A Recipe for Learning

\[
P(h | D) = \frac{P(D | h)P(h)}{P(D)}
\]

\(P(h)\) = prior probability of hypothesis \(h\)
\(P(D)\) = prior probability of training data \(D\)
\(P(h | D)\) = probability of \(h\) given \(D\)
\(P(D | h)\) = probability of \(D\) given \(h\)
A Recipe for learning

Maximum a posteriori hypothesis

\[ h_{MAP} = \arg \max_{h \in H} P(h | D) \]
\[ = \arg \max_{h \in H} \frac{P(D | h)P(h)}{P(D)} \]
\[ = \arg \max_{h \in H} P(D | h)P(h) \]

Maximum likelihood hypothesis

If \( \forall h_i, h_j \in H \quad P(h_i) = P(h_j) \),

\[ h_{ML} = \arg \max_{h \in H} P(D | h) \]
Brute Force MAP Hypothesis Learner

• For each hypothesis $h$ in $H$, calculate the posterior probability

$$P(h | D) = \frac{P(D | h)P(h)}{P(D)}$$

• Output the hypothesis with the highest posterior probability

$$h_{MAP} = \arg \max_{h_i \in H} P(D | h_i)P(h_i)$$

$$h_{ML} = \arg \max_{h_i \in H} P(D | h_i)$$
Brute Force MAP Hypothesis Learner

\[ h_{MAP} = \arg\max_{h_i \in H} P(D \mid h_i)P(h_i) \]
\[ = \arg\max_{h_i \in H} \left( \log_2 P(D \mid h_i) + \log_2 P(h_i) \right) \]
\[ = \arg\min_{h_i \in H} \left( -\log_2 P(D \mid h_i) - \log_2 P(h_i) \right) \]

\( -\log_2 P(h_i) = \) description length of \( h_i \) under optimal encoding of hypotheses

\( -\log_2 P(D \mid h_i) = \) description length of data \( D \) given \( h_i \)

under optimal encoding based on \( P(D \mid h_i) \)

\( \therefore \) Bayesian learning implies Occam's razor!
MAP Learning of 2-class classifiers

Consider a 2-class learning problem defined over an instance space $\mathbf{X}$, hypothesis space $H$, training examples

$$D = \{ e_i = (X_i, d_i) | X_i \in \mathbf{X}; d_i = c(X_i) \}$$

Consider a learning algorithm which outputs a hypothesis that is consistent with the examples in $D$

- Let $V_{H,D}$ be the subset of hypotheses in $H$ that are consistent with $D$ (the version space)
- What would Bayes rule produce as the MAP hypothesis?
MAP Learning of Binary Concepts (2-class classifiers)

Assumptions

1. The hypothesis space is finite.
2. The training examples are i.i.d.
3. The training data is noise free.
   \[ \forall d_i, \ d_i = c(X_i) \]
4. The target concept \( c \in H \)
5. All hypotheses are a priori equally likely:
   \[ \forall h \in H, \ P(h) = \frac{1}{|H|} \]
MAP Learning of 2-class classifiers

\[ P(D \mid h) = \prod_i P(e_i \mid h) = \prod_i P(X_i, d_i \mid h) = \prod_i P(d_i \mid h, X_i)P(X_i \mid h) \]

Assuming \( X_i \) are independent of \( h \),
\[ P(D \mid h) = \prod_i P(d_i \mid h, X_i)P(X_i) \]
\[ P(d_i \mid h, X_i) = 1 \text{ if } h(X_i) = d_i; \]
\[ P(d_i \mid h, X_i) = 0 \text{ otherwise} \]
MAP Learning of 2-class classifiers

If $h$ is consistent with $D$, that is,
\[ \forall i \in \{1...m\}, d_i = h(X_i) \]
then
\[
P(D \mid h) = \prod_{i=1}^{m} P(X_i)
\]

If $h$ is consistent with $D$, that is,
\[ \exists i \in \{1...m\} \text{ such that } d_i \neq h(X_i) \]
then
\[
P(D \mid h) = 0
\]
MAP Learning of 2-class classifiers

\[ P(D) = \sum_{h \in H} P(D | h)P(h) = \sum_{h \in V_{H,D}} \left( \frac{1}{|H|} \right) \left( \prod_{i} P(X_i) \right) + \sum_{h \in H} 0 \]

\[ = \frac{|V_{H,D}|}{|H|} \left( \prod_{i} P(X_i) \right) \]

\[ \therefore \forall h \in V_{H,D}, \text{ we have} \]

\[ P(h | D) = \frac{P(D|h)P(h)}{P(D)} = \left( \frac{\prod_{i} P(X_i)}{|H|} \right) \left( \frac{|H|}{|V_{H,D}| \left( \prod_{i} P(X_i) \right)} \right) = \left( \frac{1}{|V_{H,D}|} \right) \]

and \( \forall h \notin V_{H,D}, \text{ we have} P(h | D) = 0 \)

Every hypothesis in \( H \) that is consistent with \( D \) is a MAP hypothesis

Copyright Vasant Honavar, 2006.
Bayesian Recipe for Learning

If the training examples are
• independently identically distributed
• noise-free
And if each candidate hypothesis in $H$ is equally a priori equally likely
Then every hypothesis that is consistent with the training data (that is, correctly classifies each training example) maximizes $P(h|D)$

In this setting, all that the learner has to do is to produce a hypothesis from $H$ that is consistent with the training data
Effect of Data (Evidence) on the Version Space
Bayesian Recipe for Learning

\[ h_{MAP} = \arg \min_{h_i \in H} \left( -\log_2 P(D|h_i) - \log_2 P(h_i) \right) \]

The number of bits needed to encode the Data \( D \) given the hypothesis \( h_i \) (under optimal encoding) – in other words, error of \( h \) or exceptions to \( h \)

The number of bits needed to encode the hypothesis \( h_i \) (under optimal encoding)
Bayesian Recipe for Learning

Suppose the training data are iid and noise-free but each hypothesis in $H$ is not equally a priori equally likely

$$h_{MAP} = \arg \max_{h_i \in H} P(D \mid h_i)P(h_i)$$

$$= \arg \max_{h_i \in H} \left( \log_2 P(D \mid h_i) + \log_2 P(h_i) \right)$$

$$= \arg \min_{h_i \in H} \left( - \log_2 P(D \mid h_i) - \log_2 P(h_i) \right)$$
Bayesian Recipe for Learning

If
• the training data are iid and noise-free
• but each hypothesis in $H$ is not equally a priori equally likely

then
• the learner has to produce a hypothesis from $H$ that trades off the error on the training data against complexity of the hypothesis

Decision tree classifiers, which we will look at next, provide an example of this approach
Learning Decision Tree Classifiers

On average, the information needed to convey the class membership of a random instance drawn from nature is \( H(\hat{P}) \)

\[
H(\hat{P}) = -\sum_{i=1}^{m} (\hat{p}_i) \log_2 (\hat{p}_i) = \hat{H}(X)
\]

where \( \hat{P} \) is an estimate of \( P \) and \( X \) is a random variable with distribution \( P \). 

