Performance Evaluation of Machine Learning Systems

Simone Filice
filice.simone@gmail.com

University of Roma Tor Vergata

Web Mining e Retrieval 2013/2014
Motivations

- Is a ML system performing properly?
- Among a set of different algorithms/models, which one is performing better on a given task?
- What can I do to improve my system?
Overview

- Performance Evaluation Metrics
  - Classifier Evaluation Metrics
  - Information Retrieval Systems Evaluation Metrics

- Tuning and Evaluation Methods

- Error Diagnostics
Overview

- **Performance Evaluation Metrics**
  - Classifier Evaluation Metrics
  - Information Retrieval Systems Evaluation Metrics

- Tuning and Evaluation Methods

- Error Diagnostics
### Classifier Evaluation: Confusion Matrix

<table>
<thead>
<tr>
<th>ACTUAL VALUE</th>
<th>PREDICTED VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class A</td>
<td>Class A</td>
</tr>
<tr>
<td>Class B</td>
<td>Class A</td>
</tr>
<tr>
<td>Class C</td>
<td>Class A</td>
</tr>
</tbody>
</table>

#### Accuracy

\[
\text{accuracy} = \frac{\text{#correct classifications}}{\text{#classifications}} = \frac{38 + 43 + 44}{150} = 83.33\%
\]

#### Error Rate

\[
\text{error rate} = \frac{\text{#incorrect classifications}}{\text{#classifications}} = \frac{12 + 5 + 2 + 6}{150} = 16.67\%
\]
Evaluation with skewed data

- Accuracy is not a suitable metric for task with imbalanced classes (for instance a spam detector)

<table>
<thead>
<tr>
<th>ACTUAL VALUE</th>
<th>PREDICTED VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spam</td>
</tr>
<tr>
<td>Spam</td>
<td>0</td>
</tr>
<tr>
<td>Non-Spam</td>
<td>0</td>
</tr>
</tbody>
</table>

\[
\text{accuracy} = \frac{\text{#correct classifications}}{\text{#classifications}} = \frac{9990}{10000} = 99.9\%
\]
## Single Class Metrics

<table>
<thead>
<tr>
<th>ACTUAL VALUE</th>
<th>PREDICTED VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class C</td>
<td>TP</td>
</tr>
<tr>
<td></td>
<td>True Positive</td>
</tr>
<tr>
<td>Not Class C</td>
<td>FP</td>
</tr>
<tr>
<td></td>
<td>False Positive</td>
</tr>
</tbody>
</table>

**Precision**

\[
\text{precision} = \frac{TP}{TP + FP}
\]

**Recall**

\[
\text{recall} = \frac{TP}{TP + FN}
\]

**F1**

\[
F1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}
\]

what percentage of instances the classifier labeled as positive are actually positive?

what percentage of positive instances did the classifier label as positive?

F-measure is the harmonic mean of precision and recall
Overview

- Performance Evaluation Metrics
  - Classifier Evaluation Metrics
  - Information Retrieval System Evaluation Metrics

- Tuning and Evaluation Methods

- Error Diagnostics
Challenging in Evaluating IR Models

- The output provided by an Information Retrieval System is not simply correct or wrong.

- Ideally, we need to estimate user happiness.

- Happiness is elusive to measure.
  - Most common proxy: relevance of search results.
Challenging in Evaluating IR Models

- Effectiveness depends on the relevance of retrieved documents.
- Relevance is hard to model. It should be a continuous function and not a binary value.
- Relevance is:
  - Subjective: depends on the user’s point of view
  - Contextual: depends on the current user’s needs
  - Cognitive: is perceived and experienced by the user
  - Dynamic: changes over the time
Challenging in Evaluating IR Models

- A search engine is effective if it is able to provide documents that addresses user **information need**

- The **information need** is translated into a **query**
- Relevance is assessed relative to the **information need** not the **query**
- E.g., **Information need**: I’m looking for information on whether drinking red wine is more effective at reducing your risk of heart attacks than white wine.
- **Query**: **wine red white heart attack effective**
- Evaluate whether the doc addresses the information need, not whether it has these words
Evaluating IR Systems

- Tests directly involving users are the most reliable way to evaluate an IR system
  - A/B testing
  - Surveys...

