Ensemble Methods

AW

Lecture Overview

1. **Intro to Ensemble Methods**

2. **Tuning and Evaluation**

Ensemble Methods

- Train multiple classifiers (learners) and combine them
- Achieve higher accuracy
- Used in the real-world applications

Ensemble Methods

- Boosting – binary classification
- Bagging – multi-class method, less sensitive to noise
- Random Forest – tree ensemble, viewed as a variant of bagging
- Combination methods
    - Train multiple classifiers (learners) and combine them
    - Achieve higher accuracy
    - Used in the real-world applications
  - Averaging, voting, stacking...

Boosting

- "Boosting" weak classifiers to strong classifiers
- Procedure
  - Take weak classifier h1 that has correct classification in spaces X1 and X2 and wrong in X3. Thus, 1/3 error.
  - Take X3 and derive new distribution from it to train a new classifier h2
    - If accuracy is better, we are done
    - Otherwise combine h1 and h2 using appropriate method

AdaBoost Algorithm

- Initialize the weight distribution from data set
- Train classifier h1 and evaluate its error
- If error is good enough stop.
- Otherwise update the distribution using exponential loss function:

$$\ell_{\exp}(h \mid \mathcal{D}) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[e^{-f(\boldsymbol{x})h(\boldsymbol{x})}]$$

Bagging

- Parallel ensemble method – base classifiers are generated in parallel
- Bagging combines independent base classifiers (boosting uses dependence between the base learners, see h1 and h2)
- Favorable for parallel computing, which means also good for modern multi-core CPUs
- Key: to get the base learners as independent as possible

The Bagging Algorithm

- Utilizes bootstrap sampling for generating different base classifiers
  - Given m number f training examples, a sample of m training examples will be generated by sampling with replacement.
  - Some original examples might appear multiple times
  - Some might be missing
- Adopts voting and averaging for aggregating the outputs of base classifiers

The Bagging Algorithm (cont.)

- Define the number of bootstrap samples T
- For 1 to T, create bootstrap sample and train the base classifier on it.
- Evaluate error by taking maximum error of the base classifier

Combination Methods

- Instead of finding the best classifier, combine them.
- Issues:
  - Chosen hypotheses might not predict the future data very well
  - Combination methods use local search algorithm that might not be optimal (only sub-optimal)

Averaging

- Most popular
- Used for numeric outputs
- Simple:

$$H(\boldsymbol{x}) = \frac{1}{T}\sum_{i=1}^{T} h_i(\boldsymbol{x})$$

- Weighted:

$$H(\boldsymbol{x}) = \sum_{i=1}^{T} w_i h_i(\boldsymbol{x})$$

(note: when all w_i are the same – simple averaging)

Voting – Majority

- Used for nominal outputs
- Majority voting
  - Every classifier votes for one class
  - Class with more than half of the votes is the output class label. If not more than half, combined classifier makes no prediction (called rejection option)

Voting - Pluraliry

- Plurality voting
  - Every classifier votes for one class
  - The class label with the most votes wins
  - Ties decided arbitrarily depending on the algorithm (for example randomly)

Voting – Weighted

- Weighted voting
  - Every classifier votes for one class
  - Not all classifier are of equal importance
  - Gives more power to the better classifiers
- Based on data, some classifiers might be inherently better than the others
  - Same as plurality but the votes are scaled using given weight

Soft Voting

- Used for classifier that produce class probability outputs
- Each classifier's vote is vector consisting of probabilities for each class
- Simple soft voting then generated the combined output by averaging all individual outputs
- Weighted soft voting can be considered and its application is similar to the weighted voting

Stacking

- Called combining by learning
- Two levels of classifiers:
  - First-level are individual classifiers
  - Second-level are learners that are trained to combine the individual learners (also called meta-learner)

Stacking

- Idea:
  - train first-level classifiers using original data
  - then generate new data for training the second-level classifier, where outputs of the first-level classifiers are regarded as input features and original labels are still regarded as labels of the new training data
- First-level often use different algorithms

R - Bagging

- Bagged CART (rpart):

```
control <- trainControl(method="repeatedcv",
number=10, repeats=3)

fit.treebag <- train(Class~., data=dataset,
method="treebag", metric=metric,
trControl=control)
```

R - Boosting

- C5.0 (boosting of C4.5):

```
control <-
trainControl(method="repeatedcv",
number=10, repeats=3)

fit.c50 <- train(Class~., data=dataset,
method="C5.0", metric=metric,
trControl=control)
```

R - Stacking

- Example with LDA, Rpart and SVM
  - SVM and LDA are linear classifiers that we'll study shortly

```
control <- trainControl(method="repeatedcv",
number=10, repeats=3, savePredictions=TRUE,
classProbs=TRUE)

algorithmList <- c('lda', 'rpart', 'svmRadial')

models <- caretList(Class~., data=dataset,
trControl=control, methodList=algorithmList)
results <- resamples(models)
```