\( S_i \) is the multi-set of examples belonging to class \( C_i \)
Learning Decision Tree Classifiers

The task of the learner then is to extract the needed information from the training set and store it in the form of a decision tree for classification.

**Information gain based decision tree learner**

Start with the entire training set at the root.

Recursively add nodes to the tree corresponding to tests that yield the greatest expected reduction in entropy (or the largest expected information gain) until some termination criterion is met (e.g., the training data at every leaf node has zero entropy).
Learning Decision Tree Classifiers - Example

<table>
<thead>
<tr>
<th>Training Data</th>
<th>Class label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance</td>
<td></td>
</tr>
<tr>
<td>( I_1 )</td>
<td>+</td>
</tr>
<tr>
<td>( I_2 )</td>
<td>+</td>
</tr>
<tr>
<td>( I_3 )</td>
<td>–</td>
</tr>
<tr>
<td>( I_4 )</td>
<td>–</td>
</tr>
<tr>
<td>( I_5 )</td>
<td>–</td>
</tr>
<tr>
<td>( I_6 )</td>
<td>+</td>
</tr>
<tr>
<td>( I_7 )</td>
<td>+</td>
</tr>
<tr>
<td>( I_8 )</td>
<td>+</td>
</tr>
</tbody>
</table>

Instances – ordered 3-tuples of attribute values corresponding to:
- *Height* (tall, short)
- *Hair* (dark, blonde, red)
- *Eye* (blue, brown)
Learning Decision Tree Classifiers - Example

\[
\hat{H}(X) = -\frac{3}{8} \log_2 \frac{3}{8} - \frac{5}{8} \log_2 \frac{5}{8} = 0.954 \text{bits}
\]

\[
\hat{H}(X | \text{Height} = t) = -\frac{2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5} = 0.971 \text{bits}
\]

\[
\hat{H}(X | \text{Height} = s) = -\frac{2}{3} \log_2 \frac{2}{3} - \frac{1}{3} \log_2 \frac{1}{3} = 0.918 \text{bits}
\]

\[
\hat{H}(X | \text{Height}) = \frac{5}{8} \hat{H}(X | \text{Height} = t) + \frac{3}{8} \hat{H}(X | \text{Height} = s) = \frac{5}{8}(0.971) + \frac{3}{8}(0.918) = 0.95 \text{bits}
\]

Similarly, \( \hat{H}(X | \text{Eye}) = 0.607 \text{bits} \) and \( \hat{H}(X | \text{Hair}) = 0.5 \text{bits} \)

\text{Hair} is the most informative because it yields the largest reduction in entropy. Test on the value of \text{Hair} is chosen to correspond to the root of the decision tree.
Learning Decision Tree Classifiers - Example

Compare the result with Naïve Bayes

In practice, we need some way to prune the tree to avoid overfitting the training data – more on this later.
Learning, generalization, overfitting

Consider the error of a hypothesis \( h \) over

- training data: \( Error_{\text{Train}}(h) \)
- entire distribution \( D \) of data: \( Error_D(h) \)

Hypothesis \( h \in H \) over fits training data if there is an alternative hypothesis \( h' \in H \) such that

\[
Error_{\text{Train}}(h) < Error_{\text{Train}}(h')
\]

\[
Error_D(h) > Error_D(h')
\]
Over fitting in decision tree learning

![Graph showing accuracy vs size of tree (number of nodes)]
Causes of over fitting

As we move further away from the root, the data set used to choose the best test becomes smaller → poor estimates of entropy

Noisy examples can further exacerbate over fitting
Minimizing over fitting

• Use roughly the same size sample at every node to estimate entropy – when there is a large data set from which we can sample

• Stop when further split fails to yield statistically significant information gain (estimated from validation set)

• Grow full tree, then prune

• minimize $\text{size (tree)} + \text{size (exceptions (tree))}$
Reduced error pruning

Each decision node in the tree is considered as a candidate for pruning

Pruning a decision node consists of
- removing the sub tree rooted at that node,
- making it a leaf node, and
- assigning it the most common label at that node or storing the class distribution at that node (for probabilistic classification)
Reduced error pruning – Example

<table>
<thead>
<tr>
<th>Node</th>
<th>Accuracy gain by Pruning</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-20%</td>
</tr>
<tr>
<td>B</td>
<td>+10%</td>
</tr>
</tbody>
</table>

Before Pruning

After Pruning
Reduced error pruning

Do until further pruning is harmful:

- Evaluate impact on validation set of pruning each candidate node
- Greedily select a node which most improves the performance on the validation set when the sub tree rooted at that node is pruned

Drawback – holding back the validation set limits the amount of training data available – not desirable when data set is small
Reduced error pruning

Pruned Tree

Accuracy

Size of tree (number of nodes)
Evaluate Candidate split to decide if the resulting information gain is significantly greater than zero as determined using a suitable hypothesis testing method at a desired significance level.

\[ n_1 = 50, \ n_2 = 50, \ N = n_1 + n_2 = 100 \]

\[ n_{1L} = 50, \ n_{2L} = 0, \ n_L = 50, \ p = \frac{n_L}{N} = 0.5 \]

\[ n_{1e} = \text{number of class 1 instances (random split)} \]
\[ n_{1e} = pn_1 = 25, \ n_{2e} = pn_2 = 25 \]

\[ \chi^2 = \frac{(n_{1L} - n_{1e})^2}{n_{1e}} + \frac{(n_{2L} - n_{2e})^2}{n_{2e}} = 25 + 25 = 50 \]

This split is significantly better than random with confidence > 99% because \( \chi^2 > 6.64 \)
Pruning based on whether information gain is significantly greater than zero

Evaluate Candidate split to decide if the resulting information gain is significantly greater than zero as determined using a suitable hypothesis testing method at a desired significance level.

