Methods for Intelligent Systems
Lecture Notes on Machine Learning

Matteo Matteucci
matteucci@elet.polimi.it

Department of Electronics and Information
Politecnico di Milano

Evaluation and Credibility
- Evaluation of Classifiers-
**Credibility Issues**

Once we got a model extracted from data we should ask ourselves:

- How much should we believe in what was learned?
- How predictive is the model we learned?

However errors on the training data is *not* a good indicator of performance on future data, why?

- Since the classifier has been learned from the very same training data, any estimate based on that data will be **optimistic**.
- You can find patterns even in random data
- In addition, 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 need proper evaluation techniques to predict model performances on these new data.

---

**Evaluation on "LARGE" Datasets**

If many (thousands) of examples are available, including several hundred examples from each class, then a simple evaluation is sufficient:

- Randomly split data into training and test sets (usually 2/3 for train, 1/3 for test)
- Build a classifier using the *train* set and evaluate it on the *test* set.
Parameter Tuning with Cross-Validation (I)

It is important that the test data is not used in any way to create the classifier, some learning schemes operate in two stages:

- Stage 1: builds the basic structure
- Stage 2: optimizes parameter settings

The test data cannot be used for parameter tuning!

Proper procedure uses three sets:

- *training data*: to learn the model
- *validation data*: to optimize model parameters
- *test data*: to give a proper unbiased model evaluation

Generally, the larger the training data the better the classifier (but returns diminish) and the more accurate the error estimate

Parameter Tuning with Cross-Validation (II)

![Diagram of data handling and model evaluation process]

- **Data** flow:
  - Data
  - Training set
  - Validation set
  - Testing set
  - Predictions
  - Final Evaluation

- **Model Builder**:
  - Builds the basic structure
  - Optimizes parameter settings

The diagram illustrates the process of data handling and model evaluation, emphasizing the separation of training, validation, and testing datasets and the iterative process of building and evaluating models.
Dealing with Unbalanced Data

Sometimes, classes have very unequal frequency
- Attrition prediction: 97% stay, 3% attrite (in a month)
- Medical diagnosis: 90% healthy, 10% disease
- eCommerce: 99% don’t buy, 1% buy
- Security: >99.99% of Americans are not terrorists

Majority class classifier can be 97% correct, but useless

With two classes, a good approach is to build balanced train and test sets, and train model on a balanced set
- Randomly select desired number of minority class instances
- Add equal number of randomly selected majority class
- With multiple classes ensure that each class is represented with approximately equal proportions in train and test.

Are We Confident on Evaluation?

Assume the estimated error rate is 25%. How close is this to the true error rate for our classifier?

*Depends on the amount of test data!* We assume that it is 25% but we would be more confident if the estimate was done based on 10000 instances than on 100 instances

Prediction is just like tossing a *biased* coin “Head” is a “success”, “tail” is an “error”. In statistics this is called a *Bernoulli process* and we can compute confidence intervals for the true underlying proportion!

- Mean and variance for a Bernoulli trial: $p, p(1−p)$
- Expected success rate $f=S/N$ is a random variable with mean $p$ and $p(1−p)/N$ For large $N$, $f$ approaches a Normal distribution
- $c\%$ confidence interval $[-z \leq X \leq z]$ for random variable with 0 mean is given by $Pr[-z \leq X \leq z] = c$
- With a symmetric distribution, $Pr[-z \leq X \leq z] = 1-2xPr[X\geq z]$
Computing Confidence Intervals

Transformed value for f: \( \left( f - p \right) \sqrt{p(1-p) / N} \)

Resulting equation:
\[
\Pr\left[ -z \leq \frac{f - p}{\sqrt{p(1-p) / N}} \leq z \right] = c
\]

Solving for p:
\[
p = \left( f + \frac{z^2}{2N} \pm z \sqrt{\frac{f^2}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}} \right) \left( 1 + \frac{z^2}{N} \right)
\]

Confidence limits for normal distribution with mean 0 and variance 1:

