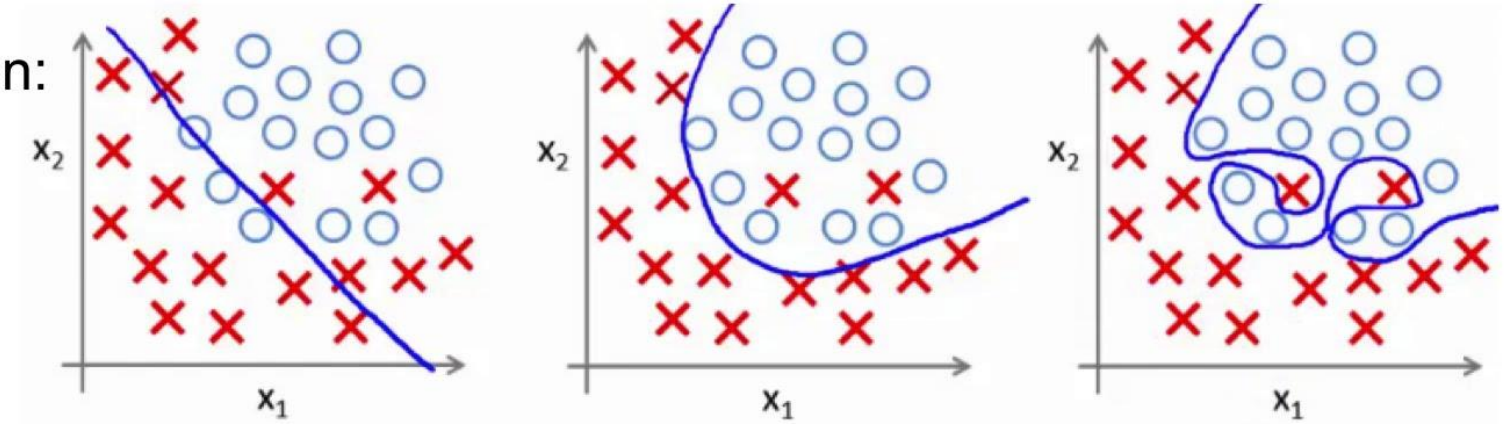
Regression:



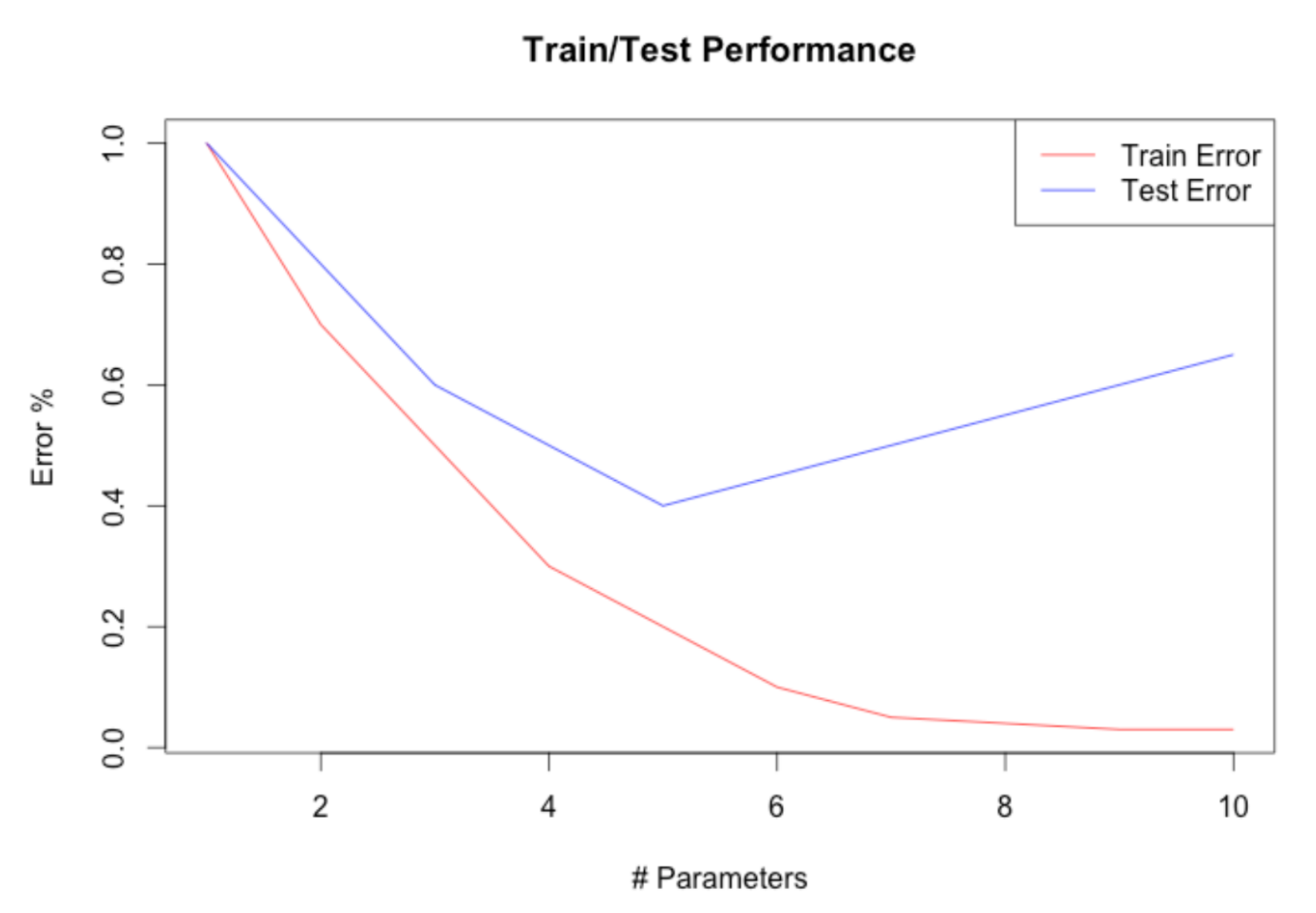
predictor too inflexible:
cannot capture pattern