Example: $\chi^2$ statistic

In the 2-class, binary (L,R) split case,

$$\chi^2 = \sum_{i=1}^{2} \frac{(n_{iL} - n_{ie})^2}{n_{ie}}$$

$n_1$ of class 1, $n_2$ of class 2; $N = n_1 + n_2$

Split sends $pN$ to L and $(1-p)N$ to R

Random split would send $pn_1$ of class 1 to L and $pn_2$ of class 2 to L

The critical value of $\chi^2$ depends on the degrees of freedom which is 1 in this case (for a given $p$, $n_{1L}$ fully specifies $n_{2L}$, $n_{1R}$ and $n_{2R}$)

In general, the number of degrees of freedom can be $> 1$
Pruning based on whether information gain is significantly greater than zero

\[ \chi^2 = \sum_{j=1}^{\text{Branches}} \sum_{i=1}^{\text{Classes}} \left( \frac{n_{ij} - n_{iej}}{n_{iej}} \right)^2 \]

The greater the value of \( \chi^2 \) the less likely it is that the split is random. For a sufficiently high value of \( \chi^2 \), the difference between the expected (random) split is statistically significant and we reject the null hypothesis that the split is random.

\[ N = n_1 + n_2 + \ldots n_{\text{Classes}} \]

\[ p = [p_1 p_2 \ldots p_{\text{Branches}}]; \sum_{j=1}^{\text{Branches}} p_j = 1 \]

\[ n_{iej} = p_j n_i \]

Degrees of freedom = \((\text{Classes} - 1)(\text{Branches} - 1)\)
Rule post-pruning

Convert tree to equivalent set of rules

IF \((Outlook = Sunny) \land (Humidity = High)\)
THEN \(PlayTennis = No\)

IF \((Outlook = Sunny) \land (Humidity = High)\)
THEN \(PlayTennis = Yes\)

...
Rule post-pruning

1. Convert tree to equivalent set of rules
2. Prune each rule independently of others by dropping a condition at a time if doing so does not reduce estimated accuracy (at the desired confidence level)
3. Sort final rules in order of lowest to highest error

Advantage – can potentially correct bad choices made close to the root

Post pruning based on validation set is the most commonly used method in practice

Development of pre pruning methods with comparable performance that do not require a validation set is an open problem
Classification of instances

- Unique classification – possible when each leaf has zero entropy and there are no missing attribute values
- Most likely classification – based on distribution of classes at a node when there are no missing attribute values
- Probabilistic classification – based on distribution of classes at a node when there are no missing attribute values
Handling different types of attribute values

Types of attributes
• Nominal – values are names
• Ordinal – values are ordered
• Cardinal (Numeric) – values are numbers (hence ordered)

…. 
Handling numeric attributes

<table>
<thead>
<tr>
<th>Attribute T</th>
<th>40</th>
<th>48</th>
<th>50</th>
<th>54</th>
<th>60</th>
<th>70</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
</tbody>
</table>

Candidate splits

\[
T > \left( \frac{48 + 50}{2} \right) \quad T > \left( \frac{60 + 70}{2} \right)
\]

\[
E(S \mid T > 49?) = \frac{2}{6} (0) + \frac{4}{6} \left( -\left( \frac{3}{4} \right) \log_2 \left( \frac{3}{4} \right) - \left( \frac{1}{4} \right) \log_2 \left( \frac{1}{4} \right) \right)
\]

- Sort instances by value of numeric attribute under consideration
- For each attribute, find the test which yields the lowest entropy
- Greedily choose the best test across all attributes
Handling numeric attributes

Axis-parallel split

Oblique split

Oblique splits cannot be realized by univariate tests
Two-way versus multi-way splits

Entropy criterion favors many-valued attributes

Pathological behavior – what if in a medical diagnosis data set, social security number is one of the candidate attributes? $A = value$ versus $A = \neg value$

Solutions

Only two-way splits (CART)

Entropy ratio (C4.5)

\[
EntropyRatio(S \mid A) \equiv \frac{Entropy(S \mid A)}{SplitEntropy(S \mid A)}
\]

\[
SplitEntropy(S \mid A) \equiv -\sum_{i=1}^{\text{Values}(A)} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}
\]
Alternative split criteria

Consider split of set $S$ based on attribute $A$

$$\text{Impurity}(S \mid A) = \sum_{j=1}^{\text{|Values}(A|)} \text{Impurity}(S_j)$$

Entropy

$$\text{Impurity}(Z) = \sum_{i=1}^{\text{Classes}} -\frac{|Z_i|}{|Z|} \log_2 \frac{|Z_i|}{|Z|}$$

Gini

$$\text{Impurity}(Z) = \sum_{i \neq j} \left( \frac{|Z_i|}{|Z|} \right) \left( \frac{|Z_j|}{|Z|} \right) = 1 - \sum_{i=1}^{\text{Classes}} \left( \frac{|Z_i|}{|Z|} \right)^2$$

(Expected rate of error if class label is picked randomly according to distribution of instances in a set)
Alternative split criteria

One-sided split criteria – often useful for exploratory analysis of data

\[
\text{Impurity}(S \mid A) = \min_{i \in \text{Values}(A)} \left\{ \text{Impurity}(S_i) \right\}
\]
Incorporating Attribute costs

Not all attribute measurements are equally costly or risky

In Medical diagnosis
  Blood-Test has cost $150
  Exploratory-Surgery may have a cost of $3000

Goal: Learn a Decision Tree Classifier which minimizes cost of classification

Tan and Schlimmer (1990) \[ \frac{Gain^2(S,A)}{Cost(A)} \]

Nunez (1988) \[ 2^{Gain(S,A)} - 1 \]
\[ (Cost(A) + 1)^w \]

where \( w \in [0, 1] \) determines importance of cost
Incorporating Different Misclassification Costs for different classes