<table>
<thead>
<tr>
<th>Pr[X ≥ z]</th>
<th>0.1%</th>
<th>0.5%</th>
<th>1%</th>
<th>5%</th>
<th>10%</th>
<th>20%</th>
<th>40%</th>
</tr>
</thead>
<tbody>
<tr>
<td>z</td>
<td>3.09</td>
<td>2.58</td>
<td>2.33</td>
<td>1.65</td>
<td>1.28</td>
<td>0.84</td>
<td>0.25</td>
</tr>
</tbody>
</table>

Thus, \( Pr[-1.65 \leq X \leq 1.65] = 90\% \)

How Many Data Should I Have?

Suppose our classifier has \( f = 75\% \), and we tested it on \( N = 1000 \). What is the \( c = 80\% \) confidence interval?
- \( c = 80\% \rightarrow z = 1.28 \)
- \( p \in [0.732, 0.767] \)

Suppose now our classifier has \( f = 75\% \), but we tested it on \( N = 100 \). What is now the \( c = 80\% \) confidence interval?
- \( c = 80\% \rightarrow z = 1.28 \)
- \( p \in [0.691, 0.801] \)

The normal distribution assumption is only valid for large \( N \) (i.e. \( N > 100 \)). When facing \( f = 75\% \), \( N = 10 \), our \( c = 80\% \) confidence interval \( p \in [0.549, 0.881] \) should be taken with a grain of salt ;-)

<table>
<thead>
<tr>
<th>Pr[X ≥ z]</th>
<th>0.1%</th>
<th>0.5%</th>
<th>1%</th>
<th>5%</th>
<th>10%</th>
<th>20%</th>
<th>40%</th>
</tr>
</thead>
<tbody>
<tr>
<td>z</td>
<td>3.09</td>
<td>2.58</td>
<td>2.33</td>
<td>1.65</td>
<td>1.28</td>
<td>0.84</td>
<td>0.25</td>
</tr>
</tbody>
</table>
Evaluation On “Small” Data

The holdout method reserves a certain amount of data for testing and uses the remainder for training, but for small or “unbalanced” datasets, samples might not be representative (i.e., few or none instances)

Stratified sample
- Advanced version of balancing the data
- Make sure that each class is represented with approximately equal proportions in both subsets

Holdout estimate can be made more reliable by repeating the process with different sub-samples
- At each iteration, a certain proportion is randomly selected for training (possibly with stratification)
- The error rates on the different iterations are averaged to yield an overall error rate

This is called the repeated holdout method, but it is still not optimum since the different test sets overlap

k-fold Cross-Validation

k-fold cross-validation is used to avoid overlapping test sets
- Data is split into k subsets of equal size
- Each subset in turn is used for testing and remainders for training
- Error estimates are averaged to yield an overall error estimate (often the subsets are stratified before cross-validation)

Standard method for evaluation stratified 10-fold cross-validation,
- Extensive experiments have shown that $k = 10$ is the best choice to get an accurate estimate
- Stratification reduces the estimate’s variance
- Even better repeated stratified cross-validation (i.e., ten-fold cross-validation is repeated ten times and results are averaged)
- Other approaches appear to be robust, e.g., 5x2 cross-validation
Leave-One-Out Cross-Validation

It is a particular form of k-fold Cross-Validation
- Set number of folds equal to the number of training instances (i.e., for \( n \) training instances, build classifier \( n \) times)
- Makes best use of the data
- Involves no random sub-sampling

Leave-One-Out has some disadvantages too:
- Very computationally expensive
- It guarantees a non-stratified sample because there is only one instance in the test set!

**Extreme example:** random dataset split equally into two classes
- Best inducer predicts majority class
- 50% accuracy on fresh data
- Leave-One-Out-CV estimate is 100% error!