Lecture Overview

1. **Intro to Ensemble Methods**

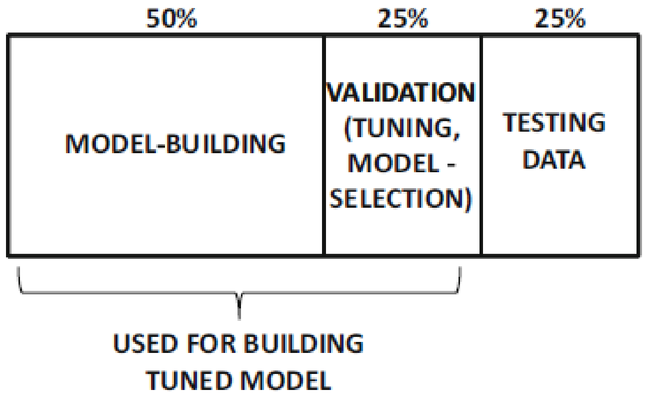
2. **Tuning and Evaluation**

Data: Training, Validation and Testing

Training data: This part of the data is used to build the training model

- Completely different algorithms may be used to build the models in multiple ways.
    - Different decision trees
    - Some other linear models
- The same algorithms might use different hyperparameters (maximum branching for DT, min number of items in leaf, confidence levels, etc.) to create different models.
- The same training data set may be tried multiple times over different choices for the hyperparameters

Data: Training, Validation and Testing (cont.)


50%
25%
25%
MODEL-BUILDING
VALIDATION (TUNING, MODEL - SELECTION)
TESTING DATA
USED FOR BUILDING TUNED MODEL

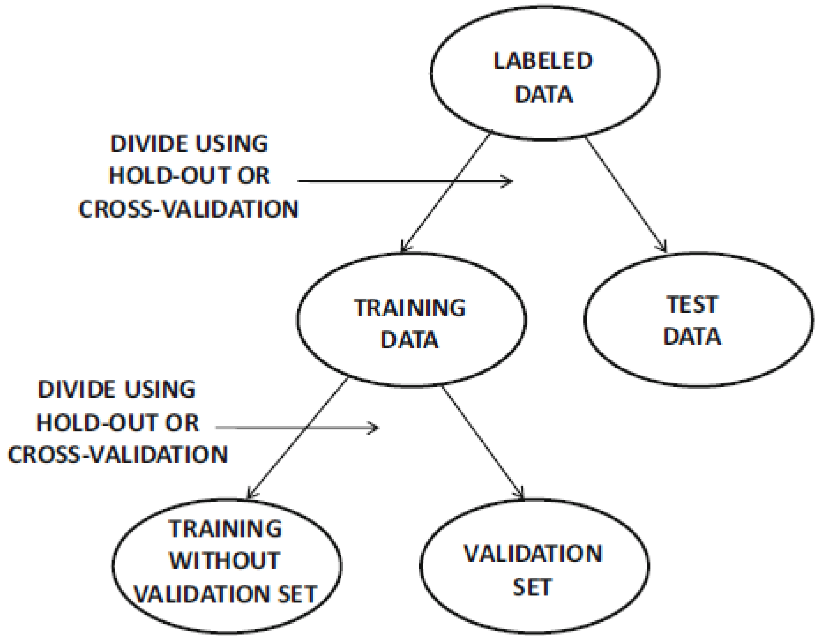
Validation data: Data set used model for selection and parameter tuning.

- *Model selection*, is the process of selection of best algorithm among different models. Actual *evaluation* of these algorithms for selecting the best model is done on a separate validation data set to avoid favoring overfitted models
  - For example, the choice of the learning rate may be tuned by constructing the model multiple times on the training data, and then using the validation set to estimate the accuracy of these different models.
- <u>Validation data is the part of model building</u>

Holdouts and Cross-Validation

Hold-out method:

- a fraction of the instances are used to build the training model.
- The remaining instances (*held-out* Instances) are used for testing. The accuracy of predicting the labels of held-out instances is reported as the overall accuracy.


LABELED DATA
DIVIDE USING HOLD-OUT OR CROSS-VALIDATION
TRAINING DATA
TEST DATA
DIVIDE USING HOLD-OUT OR CROSS-VALIDATION
TRAINING WITHOUT VALIDATION SET
VALIDATION SET

Cross-validation method:

- Labeled data is divided into q equal segments. One of q sets is used for testing, the remaining $(q-1)$ segements are used for training.
- This process is repeated *q* times by using each of the *q* segments as the test set. The average accuracy over the *q* different test sets is reported.

Holdouts and Cross-Validation: Pros and Cons

Hold-out method:

- underestimates the true accuracy: frequency of the held-in examples will always be inversely related to that of the held-out examples ⇒ leads to a pessimistic bias in the evaluation.
  - Unless we sample each class separately wrt probabilities, the held-out examples have a higher presence of a particular class than all labeled data set ⇒ held-in examples have a lower average presence of the same class. Causes a mismatch between the training and test data
  - Simple and efficient

Cross-validation:

- Can closely approximate the accuracy,
- it is usually too expensive to train the model a large number of times. Cross-validation is sparingly used in neural networks because of efficiency issues.

Sources

- Introduction to Data Mining by Kumar, Steinbach, Tan
  - Sections 3.6, 4.10.1-4.10-5
- R implementation: http://machinelearningmastery.com/machine-learning-ensembles-with-r/