Evaluating machine learning outcomes

- How should we define a good outcome for a machine learning algorithm?
- How do we know when an algorithm is doing well in a particular application?
- How can we compare different classifiers?
- How can we compare different learning algorithms?

More concrete questions:

- What quantity should we measure?
- How can we estimate it?
- Can we get quantitative guarantees for estimates and comparisons?

What to Measure?

- We have so far focused on classification and looked at accuracy
- But this depends on the application and data distribution ...

Confusion Matrix

- \(+\) - \(--\) classified as
- \(\text{TP FN} \mid \text{true label +}\)
- \(\text{FP TN} \mid \text{true label -}\)

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

- In many applications one class (typically the +) has very low frequency
- Acc no longer the most appropriate criterion

Confusion Matrix

- \(+\) - \(--\) classified as
- \(\text{TP FN} \mid \text{true label +}\)
- \(\text{FP TN} \mid \text{true label -}\)

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

- In many applications one class (typically the +) has very low frequency
- Running algorithm to optimize Acc often yields bad results. Why?
IR Terminology

- **Context**: search for items with \textit{QueryTerm}
- De-emphasize role of Neg examples
- Aims to measure quality of response
  \[ \text{Precision} = \frac{TP}{TP + FP} \quad \text{Recall} = \frac{TP}{TP + FN} \]
- Tradeoff given by Precision/Recall curve
- And by \[ F = \frac{2 \cdot R \cdot P}{R + P} \]

Medical Community Terminology

- **Context**: test to identify \textit{Med Condition}
- Sensitivity = Recall = \[ \frac{TP}{TP + FN} \]
- Specificity = \[ \frac{TN}{TN + FP} \]
- These measure “accuracy” within each class

Signal Detection Terminology

- **Context**: identify “signal”
  \[ \text{TPrate} = \text{Recall} = \frac{TP}{TP + FN} \]
  \[ \text{FPrate} = 1 - \text{Specificity} = \frac{FP}{TN + FP} \]
- The ROC curve (receiver operator characteristic) plots TPrate vs FPrate

ROC Curves

- The AROC (area under ROC curve) has a nice interpretation:
  \[ \text{AROC} = \text{probability of correctly ranking a random Pos example above a random Neg example} \]

From Ranking to ROC Curve

- Many learning algorithms (e.g., Naive Bayes, Perceptron) provide a numerical output that can be used to rank examples in addition to the prediction of +/- label
- This can be used to produce a ROC curve for the algorithm by changing its threshold specifying the transition from Neg to Pos.

Evaluating machine learning outcomes

- More concrete questions:
  - What quantity should we measure?
  - How can we estimate it?
  - Can we get quantitative guarantees for estimates and comparisons?
How to Measure?

- Validation Set Method:
  - keep aside a portion of the example set
  - Train model on remaining data
  - Measure performance on validation set

  (+) Unbiased estimate of quantity
  (-) Wastes data ... 
  (-) Variance in estimate due to choice of validation set. Can we fix this?

- Can reduce variance by repeating k times and averaging
- But this introduces bias in estimates because the train/test data in different runs are highly correlated

Cross Validation Method:

- Divide data into k portions (called folds)
- Repeat k times
  - Train model on data from all folds except k
  - Measure performance on k'th fold
- Average the performance

- Test data in different folds is disjoint
- But training sets are not

Stratified Cross Validation

- Another source of variance in cross validation is the fact that the test sets across folds do not have exactly the same class frequencies
- Stratified cross validation removes this variation by splitting into random subsets of the same size per class

Implementing Stratified Cross Validation

- Split data according to class
- Randomly permute the examples in each class
- Partition each class (by permuted index) into k equal size portions (up to +/-1)
- Join portions i from all classes to get fold i
- This gives k portions of the same size and having the same class proportions

Stratified Cross Validation

- Cross Validation has become the standard and most widely used method to form estimates when evaluating and comparing algorithms in ML.
- It is not without faults.
- Alternatives have been proposed (see reading) but we won't go further into these.
Evaluating machine learning outcomes

- More concrete questions:
  - What quantity should we measure?
  - How can we estimate it?
  - Can we get quantitative guarantees for estimates and comparisons?

Quantitative Comparisons

Review notions from Normal distributions

\[ \% \text{ of area (probability)} \text{ lies in } \mu \pm 1.28\sigma \]
\[ \% \text{ of area (probability)} \text{ lies in } \mu \pm 2\sigma \]

Image from Mitchell text/slides

Model Selection

- Cross validation can be applied to configure parameters in many algorithms during the training phase.
  - Pick k for nearest neighbors
  - Pick prune threshold in DT
- In this case we perform cross validation on training set only
- Using the result we pick best parameter value
- Finally: we train again on the entire data

Cross validation can also be used to select among algorithms, among options in an algorithm, etc. We call this Model Selection.