The Bootstrap: Sampling with Replacement

Cross-Validation uses sampling without replacement while Bootstrap uses sampling with replacement to form the training set
- Sample a dataset of \( n \) instances \( n \) times with replacement to form a dataset of \( n \) instances
- Use this data as the training set
- Use the instances from the original dataset that don’t occur in the new training set for testing

Also called the 0.632 bootstrap
- An instance has a probability of \( 1 - 1/n \) of not being picked

- Its probability of ending up in the test data is: \( \left(1 - \frac{1}{n}\right)^n \approx e^{-1} = 0.368 \)
- This means the training data will contain approximately 63.2% of the instances
Estimating Error With Bootstrap

The error estimate on the test data will be very pessimistic, since training was on just ~63% of the instances; therefore, combine it with the resubstitution error:

\[ err = 0.632 \cdot e_{\text{test instances}} + 0.368 \cdot e_{\text{training instances}} \]

Repeat process several times with different replacement samples; average the results

Probably the best way of estimating performance for very small datasets; however, it has some problems

- Consider a random dataset with a 50% class distribution
- A perfect memorizer will achieve 0% resubstitution error and ~50% error on test data
- Bootstrap estimate for this classifier:
  \[ err = 0.632 \cdot 50\% + 0.368 \cdot 0\% = 31.6\% \]
- True expected error: 50%

Evalutation and Credibility
- Results Comparison-
Comparing Models with Significance Test

We want to know which one of two learning schemes performs better
- Domain dependent!
- Use 10-fold CV estimates, but we have variance on it

Significance tests tell us how confident we can be that there really is a difference
- Null hypothesis: there is no “real” difference
- Alternative hypothesis: there is a difference
- A significance test measures how much evidence there is in favor of rejecting the null hypothesis

Let’s say we are using 10 times 10-fold CV
- Then we want to know whether the two means of the 10 CV estimates are significantly different
- Student’s paired t-test tells us whether the means of two samples are significantly different

Student’s Paired t-test

Student’s t-test tells whether the means of two samples are different
- Take individual samples from all possible CV estimates
- Use a paired t-test because the individual samples are paired
- The same CV is applied twice

Being $x_1, x_2, ..., x_k$ and $y_1, y_2, ..., y_k$ the $2k$ samples for a $k$-fold CV for two models $X$ and $Y$, $m_x$ and $m_y$ are the means
- With enough samples, the mean of a set of independent samples is normally distributed $m - \mu / \sqrt{\sigma^2 / k} \approx N(0,1)$
- Estimated variances of the means are $\sigma_x^2 / k$ and $\sigma_y^2 / k$
- If $\mu_x$ and $\mu_y$ are the true means then
  \[
  \frac{m_x - \mu_x}{\sqrt{\sigma_x^2 / k}} \quad \frac{m_y - \mu_y}{\sqrt{\sigma_y^2 / k}}
  \]
  are approximately normally distributed with mean 0, variance 1
Student’s Distribution

With small samples ($k < 100$) the mean follows Student’s distribution with $k-1$ degrees of freedom.

Confidence limits:

<table>
<thead>
<tr>
<th>Pr[$X \geq z$]</th>
<th>$z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1%</td>
<td>4.30</td>
</tr>
<tr>
<td>0.5%</td>
<td>3.25</td>
</tr>
<tr>
<td>1%</td>
<td>2.82</td>
</tr>
<tr>
<td>5%</td>
<td>1.83</td>
</tr>
<tr>
<td>10%</td>
<td>1.38</td>
</tr>
<tr>
<td>20%</td>
<td>0.88</td>
</tr>
</tbody>
</table>

9 degrees of freedom

<table>
<thead>
<tr>
<th>Pr[$X \geq z$]</th>
<th>$z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1%</td>
<td>3.09</td>
</tr>
<tr>
<td>0.5%</td>
<td>2.58</td>
</tr>
<tr>
<td>1%</td>
<td>2.33</td>
</tr>
<tr>
<td>5%</td>
<td>1.65</td>
</tr>
<tr>
<td>10%</td>
<td>1.28</td>
</tr>
<tr>
<td>20%</td>
<td>0.84</td>
</tr>
</tbody>
</table>

normal distribution

William Sealy Gosset (June 13, 1876 – October 16, 1937) was a chemist and statistician, better known by his pen name Student. Born Canterbury, England to Agnes Sealy Vidal and Colonel Frederic Gosset, Gosset attended Winchester College, the famous private school, before reading chemistry and mathematics at New College, Oxford. On graduating in 1899, he joined the Dublin brewery of Arthur Guinness & Son.