Not all misclassifications are equally costly
An occasional false alarm about a nuclear power plant meltdown is less costly than the failure to alert when there is a chance of a meltdown

Weighted Gini Impurity

$$\text{Impurity}(S) = \sum_{ij} \lambda_{ij} \left( \frac{|S_i|}{|S|} \right) \left( \frac{|S_j|}{|S|} \right)$$

$\lambda_{ij}$ is the cost of wrongly assigning an instance belonging to class $i$ to class $j$
Dealing with Missing Attribute Values (Solution 1)

Sometimes, the fact that an attribute value is missing might itself be informative –
Missing blood sugar level might imply that the physician had reason not to measure it
Introduce a new value (one per attribute) to denote a missing value
Decision tree construction and use of tree for classification proceed as before
Dealing with Missing Attribute Values Solution 2

During decision tree construction
Replace a missing attribute value in a training example with the most frequent value found among the instances at the node

During use of tree for classification
Replace a missing attribute value in an instance to be classified with the most frequent value found among the training instances at the node
Dealing with Missing Attribute Values (Solution 3)

During decision tree construction
Replace a missing attribute value in a training example with the most frequent value found among the instances at the node that have the same class label as the training example

During use of tree for classification
Assign to a missing attribute the most frequent value found at the node (based on the training sample)
Sort the instance through the tree to generate the class label
Dealing with Missing Attribute Values (Solution 4)

During decision tree construction
Generate several fractionally weighted training examples based on the distribution of values for the corresponding attribute at the node

During use of tree for classification
Generate multiple instances by assigning candidate values for the missing attribute based on the distribution of instances at the node
Sort each such instance through the tree to generate candidate labels and assign the most probable class label or probabilistically assign class label
Dealing with Missing Attribute Values

\[ n_+ = 60, n_- = 40 \]

\[ (n_+ | A = 1) = 50 \]

\[ (n_+ | A = 0) = 10; (n_- | A = 0) = 40 \]

Suppose B is missing

Replacement with most frequent value at the node \( \rightarrow B = 1 \)

Replacement with most frequent value if class is + \( \rightarrow B = 0 \)
Dealing with Missing Attribute Values

\[
\begin{align*}
A & \quad n_+ = 60, n_- = 40 \\
\quad 1 & \quad (n_+ | A = 1) = 50 \\
\quad 0 & \\
B & \\
\quad 1 & \quad (n_+ | A = 0) = 10; (n_- | A = 0) = 40 \\
\quad 0 & \\
\quad 1 & \\
\quad 0 & \quad (n_+ | A = 0, B = 1) = 40 \\
\quad & \quad (n_+ | A = 0, B = 0) = 10 \\
\quad & \\
\quad & \\
\quad & \\
\text{Suppose B is missing} \\
\end{align*}
\]

Fractional instance based on distribution at the node .. 4/5 for B=1, 1/5 for B=0

Fractional instance based on distribution at the node for class + ..

1/5 for B=0, 0 for B=1
Recent Developments in Decision Tree Classifiers

• Learning Decision Trees from Distributed Data (Caragea, Silvescu and Honavar, 2004)
• Learning Decision Trees from Attribute Value Taxonomies and partially specified data (Zhang and Honavar, 2003; 2004; Zhang et al., 2005)
• Learning Decision Trees from Relational Data (Atramentov, Leiva, and Honavar, 2003)
• Decision Trees for Multi-label Classification Tasks
• Learning Decision Trees from Very Large Data Sets
• Learning Decision Trees from Data Streams
Summary of Decision Trees

Simple

Fast (Linear in size of the tree, linear in the size of the training set, linear in the number of attributes)

Produce easy to interpret rules

Good for generating simple predictive rules from data with lots of attributes
Neural Networks

- Decision trees are good at modeling nonlinear interactions among a small subset of attributes
- Sometimes we are interested in linear interactions among all attributes
- Simple neural networks are good at modeling such interactions
- The resulting models have close connections with naïve Bayes.
Measuring classifier performance

To simplify matters, assume that class labels are binary. M-class problem is turned into M 2-class problems.
Measuring Classifier Performance

\( N \): Total number of instances in the data set
\( TP_j \): Number of True positives for class \( j \)
\( FP_j \): Number of False positives for class \( j \)
\( TN_j \): Number of True Negatives for class \( j \)
\( FN_j \): Number of False Negatives for class \( j \)

\[
Accuracy_j = \frac{TP_j + TN_j}{N} = P(class = c_j \land label = c_j)
\]

Perfect classifier \( \leftrightarrow \) Accuracy = 1

Popular measure

Biased in favor of the majority class!

Should be used with caution!
Measuring Classifier Performance: Sensitivity

\[
Sensitivity_j = \frac{TP_j}{TP_j + FN_j} \\
= \frac{Count(label = c_j \land class = c_j)}{Count(class = c_j)} \\
= P(label = c_j \mid class = c_j)
\]

Perfect classifier \(\rightarrow\) Sensitivity = 1
Probability of correctly labeling members of the target class
Also called recall or hit rate
Measuring Classifier Performance: Specificity

\[ \text{Specificity}_j = \frac{TP_j}{TP_j + FP_j} = \frac{\text{Count}(\text{label} = c_j \land \text{class} = c_j)}{\text{Count}(\text{label} = c_j)} = P(\text{class} = c_j \mid \text{label} = c_j) \]

Perfect classifier \( \Rightarrow \) Specificity = 1
Also called precision
Probability that a positive prediction is correct
## Classifier Learning -- Measuring Performance

<table>
<thead>
<tr>
<th>Class Label</th>
<th>$C_1$</th>
<th>$\neg C_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1$</td>
<td>TP= 55</td>
<td>FP=5</td>
</tr>
<tr>
<td>$\neg C_1$</td>
<td>FN=10</td>
<td>TN=30</td>
</tr>
</tbody>
</table>

$$N = TP + FN + TN + FP = 100$$

$$sensitivity_1 = \frac{TP}{TP + FN} = \frac{55}{65}$$

$$specificity_1 = \frac{TP}{TP + FP} = \frac{55}{65}$$

$$accuracy_1 = \frac{TP + TN}{N} = \frac{85}{100}$$
Measuring Performance: Precision, Recall, and False Alarm Rate

\[
\text{Precision}_j = \text{Specificity}_j = \frac{TP_j}{TP_j + FP_j}
\]

\[
\text{Recall}_j = \text{Sensitivity}_j = \frac{TP_j}{TP_j + FN_j}
\]