predictor too flexible:
fits noise in the data

Classification:



Indicators of Underfitting and Overfitting



- Model performs poorly on both training and testing data
  - Underfitting, or
  - Not relevant data
- Model performs well on training, but poorly on testing
  - Overfitting

Reducing Underfitting

- Increase model complexity, for, e.g.
  - Increase the number of levels in a decision tree
  - Increase the number of hidden layers in a neural network.
  - Decrease the number of neighbors (k) in k-NN
- Increase the number of features, or create more relevant features
- In iterative training algorithms, iterate long enough so that the objective function has converged.

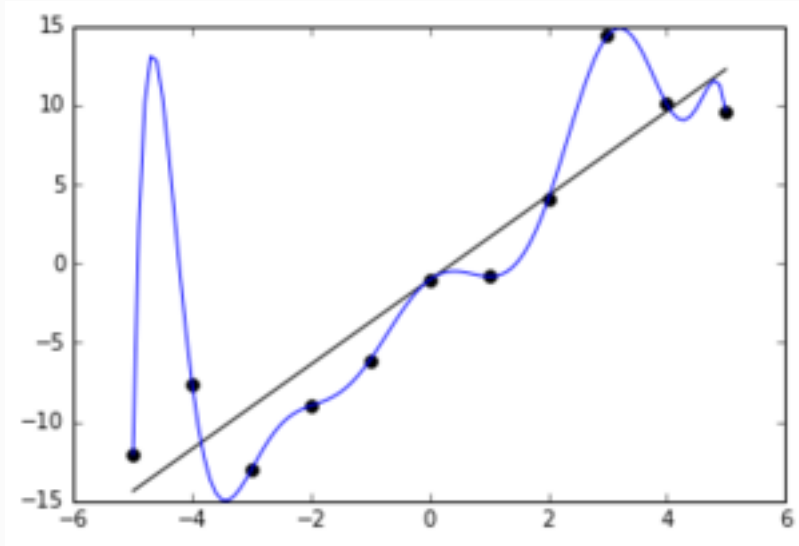
Reducing Overfitting

- Decrease model complexity, for, e.g.
  - Prune a decision tree
  - Reduce the number of hidden layers in a neural network.
  - Increase the number of neighbors (k) in k-NN
- Decrease the number of features
  - More aggressive feature selection
- Regularization (control feature complexity)
  - Penalize high weights.
  - L-1 regularization (LASSO) very efficient at pushing weights of non-informative features to 0.
- Gather more training data if possible
- In iterative training algorithms, stop training earlier to prevent "memorization" of training data

Regularization: A Popular Way of Controlling Overfitting

- Loss Function of Training
  - You can almost always increase the complexity of f_θ to reduce SSE
  - Increase the risk of overfitting
- Add regularization to control overfitting
  - L1 (LASSO) or L2 (Ridge regression) regularization

$$SSE = \sum_{i=1}^{n} (y_i - f_\theta(x_i))^2$$



$$LOSS = \sum_{i=1}^{n} (y_i - f_\theta(x_i))^2 + \lambda_1 \sum_{k=1}^{m} |\theta_k| + \lambda_2 \sum_{k=1}^{m} \theta_k^2$$

$\lambda_1, \lambda_2 \geq 0$

$\lambda_1 = 0, \lambda_2 > 0$: Ridge regression

$\lambda_2 = 0, \lambda_1 > 0$: LASSO

$\lambda_1, \lambda_2 > 0$: Elastic net

What to remember about classifiers

- Try simple classifiers first
- Better to have smart features and simple classifiers than simple features and smart classifiers
- Use increasingly powerful classifiers with more training data