Evaluating Machine-Learning Methods

Mark Craven
Computer Sciences 760
Fall 2015

www.biostat.wisc.edu/~craven/cs760/

Some of the slides in these lectures have been adapted/borrowed from materials developed by Tom Dietterich, Pedro Domingos, Tom Mitchell, David Page, and Jude Shavlik

Goals for the lecture

you should understand the following concepts

• test sets
• learning curves
• stratified sampling
• cross validation
• model selection
• validation (tuning) sets
• internal cross validation
• confusion matrices
• TP, FP, TN, FN
• ROC curves
• confidence intervals for error
• pairwise t-tests for comparing learning systems
• scatter plots for comparing learning systems
• lesion studies
Test sets revisited

How can we get an unbiased estimate of the accuracy of a learned model?

- when learning a model, you should pretend that you don’t have the test data yet (it is “in the mail”)*

- if the test-set labels influence the learned model in any way, accuracy estimates will be biased

* In some applications it is reasonable to assume that you have access to the feature vector (i.e. $x$) but not the $y$ part of each test instance.
Learning curves

How does the accuracy of a learning method change as a function of the training-set size?

This can be assessed by plotting *learning curves*.

Figure from Perlich et al. *Journal of Machine Learning Research*, 2003

---

Learning curves

given training/test set partition

- for each sample size $s$ on learning curve
  - (optionally) repeat $n$ times
    - randomly select $s$ instances from training set
    - learn model
    - evaluate model on test set to determine accuracy $a$
  - plot $(s, a)$ or $(s, \text{avg. accuracy and error bars})$
Limitations of using a single training/test partition

• we may not have enough data to make sufficiently large training and test sets
  • a larger test set gives us more reliable estimate of accuracy (i.e. a lower variance estimate)
  • but… a larger training set will be more representative of how much data we actually have for learning process

• a single training set doesn’t tell us how sensitive accuracy is to a particular training sample

Using multiple training/test partitions

• two general approaches for doing this
  • random resampling
  • cross validation
Random resampling

We can address the second issue by repeatedly randomly partitioning the available data into training and set sets.

Stratified sampling

When randomly selecting training or validation sets, we may want to ensure that class proportions are maintained in each selected set.

This can be done via stratified sampling: first stratify instances by class, then randomly select instances from each class proportionally.
Cross validation example

Suppose we have 100 instances, and we want to estimate accuracy with cross validation

<table>
<thead>
<tr>
<th>iteration</th>
<th>train on</th>
<th>test on</th>
<th>correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>s_2 s_3 s_4 s_5</td>
<td>s_1</td>
<td>11 / 20</td>
</tr>
<tr>
<td>2</td>
<td>s_1 s_3 s_4 s_5</td>
<td>s_2</td>
<td>17 / 20</td>
</tr>
<tr>
<td>3</td>
<td>s_1 s_2 s_4 s_5</td>
<td>s_3</td>
<td>16 / 20</td>
</tr>
<tr>
<td>4</td>
<td>s_1 s_2 s_3 s_5</td>
<td>s_4</td>
<td>13 / 20</td>
</tr>
<tr>
<td>5</td>
<td>s_1 s_2 s_3 s_4</td>
<td>s_5</td>
<td>16 / 20</td>
</tr>
</tbody>
</table>

accuracy = 73/100 = 73%
**Cross validation**

- 10-fold cross validation is common, but smaller values of $n$ are often used when learning takes a lot of time

- In *leave-one-out* cross validation, $n = \#$ instances

- In *stratified* cross validation, stratified sampling is used when partitioning the data

- CV makes efficient use of the available data for testing

- Note that whenever we use multiple training sets, as in CV and random resampling, we are evaluating a learning method as opposed to an individual learned model

---

**Model selection**

- *Model selection* is the task of selecting a model from a set of candidate models
  - Selecting among decision trees with various levels of pruning
  - Selecting $k$ in $k$-NN
  - Etc.

- One approach to model selection is to use a tuning set or *internal* cross validation
Validation (tuning) sets revisited

Suppose we want estimates of accuracy during the learning process (e.g. to choose the best level of decision-tree pruning)?

- **Training set**
- **Test set**

**Learning process**

- **Training set**
- **Validation set**

**Learn models**

**Select model**

**Learned model**

Partition training data into separate training/validation sets

---

Internal cross validation

Instead of a single validation set, we can use cross-validation within a training set to select a model (e.g. to choose the best level of decision-tree pruning)?