Perfect classifier → Precision=1  Perfect classifier → Recall=1

\[
\text{False Alarm Rate} = \frac{FP_j}{TN_j + FP_j}
\]

\[
= \frac{\text{Count}(\text{label} = c_j \wedge \text{class} = \neg c_j)}{\text{Count}(\text{label} = \neg c_j)}
\]

\[
= P(\text{label} = c_j \mid \text{class} = \neg c_j)
\]

Perfect classifier → False Alarm Rate = 0
Measuring Performance – Correlation Coefficient

\[ CC_j = \frac{(TP_j \times TN_j) - (FP_j \times FN_j)}{\sqrt{(TP_j + FN_j)(TP_j + FP_j)(TN_j + FP_j)(TN_j + FN_j)}} \]

\[-1 \leq CC_j \leq 1\]

Perfect classifier \( \leftrightarrow \) \( CC = 1 \), Random guessing \( \leftrightarrow \) \( CC = 0 \)

Corresponds to the standard measure of correlation between two random variables \( Label \) and \( Class \) estimated from labels \( L \) and the corresponding class values \( C \) for the special case of binary (0/1) valued labels and classes

\[ CC_j = \sum_{d_i \in D} \frac{(jlabel_i - \overline{jlabel})(jclass_i - \overline{jclass})}{\sigma_{jLABEL} \sigma_{jCLASS}} \]

where \( jlabel_i = 1 \) iff the classifier assigns \( d_i \) to class \( c_j \)

\( jclass_i = 1 \) iff the true class of \( d_i \) is class \( c_j \)
Beware of terminological confusion in the literature!

- Some bioinformatics authors use “accuracy” incorrectly to refer to recall i.e. sensitivity or precision i.e. specificity
- In medical statistics, specificity sometimes refers to sensitivity for the negative class i.e. 
  \[
  \frac{TN_j}{TN_j + FP_j}
  \]
- Some authors use false alarm rate to refer to the probability that a positive prediction is incorrect i.e.
  \[
  \frac{FP_j}{FP_j + TP_j} = 1 - Precision_j
  \]

When you write

- provide the formula in terms of \(TP, TN, FP, FN\)

When you read

- check the formula in terms of \(TP, TN, FP, FN\)
Measuring Classifier Performance

- TP, FP, TN, FN provide the relevant information
- No single measure tells the whole story
- A classifier with 90% accuracy can be useless if 90 percent of the population does not have cancer and the 10% that do are misclassified by the classifier
- Use of multiple measures recommended
- Beware of terminological confusion!
Measuring Classifier Performance

Micro-averages

Micro averaging gives equal importance to each instance
→ Classes with large number of instances dominate

\[
\text{MicroAverage Precision} = \frac{\sum_j TP_j}{\sum_j TP_j + \sum_j FP_j}
\]

\[
\text{MicroAverage Recall} = \frac{\sum_j TP_j}{\sum_j TP_j + \sum_j FN_j}
\]

\[
\text{MicroAverage FalseAlarm} = 1 - \text{MicroAverage Precision}
\]

\[
\text{MicroAverage Accuracy} = \frac{\sum_j TP_j}{N}
\]

\[
\text{MicroAverage CC} = \frac{\left( \sum_j TP_j \times \sum_j TN_j \right) - \left( \sum_j FP_j \times \sum_j FN_j \right)}{\sqrt{\left( \sum_j TP_j + \sum_j FN_j \right) \left( \sum_j TP_j + \sum_j FP_j \right) \left( \sum_j TN_j + \sum j FN_j \right) \left( \sum_j TN_j + \sum j FP_j \right)}}
\]

Etc.
Measuring Classifier Performance

Macro-averaging

Macro averaging gives equal importance to each of the $M$ classes

\[
\text{MacroAverage Sensitivity} = \frac{1}{M} \sum_j \text{Sensitivity}_j
\]

\[
\text{MacroAverage Correlation Coeff} = \frac{1}{M} \sum_j \text{Correlation Coeff}_j
\]

\[
\text{MacroAverage Specificity} = \frac{1}{M} \sum_j \text{Specificity}_j
\]
Receiver Operating Characteristic (ROC) Curve

• We can often trade off recall versus precision – e.g., by adjusting classification threshold $\theta$ e.g.,

$$
label = c_j \text { if } \frac{P(c_j | X)}{P(\neg c_j | X)} > \theta
$$

• ROC curve is a plot of Sensitivity against False Alarm Rate which characterizes this tradeoff for a given classifier
Receiver Operating Characteristic (ROC) Curve

\[ Sensitivity = \frac{TP}{TP + FN} \]

\[ False \text{ Alarm Rate} = \frac{FP}{FP + TN} \]

Perfect classifier
Measuring Performance of Classifiers – ROC curves

• ROC curves offer a more complete picture of the performance of the classifier as a function of the classification threshold
• A classifier $h$ is better than another classifier $g$ if $\text{ROC}(h)$ dominates the $\text{ROC}(g)$
• $\text{ROC}(h)$ dominates $\text{ROC}(g) \Rightarrow \text{AreaROC}(h) > \text{AreaROC}(g)$
Evaluating a Classifier

• How well can a classifier be expected to perform on novel data?

• We can estimate the performance (e.g., accuracy, sensitivity) of the classifier using a test data set (not used for training)

• How close is the estimated performance to the true performance?