Distribution of the Differences

Let $m_d = m_x - m_y$ the difference of the means ($m_d$), it also has a Student’s distribution with $k-1$ degrees of freedom.

Let $\sigma_d^2$ be the variance of the difference; the standardized version of $m_d$ is called the $t$-statistic:

$$t = \frac{m_d}{\sqrt{\frac{\sigma_d^2}{k}}}$$

We use $t$ to perform the $t$-test:

- Fix a significance level $\alpha$
- If a difference is significant at the $\alpha\%$ level, there is a $(100-\alpha)\%$ chance that there really is a difference
- Divide the significance level by 2 since the test is two-tailed
- Look up the value for $z$ that corresponds to $\alpha/2$
- If $t \leq -z$ or $t \geq z$ then the difference is significant (i.e., the null hypothesis can be rejected)
Unpaired Observations

If the CV estimates are from different randomizations, they are no longer paired (or maybe we used $k$-fold CV for one scheme, and $j$-fold CV for the other one)

- Then we have to use an unpaired t-test with $\min(k, j) - 1$ degrees of freedom
- The t-statistic becomes:

$$ t = \frac{m_d}{\sqrt{\sigma^2_d / k}} $$

All our cross-validation estimates are based on the same dataset hence the test only tells us whether a complete $k$-fold CV for this dataset would show a difference

Ideally, should use a different dataset sample for each of the $k$-fold CV estimates used in the test to judge performance across different training sets

Predicting Probabilities

Performance measure so far the success rate: **0-1 loss function**

$$ \sum_i \begin{cases} 0 & \text{if prediction is correct} \\ 1 & \text{if prediction is wrong} \end{cases} $$

Most classifiers produces class probabilities, depending on the application, we might want to check the accuracy of these estimates; we can use two different loss functions:

- Quadratic loss function: it takes into account all class probability estimates for an instance (it is bounded by $1 + \sum p_j^2$)

$$ \sum_j (p_j - a_j)^2 = \sum_{j \neq c} p_j^2 + (1 - p_c)^2 $$

- Informational loss function: it focuses only on the probability estimate for the actual class (it can be infinite)

$$ - p_1^* \log_2 p_1 - \ldots - p_k^* \log_2 p_k $$
Alternative Loss Functions

**Quadratic Loss Function:**
- \( p_1 \ldots p_k \) are probability estimates given \( k \) possible outcomes
- \( c \) is the index of instance actual class \( a_c = 1 \) and \( a_1 \ldots a_k = 0 \)
- Quadratic loss is:
  \[
  \sum_j (p_j - a_j)^2 = \sum_j p_j^2 + (1 - p_c)^2
  \]
- Want to minimize:
  \[
  E \left[ \sum_j (p_j - a_j)^2 \right]
  \]
- Let \( p_1^* \ldots p_k^* \) be the true class probabilities, it can be shown that this is minimized when \( p_j = p_j^* \), the true probabilities.

**Informational Loss Function:**
- \(- \log(p_c)\), where \( c \) is the index of the instance’s actual class
- Number of bits required to communicate the actual class
- Let \( p_1^* \ldots p_k^* \) be the true class probabilities then the expected value for the loss function is:
  \[
  -p_1^* \log p_1 - \ldots - p_k^* \log p_k
  \]
- Justification: minimized when \( p_j = p_j^* \)

Lift Charts

*Lift* is a measure of the effectiveness of a predictive model calculated as the ratio between the results obtained with/without the model.
- Cumulative gains and lift charts are visual aids for measuring model performance
- Both charts consist of a lift curve and a baseline
- The greater the area between the lift curve and the baseline, the better the model

Given a scheme that outputs probability, sort the instances in descending order according to predicted probability and select instance subset starting from the one with the highest predicted probability

<table>
<thead>
<tr>
<th>Rank</th>
<th>Predicted Probability</th>
<th>Actual Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.95</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>0.93</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>0.93</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>0.88</td>
<td>Yes</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

In lift chart, x axis is sample size and y axis is number of true positives
Example: Direct Marketing

In mass mail out of promotional offers (1000000) the proportion who normally respond is 0.1% (i.e., 1000 responses). A data mining tool can identify a subset of a 100000 for which the response rate is 0.4%, that is 400 responses.