- **Training set**
- **Test set**

**Learning process**

- $s_1$, $s_2$, $s_3$, $s_4$, $s_5$

**Learn models**

**Select model**

**Learned model**
Example: using internal cross validation to select $k$ in $k$-NN

given a training set
1. partition training set into $n$ folds, $s_1 \ldots s_n$
2. for each value of $k$ considered
   for $i = 1$ to $n$
      learn $k$-NN model using all folds but $s_i$
      evaluate accuracy on $s_i$
3. select $k$ that resulted in best accuracy for $s_1 \ldots s_n$
4. learn model using entire training set and selected $k$

the steps inside the box are run independently for each training set (i.e. if we’re using 10-fold CV to measure the overall accuracy of our $k$-NN approach, then the box would be executed 10 times)

Confusion matrices

How can we understand what types of mistakes a learned model makes?

![Confusion matrix](https://vision.jhu.edu)

**task:** activity recognition from video

**actual class**

**predicted class**

*figure from vision.jhu.edu*
Confusion matrix for 2-class problems

predicted class

\[
\begin{array}{cc}
\text{positive} & \text{negative} \\
\text{true positives (TP)} & \text{false positives (FP)} \\
\text{false negatives (FN)} & \text{true negatives (TN)}
\end{array}
\]

\[
\text{accuracy} = \frac{TP + TN}{TP + FP + FN + TN}
\]

\[
\text{error} = 1 - \text{accuracy} = \frac{FP + FN}{TP + FP + FN + TN}
\]

Is accuracy an adequate measure of predictive performance?

accuracy may not be useful measure in cases where

• there is a large class skew
  • Is 98% accuracy good when 97% of the instances are negative?

• there are differential misclassification costs – say, getting a positive wrong costs more than getting a negative wrong
  • Consider a medical domain in which a false positive results in an extraneous test but a false negative results in a failure to treat a disease

• we are most interested in a subset of high-confidence predictions
Other accuracy metrics

<table>
<thead>
<tr>
<th>predicted class</th>
<th>actual class</th>
</tr>
</thead>
<tbody>
<tr>
<td>positive</td>
<td>positive</td>
</tr>
<tr>
<td>true positives (TP)</td>
<td>false positives (FP)</td>
</tr>
<tr>
<td>false negatives (FN)</td>
<td>true negatives (TN)</td>
</tr>
<tr>
<td>negative</td>
<td>negative</td>
</tr>
</tbody>
</table>

true positive rate (recall) = \( \frac{TP}{\text{actual}} \text{ pos} \) = \( \frac{TP}{TP + FN} \)

false positive rate = \( \frac{FP}{\text{actual}} \text{ neg} \) = \( \frac{FP}{TN + FP} \)

ROC curves

A Receiver Operating Characteristic (ROC) curve plots the TP-rate vs. the FP-rate as a threshold on the confidence of an instance being positive is varied.

Different methods can work better in different parts of ROC space.
Algorithm for creating an ROC curve

let \( \left( (y^{(1)}, c^{(1)}), \ldots, (y^{(m)}, c^{(m)}) \right) \) be the test-set instances sorted according to predicted confidence

let each instance is positive

let \( num_{-}neg \), \( num_{-}pos \) be the number of negative/positive instances in the test set

\( TP = 0, FP = 0 \)

\( last_{-}TP = 0 \)

for \( i = 1 \) to \( m \)

// find thresholds where there is a pos instance on high side, neg instance on low side

if ( \( c(i) \neq c(i-1) \) ) and ( \( y(i) == \text{neg} \) ) and ( \( TP > last_{-}TP \) )

\( FPR = FP / num_{-}neg, \quad TPR = TP / num_{-}pos \)

output \( (FPR, TPR) \) coordinate

\( last_{-}TP = TP \)

if \( y(i) == \text{pos} \)