References:
• Evaluation of discrete valued hypotheses – Chapter 5, Mitchell
• Empirical Methods for Artificial Intelligence, Cohen
Better yet, take Stat 430x
Estimating the performance of a classifier

The true error of a hypothesis $h$ with respect to a target function $f$ and an instance distribution $D$ is

$$\text{Error}_D(h) \equiv \Pr_{x \in D}[f(x) \neq h(x)]$$

The sample error of a hypothesis $h$ with respect to a target function $f$ and an instance distribution $D$ is

$$\text{Error}_S(h) \equiv \frac{1}{|S|} \sum_{x \in S} \delta(f(x) \neq h(x))$$

$\delta(a,b) = 1$ iff $a \neq b$; $\delta(a,b) = 0$ otherwise
Estimating classifier performance

 Domain($X$) = \{a, b, c, d\}

 $$D(X) = \begin{cases} 1 & 1 & 1 & 1 \\ 8 & 2 & 8 & 4 \end{cases}$$

<table>
<thead>
<tr>
<th>x</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h(x)$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$f(x)$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

$$error_D(h) = Pr_D[h(x) \neq f(x)] = D(X = a) + D(X = c)$$

$$= \frac{1}{8} + \frac{1}{8} = \frac{1}{4}$$
Evaluating the performance of a classifier

- Sample error estimated from training data is an *optimistic* estimate.
  \[
  \text{Bias} = E[\text{Error}_S(h)] - \text{Error}_D(h)
  \]

- For an *unbiased* estimate, \( h \) must be evaluated on an independent sample \( S \) (which is not the case if \( S \) is the training set!)
- Even when the estimate is unbiased, it can vary across samples!
- If \( h \) misclassifies 8 out of 100 samples
  \[
  \text{Error}_S(h) = \frac{8}{100} = 0.08
  \]

How close is the sample error to the true error?
How close is the *estimated* error to the *true* error?

- Choose a sample $S$ of size $n$ according to distribution $D$
- Measure $Error_S(h)$

$Error_S(h)$ is a random variable (outcome of a random experiment)

Given $Error_S(h)$, what can we conclude about $Error_D(h)$?

More generally, given the estimated performance of a hypothesis, what can we say about its actual performance?
Evaluation of a classifier with limited data

- There is extensive literature on how to estimate classifier performance from samples and how to assign confidence to estimates (See Mitchell, Chapter 5)
- **Holdout method** – use part of the data for training, and the rest for testing
- We may be unlucky – training data or test data may not be *representative*
- **Solution** – Run multiple experiments with disjoint training and test data sets in which each class is represented in roughly the same proportion as in the entire data set
Estimating the performance of the learned classifier

K-fold cross-validation

Partition the data (multi) set $S$ into $K$ equal parts $S_1 \ldots S_K$
with roughly the same class distribution as $S$.

$\text{Error}_c = 0$

For $i=1$ to $K$ do

\[
\begin{aligned}
\{ & S_{\text{Test}} \leftarrow S_i, \quad S_{\text{Train}} \leftarrow S - S_i; \\
& \alpha \leftarrow \text{Learn}(S_{\text{Train}}) \\
& \text{Error}_c \leftarrow \text{Error}_c + \text{Error}(\alpha, S_{\text{Test}}) \}
\end{aligned}
\]

\[
\text{Error} \leftarrow \left( \frac{\text{Error}_c}{K} \right); \quad \text{Output(Error)}
\]
Leave-one-out cross-validation

• $K$-fold cross validation with $K = n$ where $n$ is the total number of samples available
• $n$ experiments – using $n-1$ samples for training and the remaining sample for testing
• Leave-one-out cross-validation does not guarantee the same class distribution in training and test data!
  Extreme case: 50% class 1, 50% class 2
    Predict majority class label in the training data
    True error – 50%;
    Leave-one-out error estimate – 100%!!!!!
Estimating classifier performance

Recommended procedure

- Use $K$-fold cross-validation ($K=5$ or 10) for estimating performance estimates (accuracy, precision, recall, points on ROC curve, etc.) and 95% confidence intervals around the mean
- Compute mean values of performance estimates and standard deviations of performance estimates
- Report mean values of performance estimates and their standard deviations or 95% confidence intervals around the mean
- Be skeptical – repeat experiments several times with different random splits of data into $K$ folds!
Evaluating a hypothesis or a learning algorithm

How well can the decision tree be expected to perform on novel data?

We can estimate the performance (e.g., accuracy) of the decision tree using a test data set (not used for training)

How close is the estimated performance to the true performance?

Reference: Evaluation of discrete valued hypotheses – Chapter 5, Mitchell
Evaluating performance when we can afford to test on a large independent test set

The *true* error of a hypothesis $h$ with respect to a target function $f$ and an instance distribution $D$ is

$$\text{Error}_D(h) \equiv \Pr_{x \in D}[f(x) \neq h(x)]$$

The sample error of a hypothesis $h$ with respect to a target function $f$ and an instance distribution $D$ is

$$\text{Error}_S(h) \equiv \frac{1}{|S|} \sum_{x \in S} \delta(f(x) \neq h(x))$$

$\delta(a, b) = 1$ iff $a \neq b$; $\delta(a, b) = 0$ otherwise
Evaluating Classifier performance

\[ \text{Bias} = E[\text{Error}_S(h)] - \text{Error}_D(h) \]

Sample error estimated from training data is an *optimistic* estimate

For an *unbiased* estimate, \( h \) must be evaluated on an independent sample \( S \) (which is not the case if \( S \) is the training set!)

Even when the estimate is unbiased, it can *vary* across samples!

If \( h \) misclassifies 8 out of 100 samples \( \text{Error}_S(h) = \frac{8}{100} = 0.08 \)

How close is the *sample error* to the *true error*?
How close is estimated error to its true value?

Choose a sample $S$ of size $n$ according to distribution $D$

Measure $Error_S(h)$

$Error_S(h)$ is a random variable (outcome of a random experiment)

Given $Error_S(h)$, what can we conclude about $Error_D(h)$?

More generally, given the estimated performance of a hypothesis, what can we say about its actual performance?
How close is estimated accuracy to its true value?

**Question**: How close is $p$ (the true probability) to $\hat{p}$?

This problem is an instance of a well-studied problem in statistics – the problem of estimating the proportion of a population that exhibits some property, given the observed proportion over a random sample of the population. In our case, the property of interest is that $h$ correctly (or incorrectly) classifies a sample.

Testing $h$ on a single random sample $x$ drawn according to $D$ amounts to performing a random experiment which succeeds if $h$ correctly classifies $x$ and fails otherwise.
How close is estimated accuracy to its true value?