- Offline tests are necessary to minimize the cost of the evaluation. Human Labeled Corpora (Gold Standard):
  - A benchmark document collection
  - A benchmark suite of queries
  - A usually binary assessment of either Relevant or Nonrelevant for each query and each document
Evaluating IR Systems

\[
\text{recall} = \frac{\text{Number of relevant documents retrieved}}{\text{Total number of relevant documents}}
\]

\[
\text{precision} = \frac{\text{Number of relevant documents retrieved}}{\text{Total number of documents retrieved}}
\]

What about accuracy???
Trade-off between Precision and Recall

- You can get high recall (but low precision) by retrieving all docs for all queries!
- Recall is a non-decreasing function of the number of docs retrieved
- In a good system, precision decreases as either the number of docs retrieved or recall increases
  - This is not a theorem, but a result with strong empirical confirmation
Trade-off between Precision and Recall

- **Precision**: Returns relevant documents but misses many useful ones too.
- **Recall**: Returns most relevant documents but includes lots of junk.
- **The ideal**: Balances between Precision and Recall.
Evaluating ranked results

- IR systems usually outputs the retrieved documents in a ranked list
  - A proper evaluating should mainly consider elements in the top of the list

```
[ ] [ ] [ ] [ ] [ ] [ ] [ ] = the relevant documents
```

```
Ranking #1
[ ] [ ] [ ] [ ] [ ] [ ] [ ]

Ranking #2
[ ] [ ] [ ] [ ] [ ] [ ] [ ]
```
Compute a recall/precision pair for each position in the ranked list that contains a relevant document.

<table>
<thead>
<tr>
<th>n</th>
<th>doc #</th>
<th>relevant</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>588</td>
<td>x</td>
</tr>
<tr>
<td>2</td>
<td>589</td>
<td>x</td>
</tr>
<tr>
<td>3</td>
<td>576</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>590</td>
<td>x</td>
</tr>
<tr>
<td>5</td>
<td>986</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>592</td>
<td>x</td>
</tr>
<tr>
<td>7</td>
<td>984</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>988</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>578</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>985</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>103</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>591</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>772</td>
<td>x</td>
</tr>
<tr>
<td>14</td>
<td>990</td>
<td></td>
</tr>
</tbody>
</table>

Total number of relevant docs = 6

Check each new recall point:

- \( R=1/6=0.167; \quad P=1/1=1 \)
- \( R=2/6=0.333; \quad P=2/2=1 \)
- \( R=3/6=0.5; \quad P=3/4=0.75 \)
- \( R=4/6=0.667; \quad P=4/6=0.667 \)
- \( R=5/6=0.833; \quad P=5/13=0.38 \)

Missing one relevant document

Never reach 100% recall
Averaging over Queries

- A precision-recall graph for one query isn’t a very sensible thing to look at
  - You need to average performance over a whole bunch of queries

- Some standard recall levels \( r_j \) are set. Typically:
  \[
  r_0 = 0.0, \quad r_1 = 0.1, \ldots, \quad r_{10} = 1.0 \quad (11\text{-point interpolated average precision})
  \]

- For each query the precision corresponding to each standard recall levels are estimated via interpolation:
  \[
P_{\text{interp}}(r_j) = \max_{r \geq r_j} P(r)
  \]

- Plot average precision/recall curves to evaluate overall system performance on a document/query corpus.
Interpolating a Recall/Precision Curve
Compare two or more Systems

- The curve closest to the upper right-hand corner of the graph indicates the best performance.

Graphs are good, but people want a summary measure....
Ranking metrics

- **Precision at fixed retrieval level**
  - Precision-at-\(k\) (P@\(k\)): Precision of top \(k\) results
  - Perhaps appropriate for most of web search: all people want are good matches on the first one or two result pages

- **Mean Average Precision (MAP)**

\[
MAP(Q) = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{|R_q|} \sum_{d \in R_q} P @ k_{q,d}
\]

- \(Q = \text{set of queries}\)
- \(R_q = \text{set of relevant documents for the query } q\)
- \(K_{q,d} = \text{ranking of the document } d \text{ retrieved through the query } q\)
Mean Average Precision

= relevant documents for query 1

<table>
<thead>
<tr>
<th>Ranking #1</th>
<th>Recall</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.2</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>0.4</td>
<td>0.67</td>
</tr>
<tr>
<td></td>
<td>0.4</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>0.4</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>0.63</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>0.38</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>0.44</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>0.5</td>
</tr>
</tbody>
</table>

= relevant documents for query 2

<table>
<thead>
<tr>
<th>Ranking #2</th>
<th>Recall</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>0.33</td>
<td>0.33</td>
</tr>
<tr>
<td></td>
<td>0.33</td>
<td>0.33</td>
</tr>
<tr>
<td></td>
<td>0.67</td>
<td>0.67</td>
</tr>
<tr>
<td></td>
<td>0.67</td>
<td>0.67</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

average precision query 1 = \(\frac{1.0 + 0.67 + 0.5 + 0.44 + 0.5}{5} = 0.62\)

average precision query 2 = \(\frac{0.5 + 0.4 + 0.43}{3} = 0.44\)

mean average precision = \(\frac{0.62 + 0.44}{2} = 0.53\)
Overview

- Performance Evaluation Metrics
  - Classifier Evaluation Metrics
  - Information Retrieval Systems Evaluation Metrics

- Tuning and Evaluation Methods

- Error Diagnostics
To obtain a reliable estimation, test data must be instances not used during the training step.