++TP

else

++FP

\( FPR = FP / num_{-}neg, \quad TPR = TP / num_{-}pos \)

output \( (FPR, TPR) \) coordinate

Plotting an ROC curve

<table>
<thead>
<tr>
<th>instance</th>
<th>confidence positive</th>
<th>correct class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ex 9</td>
<td>.99</td>
<td>+</td>
</tr>
<tr>
<td>Ex 7</td>
<td>.98</td>
<td>+</td>
</tr>
<tr>
<td>Ex 1</td>
<td>.72</td>
<td>-</td>
</tr>
<tr>
<td>Ex 2</td>
<td>.70</td>
<td>+</td>
</tr>
<tr>
<td>Ex 6</td>
<td>.65</td>
<td>+</td>
</tr>
<tr>
<td>Ex 10</td>
<td>.51</td>
<td>-</td>
</tr>
<tr>
<td>Ex 3</td>
<td>.39</td>
<td>-</td>
</tr>
<tr>
<td>Ex 5</td>
<td>.24</td>
<td>+</td>
</tr>
<tr>
<td>Ex 4</td>
<td>.11</td>
<td>-</td>
</tr>
<tr>
<td>Ex 8</td>
<td>.01</td>
<td>-</td>
</tr>
</tbody>
</table>
Plotting an ROC curve

can interpolate between points to get convex hull

- convex hull: perform all interpolations and discard any point that lies below a line
- interpolated points are achievable in theory: can flip weighted coin to choose between classifiers represented by plotted points

ROC curve example

task: recognizing genomic units called operons

figure from Bockhorst et al., *Bioinformatics* 2003
ROC curves and misclassification costs

The best operating point depends on the relative costs of FN and FP misclassifications.

![ROC curve diagram]

- best operating point when FN costs 10× FP
- best operating point when cost of misclassifying positives and negatives is equal
- best operating point when FP costs 10× FN

ROC curves

Does a low false-positive rate indicate that most positive predictions (i.e. predictions with confidence > some threshold) are correct?

suppose our TPR is 0.9, and FPR is 0.01

<table>
<thead>
<tr>
<th>fraction of instances that are positive</th>
<th>fraction of positive predictions that are correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0.989</td>
</tr>
<tr>
<td>0.1</td>
<td>0.909</td>
</tr>
<tr>
<td>0.01</td>
<td>0.476</td>
</tr>
<tr>
<td>0.001</td>
<td>0.083</td>
</tr>
</tbody>
</table>
Other accuracy metrics

<table>
<thead>
<tr>
<th>actual class</th>
<th>positive</th>
<th>negative</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>true positives (TP)</td>
<td>false positives (FP)</td>
</tr>
<tr>
<td></td>
<td>false negatives (FN)</td>
<td>true negatives (TN)</td>
</tr>
</tbody>
</table>

predicted class

positive

negative

$$\text{recall (TP rate)} = \frac{\text{TP}}{\text{actual pos}} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{precision (positive predictive value)} = \frac{\text{TP}}{\text{predicted pos}} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Precision/recall curves

A precision/recall curve plots the precision vs. recall (TP-rate) as a threshold on the confidence of an instance being positive is varied.
How do we get one ROC/PR curve when we do cross validation?

Approach 1
• make assumption that confidence values are comparable across folds
• pool predictions from all test sets
• plot the curve from the pooled predictions

Approach 2 (for ROC curves)
• plot individual curves for all test sets
• view each curve as a function
• plot the average curve for this set of functions
Comments on ROC and PR curves

both
• allow predictive performance to be assessed at various levels of confidence
• assume binary classification tasks
• sometimes summarized by calculating area under the curve

ROC curves
• insensitive to changes in class distribution (ROC curve does not change if the proportion of positive and negative instances in the test set are varied)
• can identify optimal classification thresholds for tasks with differential misclassification costs

precision/recall curves
• show the fraction of predictions that are false positives
• well suited for tasks with lots of negative instances

Confidence intervals on error

Given the observed error (accuracy) of a model over a limited sample of data, how well does this error characterize its accuracy over additional instances?

Suppose we have
• a learned model $h$
• a test set $S$ containing $n$ instances drawn independently of one another and independent of $h$
• $n \geq 30$
• $h$ makes $r$ errors over the $n$ instances

our best estimate of the error of $h$ is

$$\text{error}_S(h) = \frac{r}{n}$$
Confidence intervals on error

With approximately $C\%$ probability, the true error lies in the interval

$$error_s(h) \pm z_C \sqrt{\frac{error_s(h)(1 - error_s(h))}{n}}$$

where $z_C$ is a constant that depends on $C$ (e.g. for 95% confidence, $z_C = 1.96$)

How did we get this?