The output of a hypothesis whose true error is \( p \) as a binary random variable which corresponds to the outcome of a Bernoulli trial with a success rate \( p \) (the probability of correct prediction)

The number of successes \( r \) observed in \( N \) trials is a random variable \( Y \) which follows the Binomial distribution

\[
P(r) = \frac{n!}{r!(n-r)!} p^r (1 - p)^{n-r}
\]
**Error$_S(h)$ is a Random Variable**

Probability of observing $r$ misclassified examples in a sample of size $n$:

$$P(r) = \frac{n!}{r!(n-r)!} p^r (1-p)^{n-r}$$

Binomial distribution for $n = 40, p = 0.3$
Recall basic statistics

Consider a random experiment with discrete valued outcomes \( y_1, y_2, \ldots, y_M \)

The expected value of the corresponding random variable \( Y \) is

\[
E(Y) \equiv \sum_{i=1}^{M} y_i \Pr(Y = y_i)
\]

The variance of \( Y \) is

\[
Var(Y) \equiv E[(Y - E[Y])^2]
\]

The standard deviation of \( Y \) is

\[
\sigma_Y \equiv \sqrt{Var(Y)}
\]
How close is estimated accuracy to its true value?

The mean of a Bernoulli trial with success rate $p = p$

$\text{Variance} = p (1-p)$

If $N$ trials are taken from the same Bernoulli process, the observed success rate $\hat{p}$ has the same mean $p$

and variance $\frac{p (1 - p)}{N}$

For large $N$, the distribution of $\hat{p}$ follows a Gaussian distribution
Binomial Probability Distribution

\[ P(r) = \frac{n!}{r!(n-r)!} p^r (1 - p)^{n-r} \]

Probability \( P(r) \) of \( r \) heads in \( n \) coin flips, if \( p = Pr(\text{heads}) \)

- Expected, or mean value of \( X \), \( E[X] \), is:
  \[ E[X] \equiv \sum_{i=0}^{N} iP(i) = np \]

- Variance of \( X \) is:
  \[ Var(X) \equiv E[(X - E[X])^2] = np(1 - p) \]

- Standard deviation of \( X \), \( \sigma_X \), is:
  \[ \sigma_X \equiv \sqrt{E[(X - E[X])^2]} = \sqrt{np(1 - p)} \]
Estimators, Bias, Variance, Confidence Interval

\[
\sigma_{Error_S(h)} = \sqrt{\frac{p(1-p)}{n}}
\]

\[
Error_S(h) = \frac{r}{n}
\]

\[
Error_D(h) = p
\]

\[
\sigma_{Error_D(h)} = \sqrt{\frac{Error_D(h)(1-Error_D(h))}{n}}
\]

\[
\sigma_{Error_S(h)} \approx \sqrt{\frac{Error_S(h)(1-Error_S(h))}{n}}
\]

An \(N\)% confidence interval for some parameter \(p\) that is the interval which is expected with probability \(N\)% to contain \(p\)
Normal distribution approximates binomial

$Error_S(h)$ follows a Binomial distribution, with

- mean $\mu_{Error_S(h)} = Error_D(h)$
- standard deviation $\sigma_{Error_S(h)} = \sqrt{\frac{Error_D(h)(1-Error_D(h))}{n}}$

We can approximate this by a Normal distribution with the same mean and variance when $np(1-p) \geq 5$
Normal distribution

\[ p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \]

The probability that \( X \) will fall in the interval \((a, b)\) is given by \( \int_a^b p(x) \, dx \)

Expected, or mean value of \( X \) is given by \( E[X] = \mu \)

Variance of \( X \) is given by \( \text{Var}(X) = \sigma^2 \)

Standard deviation of \( X \) is given by \( \sigma_X = \sigma \)
How close is the estimated accuracy to its true value?

Let the probability that a Gaussian random variable X, with zero mean, takes a value between \(-z\) and \(z\),
\[
Pr[-z \leq X \leq z] = c
\]

Pr\( X \geq z \) = 5\% implies there is a 5\% chance that \(X\) lies more than 1.65 standard deviations from the mean, or
\[
Pr \left[ -1.65 \leq X \leq 1.65 \right] = 90\%
\]

<table>
<thead>
<tr>
<th>Pr( X \geq z )</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.001</td>
<td>3.09</td>
</tr>
<tr>
<td>0.005</td>
<td>2.58</td>
</tr>
<tr>
<td>0.01</td>
<td>2.33</td>
</tr>
<tr>
<td>0.05</td>
<td>1.65</td>
</tr>
<tr>
<td>0.10</td>
<td>1.28</td>
</tr>
</tbody>
</table>
How close is the estimated accuracy to its true value?

But $\hat{p}$ does not have zero mean and unit variance so we normalize to get

$$\Pr \left[ -z < \frac{\hat{p} - p}{\sqrt{\frac{p(1-p)}{n}}} < z \right] = c$$
How close is the estimated accuracy to its true value?

To find confidence limits:

Given a particular confidence figure $c$, use the table to find the $z$ corresponding to the probability $\frac{1}{2} (1-c)$. Use linear interpolation for values not in the table

\[
p = \left[ \hat{p} + \frac{z^2}{2n} \pm z \sqrt{\frac{\hat{p}}{n} - \frac{\hat{p}^2}{n} + \frac{z^2}{4n^2}} \right] \left[ 1 + \frac{z^2}{n} \right]^{-1}
\]
How close is the estimated accuracy to its true value?

Example

\[ \hat{p} = 0.75; \quad n = 1000; \quad c = 0.80; \quad z = 1.28 \]

Then with 80% confidence, we can say that the value of \( p \) lies in the interval \([0.733, 0.768]\)

Note: the normal distribution assumption is valid only for large \( n \) (i.e. \( np(1-p) \geq 5 \) or \( n > 30 \)) so estimates based on smaller values of \( n \) should be taken with a generous dose of salt
80% of area (probability) lies in $\mu \pm 1.28\sigma$

$N\%$ of area (probability) lies in $\mu \pm z_N\sigma$

<table>
<thead>
<tr>
<th>$N%$</th>
<th>50%</th>
<th>68%</th>
<th>80%</th>
<th>90%</th>
<th>95%</th>
<th>98%</th>
<th>99%</th>
</tr>
</thead>
<tbody>
<tr>
<td>$z_N$</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>
<td>2.58</td>
</tr>
</tbody>
</table>
Confidence intervals

If

$S$ contains $n$ examples, drawn independently of $h$ and each other
and $n \geq 30$ or $np(1-p) \geq 5$,

Then

With approximately N% probability, $Error_s(h)$ lies in interval

$$Error_D(h) \pm Z_N \sqrt{\frac{Error_D(h)(1-Error_D(h))}{n}}$$

equivalently, $Error_D(h)$ lies in interval

$$Errors_s(h) \pm Z_N \sqrt{\frac{Errors_s(h)(1-Errors_s(h))}{n}}$$

which is approximately

$$Errors_s(h) \pm Z_N \sqrt{\frac{Errors_s(h)(1-Errors_s(h))}{n}}$$
One sided confidence intervals

What is the probability that $\text{Error}_D (h)$ is at most $U$?

