http://goo.gl/Jv7Vj9

Course website

KYOTO UNIVERSITY

Statistical Machine Learning Theory

Model Evaluation

Hisashi Kashima
kashima@i.Kyoto-u.ac.jp

DEPARTMENT OF INTELLIGENCE SCIENCE
AND TECHNOLOGY

Topics:
Performance measures and evaluation frameworks

- You want to know the final performance of your model,
  or select the best one among possible models (or both)
- Performance measure: accuracy, precision/recall, DCG@k, AUC
- Evaluation framework: cross validation
- Model stacking

Performance Measures

Various performance measures: Should be chosen according to your applications

- There are various evaluation measure to quantify the performance of a trained model especially in supervised classification
  - Accuracy, precision/recall, DCG@k, AUC, …
- They should be appropriately chosen depending on applications
  - Classification with decision thresholds: accuracy, precision/recall, …
  - Classification without decision thresholds: AUC, …
  - Ranking: DCG@k, …

Decision model and confusion matrix:
Decisions on a dataset give a confusion matrix

- The trained model gives confidence $f(\mathbf{x})$ on given instance $\mathbf{x}$ belonging to the positive class $(+1)$
- Assign $+1$ to $\mathbf{x}$ whose $f(\mathbf{x})$ is larger than decision threshold τ
- Fixing a model, a dataset, and a decision threshold gives a confusion matrix

		predicted label	
		positive	negative
true label	positive	#true positives ☺	#false negatives
	negative	#false positives	#true negatives ☺

Accuracy, precision, recall, and F-measure: Basic predictive performance measures

- Accuracy: percentage of $\frac{\#\text{true positives} + \#\text{true negatives}}{\#\text{all predictions}}$

- Precision/Recall

		predicted label	
		positive	negative
true label	positive	#true positives ☺	#false negatives
	negative	#false positives	#true negatives ☺

  – $\text{Precision} = \frac{\#\text{true positives}}{\#\text{true positives} + \#\text{false positives}}$

  – $\text{Recall} = \frac{\#\text{true positives}}{\#\text{true positives} + \#\text{false negatives}}$

  – $\text{F}-\text{measure} = \frac{\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}}$

    • an integrated measure of precision and recall

DCG@k: Performance measure for ranking

- In ranking (of web pages), accuracy of top-ranked items is more important
- Precision@k: precision calculated using the top-k scored items
- DCG(Discounted Cumulative Gain)@k is a weighted variant of Precision@k: $\sum_{i=1}^{k} \frac{\mathrm{rel}(i)}{\log(i+1)}$
  - $\mathrm{rel}(i)$ is the relevance score for the i-th ranked item

AUC:
Performance measure not depending on the threshold

- Evaluation needs fixing the decision threshold
- Imbalanced data generally results in a high accuracy
- AUC:
  - A performance measure directly defined with confidence score $f(\mathbf{x})$
  - Probability of A being larger than B
    - A: confidence score of a randomly chosen positive instance
    - B: confidence score of a randomly chosen negative instance
  - takes 1 for perfect predictions, 0.5 for random predictions

Evaluation Framework

Evaluation framework:
We want to predict model performance

- Performance of a model for training data and that for future data are different
  - What we are interested in is the latter
- Many models have hyper-parameters to be specified by users
  - We want to tune them so that the final performance gets better

First principle:
Evaluation must use a dataset not used in training

- You must not evaluate your classifier on the dataset you used for training
- Usually, first divide a given dataset into a training dataset and a test dataset
  1. Train a classifier using the training dataset
  2. Evaluate its performance on the test dataset
- Sometimes ordering of data instances (unintentionally) has some patterns in their labels
  - Partitioning should be done carefully

Cross validation (for performance testing): A statistical framework for performance evaluation

- You want to know the performance of the classifier (will be obtained using your algorithm) when it is deployed
- (K-fold) cross validation do this
- Divide a given dataset into K non-overlapping sets
  - Use $K - 1$ of them for training
  - Use the remaining one for testing
- Changing the "test" datasets results in K measurements
  - Take their average to get a final performance estimate

Cross validation for tuning hyper-parameters: A statistical framework for performance evaluation

- Most of machine learning algorithms have hyper-parameters
  - Hyper-parameters: Parameters not automatically tuned in the training phase; given by users
- (K-fold) cross validation can be used for this
  - Use $K - 1$ of K sets for training models for various hyper-parameter settings
  - Use the remaining one for testing
  - Choose the hyper-parameter setting with the best averaged performance
    - Note that this is **NOT** its final performance estimate

Double loop of cross validation: Tuning hyper-parameters and performance evaluation

