Key Questions

Assume that dataset $S$ is generated from a probability distribution $P$.

1. When algorithm $A$ is run on $S$ to produce $f_a$, what is the error rate of $f_a$?
2. When two algorithms $A$ and $A'$ are run on $S$ to produce $f_a$ and $f_a'$, does $f_a$ have a lower error rate than $f_a'$?

The *train-and-test* method uses $S$ to empirically answer these questions.

---

Train and Test

1. Split the sample $S$ into a training set $T_1$ and a test set $T_2$.
2. Run the algorithm $A$ on $T_1$ to produce $f_{a1}$.
3. Determine the error rate of $f_{a1}$ on $T_2$?
Properties of Holdout Method

This is often called the *holdout* method.

Because $T_1 \neq S$, it is likely that $f_{a1} \neq f_a$.

There is a tradeoff in the sizes of $T_1$ and $T_2$.

Making $T_1$ larger likely decreases the difference between $f_{a1}$ and $f_a$.

Making $T_1$ larger decreases the accuracy of the test error rate on $f_{a1}$.

---

Difference between Test Error and True Error

For different test set sizes, this shows a 95% confidence bound on $|f_{a1} \text{ error} - \text{test error}|$
More on Holdout Method

Traditionally, 2/3 of the dataset is used for training, 1/3 for testing.

Results can be analyzed using standard statistics.

Holdout tends to be pessimistic because it doesn’t account for what is learned from $T_2$.

Holdout doesn’t account for variance due to algorithm.

Statistics for Holdout Method

\[
\begin{align*}
n &= |T_2| = \text{the number of test examples} \\
\delta_i &= \begin{cases} 
0 & \text{if } f_{a1}(x_i) = y_i \\
1 & \text{if } f_{a1}(x_i) \neq y_i
\end{cases} \text{ or 0-1 loss calc.} \\
u &= \frac{\sum_{i=1}^{n} \delta_i}{n} = \text{average 0-1 loss} \\
s^2 &= \frac{\sum_{i=1}^{n} (\delta_i - u)^2}{n - 1} = \text{sample variance} \\
z &= \frac{s}{\sqrt{n}} = z \text{ statistic}
\end{align*}
\]
Poor but quick: \( u \pm 1/\sqrt{n} \) loosely estimates a 95% confidence interval.

Good: For large enough \( n \), the following is a 95% confidence interval:

\[
    u \pm 1.96z
\]

If \( 5 \geq n u(1 - u) \), then this is an acceptable approximation.

Better: Use the critical value from the \( t \) distribution instead of 1.96. Here are some critical values for a 95% confidence interval.

<table>
<thead>
<tr>
<th>( n )</th>
<th>Critical Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2.262</td>
</tr>
<tr>
<td>20</td>
<td>2.093</td>
</tr>
<tr>
<td>30</td>
<td>2.045</td>
</tr>
<tr>
<td>40</td>
<td>2.023</td>
</tr>
<tr>
<td>50</td>
<td>2.010</td>
</tr>
<tr>
<td>60</td>
<td>2.001</td>
</tr>
<tr>
<td>70</td>
<td>1.995</td>
</tr>
<tr>
<td>80</td>
<td>1.990</td>
</tr>
<tr>
<td>90</td>
<td>1.987</td>
</tr>
<tr>
<td>100</td>
<td>1.984</td>
</tr>
</tbody>
</table>
Multiple Train-and-Test

A single train-and-test experiment can be misleading.

1. Sampling variance especially for small samples.

2. Algorithm variance, more for NNs than for others.

3. Does not measure learning from $T_2$.

Multiple train-and-test experiments can alleviate these difficulties.

Leave-One-Out Cross-Validation

For each $(x_i, y_i) \in S$,

1. $T_i = S - \{(x_i, y_i)\}$.

2. Use algorithm $A$ on $T_i$ to produce $f_{ai}$

3. Run $f_{ai}$ on $(x_i, y_i)$

Apply same statistics as holdout, using:

$$\delta_i = \begin{cases} 0 & \text{if } f_{ai}(x_i) = y_i \\ 1 & \text{if } f_{ai}(x_i) \neq y_i \end{cases}$$ or 0-1 loss calculation
Properties of Leave-One-Out

Generally, leave-one-out is the most accurate method, but also requires the most computation.

Each $f_{ai}$ is likely very close to $f_a$ because the sample only differs by one example.

Each example is used once as a test example.

Bootstrapping (see handout) might be more accurate for very small samples.

$k$-Fold Cross-Validation

1. Split $S$ into $k$ subsets $F_1, F_2, \ldots, F_k$ as evenly as possible.

2. For each fold $F_i$,
   
   (a) $T_i = S - F_i$
   
   (b) Use algorithm $A$ on $T_i$ to produce $f_{ai}$
   
   (c) Determine error rate (average 0-1 loss) of $f_{ai}$ on $F_i$
Statistics for $k$-Fold Cross-Validation

$\Delta_i = \text{average 0-1 loss of } f_{ai} \text{ on } F_i$

$u = \frac{\sum_{i=1}^{k} \Delta_i}{k} = \text{average 0-1 loss}$

$s^2 = \frac{\sum_{i=1}^{k} (\Delta_i - u)^2}{k - 1} = \text{sample variance}$

$t = \frac{s}{\sqrt{k}} = t \text{ statistic}$

Use $t$-dist. critical value for $n = k$. E.g., for $k = 10$, confidence interval is $u \pm 2.262s/\sqrt{k}$.

Stratified $k$-Fold Cross-Validation

Randomly distribute the examples among the folds subject to the following conditions:

1. Each fold contains either $\lfloor n/k \rfloor$ or $\lceil n/k \rceil$ examples.

2. If there are $n_i$ examples of class $i$, then each fold contains either $\lfloor n_i/k \rfloor$ or $\lceil n_i/k \rceil$ examples of class $i$.

This ensures that each training set and fold approximates the prevalence of each class in the overall sample.
Properties of $k$-Fold Cross-Validation

10-fold CV is the most commonly used method. It is not as accurate as leave-one-out, but it requires much less computation, though still over 10 times more than holdout.

Each example is used once as a test example. Because each training set is 90% of the sample, some variance will result from algorithm. Stratified CV reduces algorithm variance.