Statistical Machine Learning Theory

Model Evaluation and Selection

Hisashi Kashima
kashima@i.Kyoto-u.ac.jp
You *empirically* want to know the final performance of your model, or select the best one among possible models (or both)

Performance measures (especially for binary classification): accuracy, precision/recall, DCG@k, AUC

Empirical model evaluation and selection framework: cross validation

Model stacking
Performance Measures
Various performance measures of classifiers: Should be chosen according to applications

- Evaluation measures 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@$k$, ...
Confusion matrix:
Set of predictions on a dataset gives a confusion matrix

- Binary classifier makes positive (+1) or negative (−1) predictions
  - Linear classifier: \( y = \text{sign}(f(x)), f(x) = w^T x \)
  
- Once we have a set of predictions on a dataset, we have a confusion matrix:

<table>
<thead>
<tr>
<th>true label</th>
<th>predicted label</th>
</tr>
</thead>
<tbody>
<tr>
<td>positive</td>
<td>positive</td>
</tr>
<tr>
<td></td>
<td>#true positives</td>
</tr>
<tr>
<td>negative</td>
<td>#false positives</td>
</tr>
<tr>
<td></td>
<td>#true negatives</td>
</tr>
</tbody>
</table>
Accuracy, precision, recall, and F-measure: Basic predictive performance measures

- **Accuracy:** percentage of \( \frac{\text{#true positives} + \text{#true negatives}}{\text{#all predictions}} \)
  - In other words, averaged 0-1 loss

- **Precision/Recall**
  - **Precision** = \( \frac{\text{#true positives}}{\text{#true positives} + \text{#false positives}} \)
  - **Recall** = \( \frac{\text{#true positives}}{\text{#true positives} + \text{#false negatives}} \)

- **F−measure** = \( \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \)
  - Harmonic mean of precision and recall
Performance measure for ranking:
Evaluate the confidence score directly

- In ranking (of web pages), accuracy of top-ranked items is more important
  - We usually check only the first page of Google search results

- In a linear classifier: $y = \text{sign}(f(x))$, $f(x) = w^T x$,
  - $f(x)$ indicates how likely $x$ belongs to class $+1$
  - $|f(x)|$ is considered as a confidence level for the prediction

- Usually, we use a decision threshold $\tau$ to make a decision:
  - $y = \text{sign}(f(x) - \tau)$
  - Predict $+1$ for $x$ if $f(x)$ is larger than decision threshold $\tau$
Performance measure for ranking: 
**Precision@**$k$ and **DCG@**$k$

- Precision@$k$ : precision calculated using the top-$k$ scored items
  - Or, with the threshold model, we can have different precision values for different thresholds $\tau$

- DCG(Discounted Cumulative Gain)@$k$ is a weighted variant of

  **Precision@**$k$: $\sum_{i=1}^{k} \frac{\text{rel}(i)}{\log(i+1)}$

  - rel$(i)$ is the relevance score for the $i$-th ranked item
AUC: A standard performance measure of classification

- We want a performance measure that
  - is not affected by class (im)balance
    - Imbalanced data generally results in a high accuracy
  - does not depend on $k$ or $\tau$

- AUC: a performance measure directly given by confidence score $f(x)$
  - Probability of $P$ being larger than $N$
    - $P$: confidence score of a randomly chosen positive instance
    - $N$: confidence score of a randomly chosen negative instance
  - AUC=1 for perfect predictions, 0.5 for random predictions
Evaluation and Selection Framework
Model evaluation and selection framework: We want to predict final performance of models

- We are interested in the future performance of the obtained model when it is deployed
  - Model performance for training data and that for future data are different

- We often have some hyper-parameters to be tuned so that the final performance gets better
  - E.g. Training target of the ridge regression:
    \[
    \text{minimize}_w \| y - Xw \|_2^2 + \lambda \| w \|_2^2
    \]
  - Hyper-parameters are not optimized in the training
The first principle: Evaluation must use a dataset not used in training

- You *must not* evaluate your classifier based on the performance on the dataset you already used for training.

- Usually, a given dataset must be divided into a *training dataset* and a *test dataset*.

1. Train a classifier using the training dataset.
2. Evaluate its performance on the test dataset.

- Partitioning should be done carefully.
  - Time series data: if your dataset explicitly has time stamps and you are interested in predicting the future, you should divide it into past and future.
A statistical framework for performance evaluation: Cross validation

- \((K\text{-fold})\) cross validation gives an estimate of the future performance of the classifier when it is deployed

- 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 dataset results in \(K\) measurements
  - Take their average to get a final performance estimate
Statistical framework for tuning hyper-parameters: Cross validation (for model selection)

- Most of machine learning algorithms have hyper-parameters
  - Hyper-parameters are not automatically tuned in the training phase and must be given by users

- (K-fold) cross validation can also be used for this purpose:
  - 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** the estimate of its final performance
Sometimes you want to do both hyper-parameter tuning and estimation of future performance.

Doing both with one $K$-fold cross validation is guilty

- You saw the test dataset 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