- In marketing terminology, the increase of response rate is known as the lift factor yielded by the model.
- The same data mining tool, may be able to identify 400000 households for which the response rate is 0.2%, that is 800 respondents corresponding to a lift factor of 2.

The overall goal is to find subsets of test instances that have a high proportion of true positive.

ROC Curves

ROC (Receiver Operating Characteristic) curves are similar to lift charts:
- y axis shows percentage of true positives in sample, rather than absolute number
- x axis shows percentage of false positives in sample, rather than sample size

Simple method of getting a ROC curve using cross-validation:
- Collect instances probabilities in test folds
- Sort instances according to probabilities

However, this is just one possibility
- You can generate an ROC curve for each fold and averages them
**ROC Curves for Two Schemas**

Given 2 learning schemes we can achieve any point on the convex hull!

- TP and FP rates for scheme 1: $t_1$ and $f_1$
- TP and FP rates for scheme 2: $t_2$ and $f_2$

If scheme 1 is used to predict $100\times q$ % of the cases and scheme 2 for the rest, then

- TP rate for combined: $q \times t_1 + (1-q) \times t_2$
- FP rate for combined: $q \times f_2 + (1-q) \times f_2$

**Typical Measures in Information Retrieval**

Other measures are typically used in Information Retrieval:

- Percentage of retrieved documents that are relevant: 
  - Precision = $\frac{TP}{TP+FP}$
- Percentage of relevant documents that are returned: 
  - Recall = $\frac{TP}{TP+FN}$
- Precision/Recall Curves (they have hyperbolic shape)
- Average precision at 20%, 50% and 80% recall (three-point average recall)
- $F$-measure = $\frac{2 \times \text{recall} \times \text{precision}}{\text{recall} + \text{precision}}$

Usually used in different domains:

<table>
<thead>
<tr>
<th>Lift chart</th>
<th>Domain</th>
<th>Plot</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Marketing</td>
<td>TP Subset size</td>
<td>$\frac{TP}{(TP+FP+TN+FN)}$</td>
</tr>
<tr>
<td>ROC curve</td>
<td>Communications</td>
<td>TP rate</td>
<td>$\frac{TP}{(TP+FN)}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FP rate</td>
<td>$\frac{FP}{(FP+TN)}$</td>
</tr>
<tr>
<td>Recall-precision curve</td>
<td>Information retrieval</td>
<td>Recall Precision</td>
<td>$\frac{TP}{(TP+FN)}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$\frac{TP}{(TP+FP)}$</td>
</tr>
</tbody>
</table>
Counting The Cost

So far we evaluated classification results without taking into account the cost of making wrong classifications, but in many applications wrong some misclassification errors may be more costly than others:

- Medical diagnostic tests: does $X$ have leukaemia?
- Loan decisions: approve mortgage for $X$?
- Web mining: will $X$ click on this link?
- Promotional mailing: will $X$ buy the product?

If we consider the 2 class scenario:
- Two-class case (yes, no)
- Four different outcomes:
  o true positive
  o true negative
  o false positive
  o false negative
- Usually displayed in a confusion matrix

<table>
<thead>
<tr>
<th>Actual class</th>
<th>Predicted class</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
<td>TP: True positive</td>
</tr>
<tr>
<td>No</td>
<td>FP: False positive</td>
</tr>
</tbody>
</table>

Cost-sensitive Learning

Most learning schemes do not perform cost-sensitive learning
- They generate the same classifier no matter what costs are assigned to the different classes
- Example: standard decision tree learner

Simple methods for cost-sensitive learning:
- Re-sampling of instances according to costs
- Weighting of instances according to costs

Some schemes are inherently cost-sensitive, e.g. Naïve Bayes Classifiers, if the cost are known then current classification scheme can be adapted to consider such costs when building the model

*But we won’t go in the details ...*
Evaluating Numeric Prediction

Same strategies: independent test set, cross-validation, significance tests, etc.

