Leave-One-Out

Observations:

- The quality of the test error estimate $err_Q$ in the hold-out method greatly depends on the random split of the data set $D$ into a training set and a test set.

- A poorly executed split can adversely affect the model evaluation.

- One way to mitigate the bias of the random split of $D$ is to perform the split-train-test cycle multiple times.
Leave-One-Out

In the leave-one-out method we split the data set $D$ of size $l$ into $l$ partitions of size 1 such that,

$$D = Q_1 \cup Q_2 \cup \ldots \cup Q_{l-1} \cup Q_l,$$

and

$$Q_i \cap Q_j = \emptyset,$$

where $Q_i = \{(x_i, y_i)\}$ and $Q_j = \{(x_j, y_j)\}$ for $i, j = 1, \ldots, l$ and $i \neq j$.

Each partition $Q_i$ is used systematically for testing exactly once whereas the remaining partitions are used for training. Let $P_i = D - Q_i$ be the training set with respect to the test partition $Q_i$ with $i = 1, \ldots, l$, then we can compute the error for each test partition as

$$\text{err}_{Q_i} \left[ \hat{f}_{P_i}[k, \lambda, C] \right] = \mathcal{L} \left( y_i, \hat{f}_{P_i}[k, \lambda, C](\overline{x}_i) \right),$$

where $\hat{f}_{P_i}[k, \lambda, C]$ is the model trained on data set $P_i$ with parameters $k$, $\lambda$, and $C$.

The test error $\text{err}_{Q_i}$ is computed as the loss over the single element in the test partition $Q_i$. 
The leave-one-out error (LOOE) is the average error over all partitions,

\[
\text{LOOE}_D [k, \lambda, C] = \frac{1}{l} \sum_{i=1}^{l} \text{err}_{Q_i} \left[ \hat{f}_{P_i} [k, \lambda, C] \right].
\]

**Observation:** The leave-one-out error is an error estimate only in terms of the model parameters.

We can compute the set of parameters that minimizes the leave-one-out error over all partitions as,

\[
(k^*, \lambda^*, C^*) = \arg\min_{k, \lambda, C} \text{LOOE}_D [k, \lambda, C],
\]

and this parameter set gives rise to the optimal model

\[
\hat{f}_D [k^*, \lambda^*, C^*].
\]
Leave-One-Out

Observation: For a data set $D$ of length $l$ we have to build $l$ models for each parameter set evaluation. This implies that for most real-world data sets whose lengths is in the thousands and perhaps millions of observations this approach becomes unfeasible.
A good compromise between the potential bias of the hold-out method and the computational complexity of the leave-one-out method is \textit{N-fold cross-validation}.

Here we split the data set \( D \) into \( N \) partitions or \textit{folds} with \( N \ll l \) such that

\[
D = Q_1 \cup Q_2 \cup \ldots \cup Q_{N-1} \cup Q_N,
\]

and

\[
Q_i \cap Q_j = \emptyset,
\]

with \(|Q_i| = |Q_j| = l/N\) for \( i, j = 1, \ldots, N \) and \( i \neq j \).

We will use each fold for testing exactly once and the remaining folds are used to train the models. Let \( Q_i \) be a fold of the dataset \( D \), then we can construct our corresponding training set \( P_i \) as

\[
P_i = D - Q_i,
\]

with \( i = 1, \ldots, N \). We can compute the error of some fold \( Q_i \) as

\[
\text{err}_{Q_i} \left[ \hat{f}_{P_i}[k, \lambda, C] \right] = \frac{1}{|Q_i|} \sum_{(x_j, y_j) \in Q_i} \mathcal{L} \left( y_j, \hat{f}_{P_i}[k, \lambda, C](x_j) \right),
\]

where \( \hat{f}_{P_i}[k, \lambda, C] \) is the model trained on dataset \( P_i \) with parameters \( k, \lambda, \) and \( C \).
We compute the *cross-validated error* (CVE) of the parameter set $k$, $\lambda$, and $C$ as the average over the individual fold errors,

$$\text{CVE}_D [k, \lambda, C] = \frac{1}{N} \sum_{i=1}^{N} \text{err}_{Q_i} \left[ \hat{f}_{P_i} [k, \lambda, C] \right].$$

And we find the optimal parameter set by minimizing the cross-validated error,

$$(k^*, \lambda^*, C^*) = \arg\min_{k,\lambda,C} \text{CVE}_D [k, \lambda, C].$$

The optimal model $\hat{f}_D [k^*, \lambda^*, C^*]$ can then be constructed using the full data set $D$. 
> svm.model <- svm(Diagnosis ~ .,
                 data=wdbc.df,
                 type="C-classification",
                 kernel="polynomial",
                 degree=3,
                 cost=1000,
                 cross=10)

> summary(svm.model)

10-fold cross-validation on training data:

Total Accuracy: 94.55185
Single Accuracies:
  91.07143 94.73684 98.24561 96.49123 100
  87.7193 94.73684 94.73684 94.73684 92.98246
### $N$-Fold Cross-Validation

<table>
<thead>
<tr>
<th>ID</th>
<th>Kernel</th>
<th>Cost Constant</th>
<th>Training Error</th>
<th>Cross-Validated Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Linear</td>
<td>0.01</td>
<td>2.46%</td>
<td>3.51%</td>
</tr>
<tr>
<td>2</td>
<td>Linear</td>
<td>0.10</td>
<td>1.41%</td>
<td>2.46%</td>
</tr>
<tr>
<td>3</td>
<td>Linear</td>
<td>1.00</td>
<td>1.23%</td>
<td>2.81%</td>
</tr>
<tr>
<td>4</td>
<td>Linear</td>
<td>10.00</td>
<td>0.88%</td>
<td>3.34%</td>
</tr>
<tr>
<td>5</td>
<td>Linear</td>
<td>100.00</td>
<td>0.35%</td>
<td>3.34%</td>
</tr>
<tr>
<td>6</td>
<td>Linear</td>
<td>1000.00</td>
<td>0.35%</td>
<td>3.87%</td>
</tr>
<tr>
<td>7</td>
<td>Polynomial, degree = 3</td>
<td>10.00</td>
<td>2.81%</td>
<td>4.39%</td>
</tr>
<tr>
<td>8</td>
<td>Polynomial, degree = 3</td>
<td>100.00</td>
<td>0.53%</td>
<td>3.34%</td>
</tr>
<tr>
<td>9</td>
<td>Polynomial, degree = 3</td>
<td>1000.00</td>
<td>0.00%</td>
<td>5.45%</td>
</tr>
</tbody>
</table>