- Cross validation can be used to estimate how well algorithm does, and to compare algorithms - we develop this topic next

The only problem is that this becomes expensive if we want to tune multiple parameters or settings of the algorithm.
Quantitative Comparisons

Review notions from Normal distributions

- 80% of area (probability) lies in $\mu \pm 1.28\sigma$
- N% of area (probability) lies in $\mu \pm z\sigma$

<table>
<thead>
<tr>
<th>N%</th>
<th>5%</th>
<th>68%</th>
<th>90%</th>
<th>95%</th>
<th>98%</th>
<th>99%</th>
</tr>
</thead>
<tbody>
<tr>
<td>zσ</td>
<td>0.67</td>
<td>1.00</td>
<td>1.28</td>
<td>1.64</td>
<td>1.96</td>
<td>2.33</td>
</tr>
</tbody>
</table>

Image from Mitchell text/slides

- From probability to estimates
  - N% of area (probability) lies in $\mu \pm z\sigma$

- Probability Statement
  - $x \in [\mu \pm z\sigma]$ [with probability N%]

- Confidence interval
  - $\mu \in [x \pm z\sigma]$ [with confidence N%]

**Confidence Interval**

- [with confidence N%] $\mu \in x \pm z\sigma$
- Assume $x \sim N(\mu, \sigma^2)$
- And we sample to observe value of $x$

  [with confidence 95%] $\mu \in x \pm 1.96\sigma$

**Evaluating one classifier**

- We measure the performance on test set of $n$ examples
- If the error rate is $p$ then the number of mistakes on the test set is distributed as Binomial(n, p)
- Our estimate is $\hat{p} = \frac{\# \text{ mistakes}}{n}$
- $\hat{p}$ distributed approx as $\hat{p} \sim N(p, \frac{p(1-p)}{n})$

**Evaluating one classifier**

- $\hat{p}$ distributed approx as $\hat{p} \sim N(p, \frac{p(1-p)}{n})$
- Therefore, we can apply

  [with confidence N%] $\mu \in x \pm z\sigma$

- To get

  [with confidence N%] $p \in \hat{p} \pm z\sigma \sqrt{\frac{p(1-p)}{n}}$

**Evaluating one classifier**

- Good news: we have an interval for $p$

  [with confidence N%] $p \in \hat{p} \pm z\sqrt{\frac{p(1-p)}{n}}$

- But: we cannot plug in $p$ in variance part

  Solution 1: $p(1-p)=0.25$

  [with confidence N%] $p \in \hat{p} \pm z\sqrt{\frac{1}{4n}}$

  Solution 2: (not justified)

  [with confidence N%] $p \in \hat{p} \pm z\sqrt{\frac{p(1-p)}{n}}$

Warning: Beware to check if using tables for one-sided bounds or two sided bounds. These slides use two sided bounds. Some other sources use one sided bounds. It is straightforward to extract one from the other.
Evaluating Learning Algorithm

- Cross validation error estimate \( \hat{e} = \frac{1}{k} \sum e_i \)
- Define \( e \) as the expected error rate when running \( \text{alg} \) (on data of this size)
- Then we have:
  \[ e_i \sim N(e, \sigma_e^2) \]
  \[ \hat{e} \sim N(e, \sigma_e^2/k) \]
- And we can use
  \[ \hat{e} \sim N(e, \sigma_e^2/k) \]
  \[ e \sim \hat{e} \pm z_{N\%} \sqrt{\frac{\sigma_e^2}{k}} \]

- But: we do not know \( \sigma_e \)
- Solution 1: plug in the estimate \( s \) instead of \( \sigma_e \)
  \[ s = \sqrt{\frac{1}{k-1} \sum (e_i - \hat{e})^2} \]
- Solution 2: a more accurate interval through \( T \) random variable
  Sol2 is more accurate and as easy to implement so avoid Sol1.

- It turns out [we skip definitions and details] that
  \[ \frac{\hat{e} - e}{(s/\sqrt{k})} \sim T_{k-1} \]
- And that this implies
  \[ e \in \hat{e} \pm t_{N\%} \frac{s}{\sqrt{k}} \]
- And more concretely
  \[ e \in \hat{e} \pm t_{N\%} \frac{\sum (e_i - \hat{e})^2}{(k-1)k} \]

The main difference from solution 1 (that plugs in value of \( s \), is that \( Z_{N\%} \) is replaced with \( t_{N\%} \)).
Comparing Two Algorithms

• We can calculate an interval for the error of each and say that one is better if the intervals do not overlap.
• But we can do much better when variance $\sigma_e$ is large.

<table>
<thead>
<tr>
<th>Alg1</th>
<th>Alg2</th>
<th>Diff</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>68</td>
<td>8</td>
</tr>
<tr>
<td>70</td>
<td>77</td>
<td>7</td>
</tr>
<tr>
<td>80</td>
<td>88</td>
<td>8</td>
</tr>
</tbody>
</table>

Here individual intervals overlap.
But interval for Diff is narrow and clearly Alg2 > Alg1.

Running many experiments and tests

• We develop 100 new algorithms.
• Calculate interval with N=95% confidence for diff over baseline alg.
• What is the probability of wrongly claiming a significant difference in at least one of these?

$$1 - 0.95^{100} = 1 - 0.059 = 0.9941$$

this is very bad
Evaluating machine learning outcomes

• You should now have some answers to these questions:

• What quantity should we measure?
• How can we estimate it?
• Can we get quantitative guarantees for estimates and comparisons?