- Error on the training data is *not* a good indicator of performance on future data, because new data will probably not be *exactly* the same as the training data!
- Overfitting – fitting the training data too precisely usually leads to poor results on new data.
- We want to evaluate how predictive the model we learned is, and not its memorization capability.
Step 1: dataset splitting

For instance, 70% in the training set and 30% in the test set.
Step 2: learning phase

Data

Results Known

Training set

Learning algorithm

Testing set
Step 3: testing the model

Results Known

Data

<table>
<thead>
<tr>
<th>+</th>
<th>+</th>
<th>+</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

Training set

Learning algorithm

Testing set

Evaluation: comparison with the oracle
Evaluation on Few Data

- When data is scarce (totally or for a single class), a single evaluation process could not be enough representative
  - The testing set could contain too few instances to produce a reliable result

- The evaluation process must be repeated with different splitting
N-Fold Cross Validation

- Data is split into $n$ subsets of equal size
- Each subset in turn is used for testing and the remainders $n-1$ for training
- The metrics estimated in each round are averaged

5 fold splitting

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Testing fold

<table>
<thead>
<tr>
<th>Round 1</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

... Testing fold

<table>
<thead>
<tr>
<th>Round 5</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Tuning a Classifier

- Most of ML algorithms depend on some parameters (example $k$ in KNN)
- The best configuration must be chosen after a proper tuning stage:
  - A set of configurations must be established (for instance $k=1,2,5,10,15,20,30,50$)
  - Each configuration must be evaluated on a validation (or tuning) set
Complete ML Process

Data

Results Known

Training set

Evaluation Predictions

Learning Algorithm

Validation set

Learning with the best configuration

Test Set

Final Model

Final Evaluation
Overview

- Performance Evaluation Metrics
  - Classifier Evaluation Metrics
  - Information Retrieval Systems Evaluation Metrics

- Tuning and Evaluation Methods

- Error Diagnostics
Error Diagnostics

- Error Diagnostics helps in identifying what problem is affecting an ML system that performs poorly.
- Understanding the problem is useful in coming up with promising solutions for improving the system.

- Two opposite issues:
  - **Bias Problem**
  - **Variance Problem**
Bias Versus Variance

- Example in Regression

**BIAS PROBLEM:**
Learned function with too simple model

**VARIANCE PROBLEM:**
Learned function with too complex model

- Function to be learned
- Learned function
- Example
Diagnosing Bias vs Variance

- **Bias**
  - *Underfitting*: the model is not enough expressive to fit the complexity of the underlying concept to be learned
  - A high error is observed both in training and testing

- **Variance**
  - *Overfitting*: the model perfectly fits training data but is too complex (example: an extremely deep decision tree) and does not generalize well on new data
  - A high difference between the training error and the testing error
Diagnosing High Bias via Learning Curve

Example in regression: we want to fit a 2D data distribution with a straight line

\[ h_\theta(x) = \theta_0 + \theta_1 x_1 \]

After a certain value of \( m \), the learning process saturates and the testing error becomes similar to the training error \( \Rightarrow \) getting more example will not help too much
Diagnosing High Variance via Learning Curve

Example in regression: we want to fit a 2D data distribution with 10-th degree polynomial function

\[ h_\theta(x) = \theta_0 + \theta_1 x_1 + \ldots + \theta_{10} x_1^{10} \]

A large gap between the training error and the testing error is observed. The saturation point is still not reached \( \rightarrow \) new examples should help
Solutions for Bias and Variance

- **Bias**
  - Add new informative features
  - Use a more sophisticated algorithm (or the same algorithm with a more complex parameterization)

- **Variance**
  - Add new examples
  - Remove irrelevant and noisy features
  - Use a less complicated parameterization (example simpler polynomial function in regression)
Summary

- The effectiveness of ML or IR systems can be assessed with different evaluation metrics
  - we saw just the most popular, but a lot of other metrics exist!!!

- A reliable evaluation should follow some guideline

- An error diagnostics is useful for understanding how improving the system performance