- Sometimes you want to do both hyper-parameter tuning and performance evaluation
- Doing both with one K-fold cross validation is guilty
  - You see the test for tuning hyper-parameters
- Double loop cross validation
  - Outer loop for performance evaluation
  - Inner loop for hyper-parameter tuning
  - High computational costs…

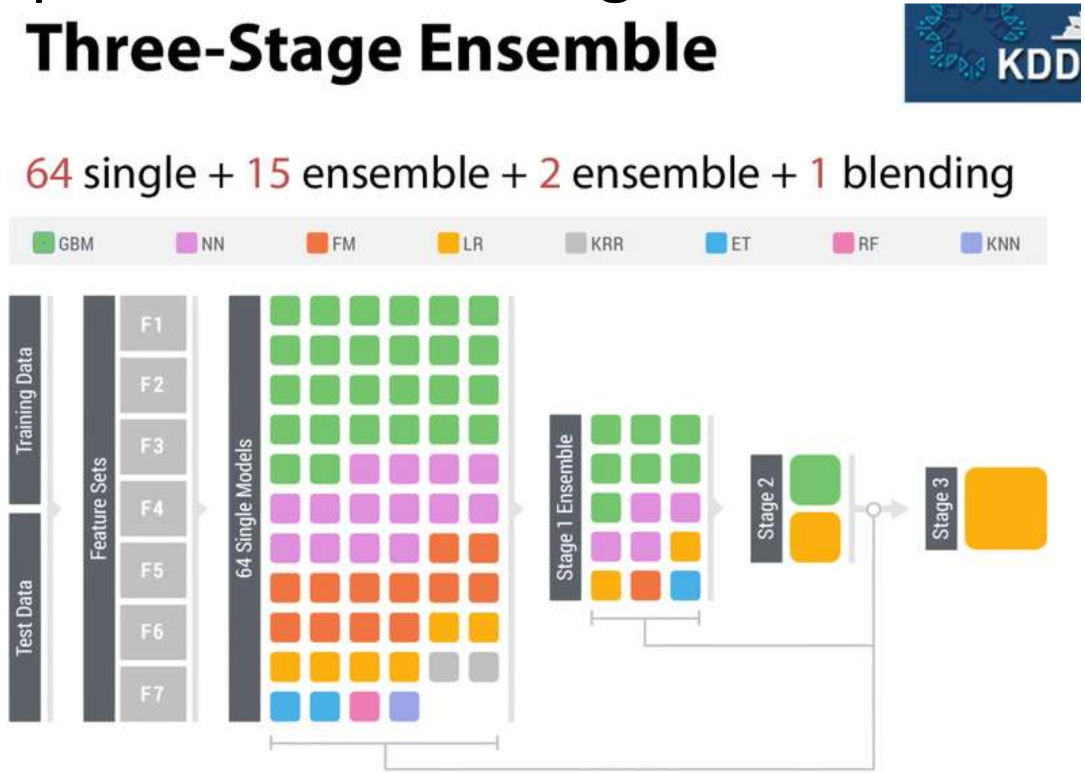
A simple alternative of double-loop cross validation: "Development set" approach

- A simple alternative for the double-loop cross validation
- "Development set" approach
  - Use $K-2$ of K sets for training
  - Use one for tuning hyper-parameters
  - Use one for testing

Model Stacking

Model ensemble:
Combines different models to improve performance

- One model cannot fit all
- Ensemble of different predictors to improve performance
- Commonly used technique in predictive modeling competitions (e.g. Kaggle)



Model stacking:
An ensemble method to combine different models

- Outputs of the level-0 models are inputs of the level-1 models
  - Original feature vector $\mathbf{x}$
  - Outputs of the level-0 models $\mathbf{y}$
  - New extended feature vector $\tilde{\mathbf{x}} = \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}$ for level-1 models
- Stacking:
  - is similar to the multi-layer neural network
    - Stacked one-layer perceptron
  - but has heterogeneous components

Difficulty in model stacking:
An easy solution is biased

- How can we train staked models?:

- An easy solution:

1. train a classifier f using the training dataset L
2. add the prediction values of f as a new feature

.... seemingly works... but actually NOT

- Remember the first principle: you cannot make a prediction for the data you used in the training
  - The prediction value to the training data are biased because your model has been trained to reproduce the labels

A solution:
Use cross-validation to extend features

- Divide a given dataset into K non-overlapping sets
  1. Use $K-1$ of them for training a model
  2. Use the model to add a new feature to the remaining set
  - Doing steps 1&2 for K holdout sets gives the new feature for the whole dataset
- Train the level-1 predictor using the extended dataset
- The level-0 predictor is (re-)trained using the original whole dataset
  - Because the extended feature for training the level-1 predictor is produced by different level-0 predictors