MODEL SELECTION

Dr. Hilman F. Pardede

Research Center for Informatics
Indonesian Institute of Sciences

The materials are compiled from the following resources:

- https://github.com/joaquinvanschoren/ML-course
- https://www.cse.iitk.ac.in/users/piyush/courses/ml_autumn16/ML.html
- http://sli.ics.uci.edu/Classes/2015W-273a

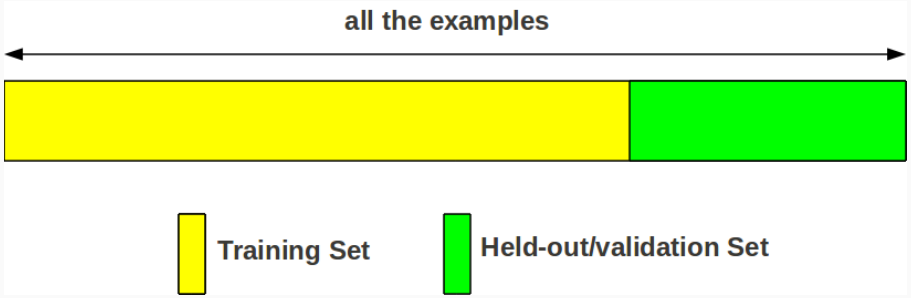
LINEAR MODELS

Given a set of models $M = \{M_1, M_2, ..., M_R\}$, choose the model that is expected to do the best on the test data. The set M may consist of:

- Instances of same model with different complexities or hyperparams. E.g.,
  - K -Nearest Neighbors: Different choices of K
  - Decision Trees: Different choices of the number of levels/leaves
  - Polynomial Regression: Polynomials with different degrees
  - Kernel Methods: Different choices of kernels
  - Regularized Models: Different choices of the regularization hyperparameter
- Different types of learning models (e.g., SVM, KNN, DT, etc.)

HELD-OUT DATA

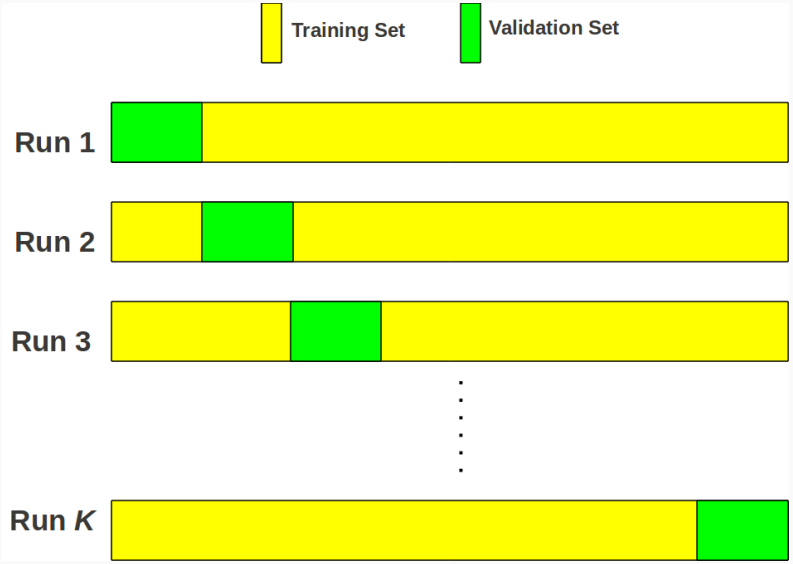
- Set aside a fraction of the training data. This will be our held-out data.
  - Other names: validation/development data.



  - Remember: Held-out data is NOT the test data. DO NOT peek into the test data during training
    - Train each model using the remaining training data
    - Evaluate error on the held-out data (cross-validation)
    - Choose the model with the smallest held-out error
    - Problems:
  - Wastes training data. Typically used when we have plenty of training data
  - What if there was an unfortunate train/held-out split?