1. Our estimate of the error follows a binomial distribution given by $n$ and $p$ (the true error rate over the data distribution)

![Binomial distribution with n = 15 and p = 0.2](image)

2. Most common way to determine a binomial confidence interval is to use the normal approximation (although can calculate exact intervals if $n$ is not too large)
Confidence intervals on error

2. When $n \geq 30$, and $p$ is not too extreme, the normal distribution is a good approximation to the binomial.

3. We can determine the $C\%$ confidence interval by determining what bounds contain $C\%$ of the probability mass under the normal.

Comparing learning systems

How can we determine if one learning system provides better performance than another

- for a particular task?
- across a set of tasks / data sets?
Motivating example

<table>
<thead>
<tr>
<th></th>
<th>Accuracies on test sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>System A:</td>
<td>80% 50 75 ... 99</td>
</tr>
<tr>
<td>System B:</td>
<td>79 49 74 ... 98</td>
</tr>
<tr>
<td>$\delta$:</td>
<td>+1 +1 +1 ... +1</td>
</tr>
</tbody>
</table>

- Mean accuracy for System A is better, but the standard deviations for the two clearly overlap
- Notice that System A is always better than System B

Comparing systems using a paired $t$ test

- consider $\delta$’s as observed values of a set of i.i.d. random variables

- *null hypothesis*: the 2 learning systems have the same accuracy
- *alternative hypothesis*: one of the systems is more accurate than the other

- hypothesis test:
  - use paired $t$-test do determine probability $p$ that mean of $\delta$’s would arise from null hypothesis
  - if $p$ is sufficiently small (typically $< 0.05$) then reject the null hypothesis
Comparing systems using a paired \( t \) test

1. calculate the sample mean

\[
\bar{\delta} = \frac{1}{n} \sum_{i=1}^{n} \delta_i
\]

2. calculate the \( t \) statistic

\[
t = \frac{\bar{\delta}}{\sqrt{\frac{1}{n(n-1)} \sum_{i=1}^{n} (\delta_i - \bar{\delta})^2}}
\]

3. determine the corresponding \( p \)-value, by looking up \( t \) in a table of values for the Student's \( t \)-distribution with \( n-1 \) degrees of freedom

The null distribution of our \( t \) statistic looks like this.

The \( p \)-value indicates how far out in a tail our \( t \) statistic is.

If the \( p \)-value is sufficiently small, we reject the null hypothesis, since it is unlikely we'd get such a \( t \) by chance.

For a two-tailed test, the \( p \)-value represents the probability mass in these two regions.
Why do we use a two-tailed test?

- a two-tailed test asks the question: is the accuracy of the two systems different
- a one-tailed test asks the question: is system A better than system B
- a priori, we don’t know which learning system will be more accurate (if there is a difference) – we want to allow that either one might be

Comments on hypothesis testing to compare learning systems

- the paired \( t \)-test can be used to compare two learning systems
- other tests (e.g. McNemar’s \( \chi^2 \) test) can be used to compare two learned models
- a statistically significant difference is not necessarily a large-magnitude difference
Scatter plots for pairwise method comparison

We can compare the performance of two methods $A$ and $B$ by plotting ($A$ performance, $B$ performance) across numerous data sets.

![Graph 1](image1.png)

![Graph 2](image2.png)

Figure from Freund & Mason, ICML 1999

Figure from Noto & Craven, BMC Bioinformatics 2006

---

Lesion studies

We can gain insight into what contributes to a learning system’s performance by removing (lesioning) components of it.

The ROC curves here show how performance is affected when various feature types are removed from the learning representation.

![Graph 3](image3.png)

Figure from Bockhorst et al., Bioinformatics 2003
To avoid pitfalls, ask

1. Is my held-aside test data really representative of going out to collect new data?
   - Even if your methodology is fine, someone may have collected features for positive examples differently than for negatives – should be randomized
   - Example: samples from cancer processed by different people or on different days than samples for normal controls

To avoid pitfalls, ask

2. Did I repeat my entire data processing procedure on every fold of cross-validation, using only the training data for that fold?
   - On each fold of cross-validation, did I ever access in any way the label of a test instance?
   - Any preprocessing done over entire data set (feature selection, parameter tuning, threshold selection) must not use labels
To avoid pitfalls, ask

3. Have I modified my algorithm so many times, or tried so many approaches, on this same data set that I (the human) am overfitting it?

- Have I continually modified my preprocessing or learning algorithm until I got some improvement on this data set?

- If so, I really need to get some additional data now to at least test on