- **The mean-squared error:**  \( (p_1 - a_1)^2 + ... + (p_n - a_n)^2 / n \)
- **The root mean-squared error:**  \( \sqrt{\frac{(p_1 - a_1)^2 + ... + (p_n - a_n)^2}{n}} \)
- **The mean absolute error:**
  
  (less sensitive to outliers)
  \[ \frac{|p_1 - a_1| + ... + |p_n - a_n|}{n} \]
- **The relative squared error:**
  \[ \frac{(p_1 - a_1)^2 + ... + (p_n - a_n)^2}{(\bar{a} - \bar{a})^2 + ... + (\bar{a} - \bar{a})^2} \]
- **The relative absolute error:**
  \[ \frac{|p_1 - a_1| + ... + |p_n - a_n|}{|\bar{a} - \bar{a}| + ... + |\bar{a} - \bar{a}|} \]

- Statistical correlation between predicted values and actual ones:
  - Between –1 and +1
  - Good performance leads to large values!

*No single best to look at, better to consider all of them!*

---

Evaluation and Credibility

- Model Selection
Model Selection Criteria

Model selection criteria attempt to find a good compromise between:
- The complexity of a model
- Its prediction accuracy on the training data

Reasoning: a good model is a simple model that achieves high accuracy on the given data

Also known as Occam’s Razor: “The best theory is the smallest one that describes all the facts”

Entia non sunt multiplicanda praeter necessitatem.
William of Ockham (c.1285–1349)

Things should be made as simple as possible, but not any simpler.
Albert Einstein (1879-1955)

The MDL Principle

MDL stands for Minimum Description Length defined as:

\[
\text{space required to describe a theory} + \\
\text{space required to describe the theory’s mistakes}
\]

In our case the theory is the classifier and the mistakes are the errors on the training data
- We seek a classifier with minimal DL
- MDL principle is a model selection criterion

Example: Theory 1 is simple, elegant, and explains data almost perfectly; Theory 2 is more complex, reproduces the data without mistakes ... Theory 1 is probably preferable!

Kepler’s three laws on planetary motion less accurate than Copernicus’s latest refinement of the Ptolemaic theory of epicycles
**MDL and Compression**

"The best theory is the one that compresses data the most, i.e., we generate a model, store it, and its mistakes"

We need to compute:
- size of the model, and space needed to encode the errors
- easy: use the informational loss function
- need a method to encode the model

Bayes’ theorem gives a posteriori probability of theory given the data:

\[
L[T] = \text{“length” of the theory}
\]
\[
L[E|T] = \text{training set encoded wrt the theory}
\]
\[
L[T] + L[E|T] = \text{Description length}
\]

\[
\Pr[T | E] = \frac{\Pr[E | T] \Pr[T]}{\Pr[E]}
\]

Equivalent to:

\[
- \log \Pr[T | E] = - \log \Pr[E | T] - \log \Pr[T] + \log \Pr[E]
\]

**MDL and MAP**

MAP stands for *maximum a posteriori probability*; finding the MAP theory corresponds to finding the MDL theory:

- Difficult bit in applying the MAP principle determining the prior probability \( \Pr[T] \) of the theory
- Corresponds to difficult part in applying the MDL principle: coding scheme for the theory

I.e., if we know a priori that a particular theory is more likely we need less bits to encode it

- Advantage: full use of training data when selecting a model
- Disadvantage: appropriate coding scheme/prior probabilities for theories are crucial
- Disadvantage 2: no guarantee that the MDL theory is the one which minimizes the expected error

*Reflects Epicurus’ principle: all theories are used for prediction weighted according to \( P[T|E] \)*