K-FOLD CROSS VALIDATION

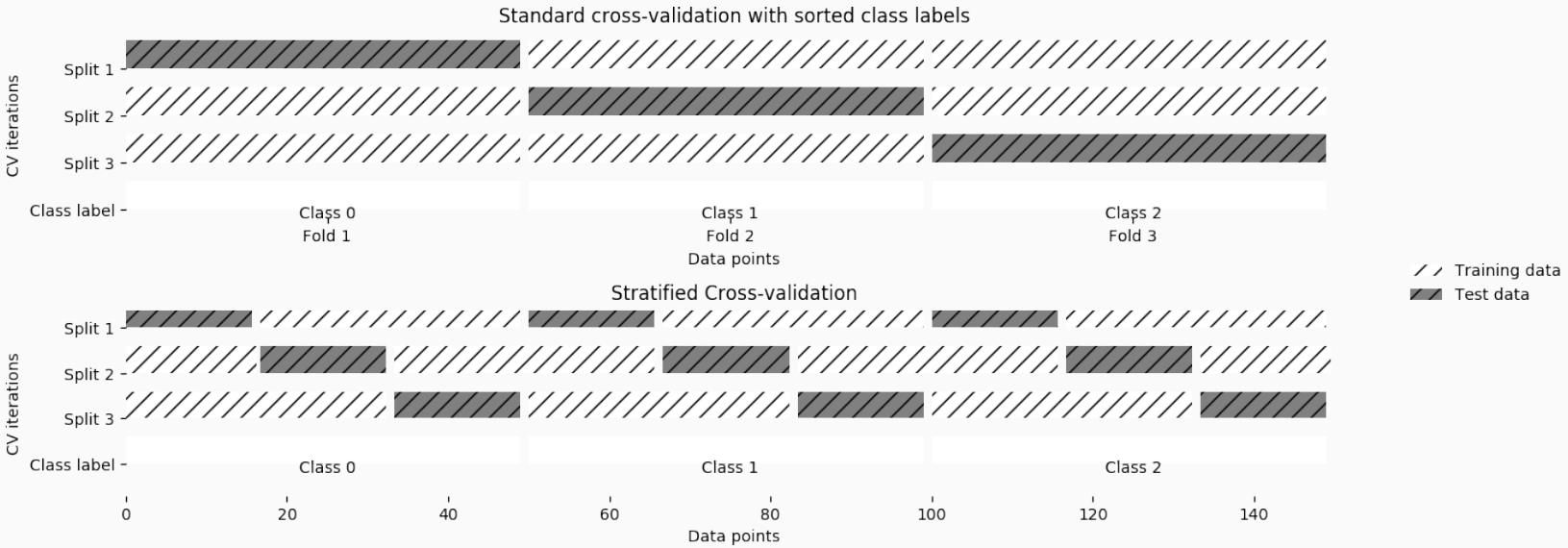
- Create K (e.g., 5 or 10) equal sized partitions of the training data
- Each partition has N/K examples
- Train using $K - 1$ partitions, validate on the remaining partition
- Repeat this K times, each with a different validation partition



- Average the K validation errors
- Choose the model that gives the smallest average validation error

STRATIFIED K-FOLD CROSS-VALIDATION

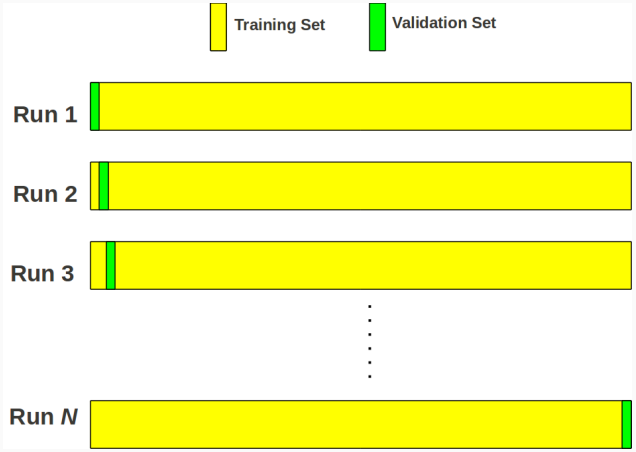
- If the data is unbalanced, some classes have many fewer samples
- Likely that some classes are not present in the test set
- Stratification: make sure that proportions between classes are conserved in each fold
  - Order examples per class
  - Separate the samples of each class in k sets (strata)
  - Combine corresponding strate into folds



Leave-one-out (LOO) cross-validation

Special case of K -fold CV when $K = N$

- Each partition is now a single example
- Train using $N - 1$ examples, validate on the remaining example
- Repeat the same N times, each with a different validation example



- Average the N validation errors. Choose the model with smallest error
- Can be expensive in general, especially for large N
  - But very efficient when used for selecting the number of neighbors to consider in nearest neighbor methods (reason: NN methods require no training)

CHOOSING A PERFORMANCE ESTIMATION PROCEDURE

No strict rules, only guidelines:

- Always use stratification for classification
- Use holdout for very large datasets (e.g. $>$ 1.000.000 examples)
  - Or when learners don't always converge (e.g. deep learning)
- Choose k depending on dataset size and resources
  - Use leave-one-out for small datasets (e.g. $<$ 500 examples)
  - Use cross-validation otherwise
    - Most popular (and theoretically sound): 10-fold CV
- Use grouping or leave-one-subject-out for grouped data

BIAS-VARIANCE DECOMPOSITION

- three sources of errors:

$$E[(y - \hat{f}(x))^2] = Bias[f(x)]^2 + Var[f(x)] + \sigma^2 \quad (1)$$

  - Bias: systematic error (independent of the training sample). The classifier always gets certain points wrong (perhaps due too simple model)
  - Variance: error due to variability of the model with respect to the training sample. The classifier predicts some points accurately on some training sets, but inaccurately on others.
  - Intrinsic (noise) error, but there's nothing we can do against that.
- Bias is associated with underfitting, and variance with overfitting
- Bias-variance trade-off: you can often exchange bias for variance through (de)regularization
- The challenge is to find the right trade-off (minimizing total error)

How to measure bias and variance (for regression):

- Take 100 or more bootstraps (or shuffle-splits)
- For each data point x:
  - $bias(x) = (x_{true} - mean(x_{predicted}))^2$
  - $variance(x) = var(x_{predicted})$
- Total bias: $\sum_x bias(x) * w_x$, with w_x the ratio of x occuring in the test set
- Total variance: $\sum_x variance(x) * w_x$

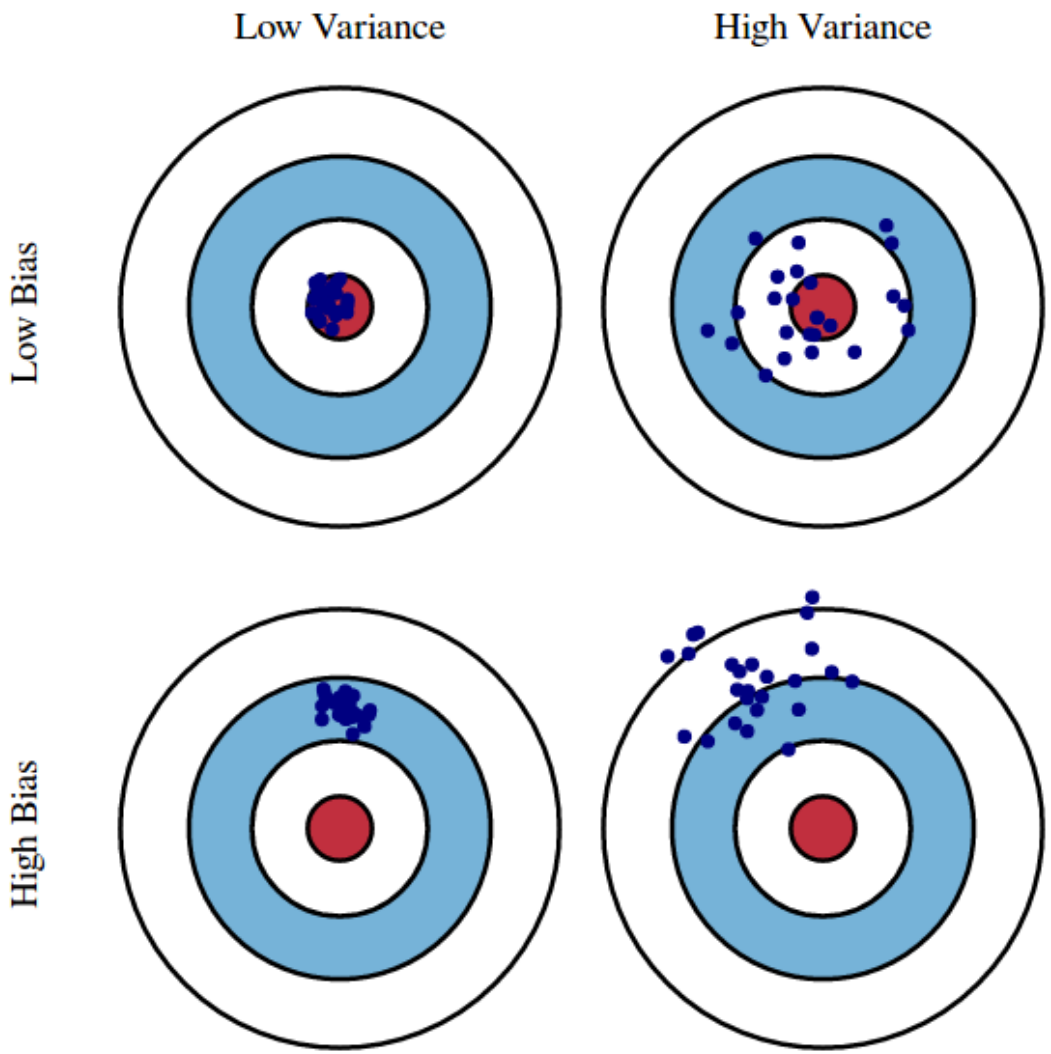


Fig. 1 Graphical illustration of bias and variance.

Now that we know how to evaluate models, we can improve them by tuning their hyperparameters

- Grid search
- Random search
- Local search
- Racing algorithms
- Model-based optimization
- Multi-armed bandits
- Genetic algorithms

GRID SEARCH

- For each hyperparameter, create a list of interesting/possible values
  - E.g. For kNN: k in $[1, 3, 5, 7, 9, 11, 33, 55, 77, 99]$
- Evaluate all possible combination of hyperparameter values
  - E.g. using cross-validation
- Select the hyperparameter values yielding the best results

RANDOM SEARCH

- Grid Search has a few downsides:
  - Optimizing many hyperparameters creates a combinatorial explosion
  - You have to predefine a grid, hence you may jump over optimal values
- Executing random search in scikit-learn:
  - `RandomizedSearchCV` works like `GridSearchCV`
  - Has n_iter parameter for the number of iterations
  - Search grid can use distributions instead of fixed lists