Symmetry of Gaussian distribution implies that confidence interval with $100(1-\alpha)\%$ confidence with lower bound $L$ and upper bound $U$ corresponds to a confidence interval with confidence $100\left(1-\frac{\alpha}{2}\right)\%$ and with upper bound $U$ but no lower bound (or vice versa)
General approach to deriving confidence intervals

1. Identify the population parameter $p$ to be estimated e.g., $Error_D(h)$
2. Define a suitable estimator $W$ – preferably unbiased, minimum variance
3. Determine the distribution $D_W$ obeyed by $W$, and the mean and variance of $W$
4. Determine the confidence interval by finding the thresholds $L$ and $U$ such that $N\%$ of the mass of the probability distribution $D_Y$ falls within the interval $[L,U]$. 
Central Limit Theorem Simplifies Confidence Interval Calculations

Consider a set of independent, identically distributed random variables $Y_1... Y_n$, all governed by an arbitrary probability distribution with mean $\mu$ and finite variance $\sigma^2$. Define the sample mean,

$$\bar{Y} \equiv \frac{1}{n} \sum_{i=1}^{n} Y_i$$

Central Limit Theorem As $n \rightarrow \infty$, the distribution governing $\bar{Y}$ approaches a Normal distribution, with mean $\mu$ and variance $\sigma^2/n$
Evaluation of a classifier with limited data

**Holdout method** – use part of the data for training, and the rest for testing

We may be unlucky – training data or test data may not be *representative*

**Solution** – Run multiple experiments with disjoint training and test data sets in which each class is represented in roughly the same proportion as in the entire data set
Estimating the performance of the learned classifier

K-fold cross-validation

Partition the data (multi) set $S$ into $K$ equal parts $S_1 \ldots S_K$ where each part has roughly the same class distribution as $S$.

$A = 0$

For $i=1$ to $K$ do

\[
\begin{align*}
S_{Train} & \leftarrow S - S_i; & S_{Test} & \leftarrow S_i \\
\alpha & \leftarrow \text{Learn}(S_{Train}) \\
A & \leftarrow A + \text{Accuracy}(\alpha, S_{Test})
\end{align*}
\]

Accuracy $\leftarrow A/K$; Output (Accuracy)
K-fold cross-validation

Recommended procedure for evaluating classifiers when data are limited

Use K-fold cross-validation ($K=5$ or $10$)

Better still, repeat K-fold cross-validation $R$ times and average the results
Difference in error between two hypotheses

We wish to estimate  

\[ d \equiv \text{Error}_D(h_1) - \text{Error}_D(h_2) \]

Suppose \( h_1 \) has been tested on a sample \( S_1 \) of size \( n_1 \) drawn according to \( D \) and \( h_2 \) has been tested on a sample \( S_2 \) of size \( n_2 \) drawn according to \( D \).

An unbiased estimator  

\[ \widehat{d} \equiv \text{Error}_{S_1}(h_1) - \text{Error}_{S_2}(h_2) \]

For large \( n_1 \) and large \( n_2 \) the corresponding error estimates follow Normal distribution.

Difference of two Normal distributions yields a normal distribution with variance equal to the sum of the variances of the individual distributions.
Difference between errors of two hypotheses

\[ d \equiv \text{Error}_D(h_1) - \text{Error}_D(h_2) \]

\[ \hat{d} \equiv \text{Errors}_{S_1}(h_1) - \text{Errors}_{S_2}(h_2) \]

\[ \sigma_{\hat{d}} \approx \sqrt{\frac{\text{Errors}_{S_1}(h_1)(1 - \text{Error}_{S_1}(h_1))}{n_1} + \frac{\text{Error}_{S_2}(h_2)(1 - \text{Error}_{S_2}(h_2))}{n_2}} \]

\[ \hat{d} \pm z_N \sqrt{\frac{\text{Error}_{S_1}(h_1)(1 - \text{Error}_{S_1}(h_1))}{n_1} + \frac{\text{Error}_{S_2}(h_2)(1 - \text{Error}_{S_2}(h_2))}{n_2}} \]

When \( S_1 = S_2 \), the variance of \( \hat{d} \) is smaller and the confidence interval correct but overly conservative.
Hypothesis testing

Is one hypothesis likely to be better than another?
What is the probability that $Error_D(h_1) > Error_D(h_2)$?

Suppose $Error_{S_1}(h_1) = 0.30; \ Error_{S_2}(h_2) = 0.20; \ \hat{d} = 0.10$

What is the probability that $d > 0$ given that $\hat{d} = 0.10$?

$Pr(d > 0 | \hat{d} = 0.10) = Pr(\hat{d} < \mu_{\hat{d}} + 0.10)$
Hypothesis testing

If \( n_1 = n_2 = 100, \sigma_{\hat{d}} \approx 0.061 \)

\[
\Pr(d > 0 \mid \hat{d} = 0.10) \approx \Pr(\hat{d} < \mu_{\hat{d}} + 1.64\sigma_{\hat{d}}) = 0.95
\]

We accept the hypothesis that

\[
Error_D(h_1) > Error_D(h_2)
\]

with 95% confidence

Equivalently, we reject the opposite hypothesis – the null hypothesis at a \((1-0.95) = 0.05\) level of significance
Comparing learning algorithms $L_A$ and $L_B$

Which learning algorithm is better at learning $f$?

**Unlimited data** –

Run $L_A$ and $L_B$ on *large* training set $S_{train}$ drawn according to $D$

Test the resulting hypotheses on a *large independent* test set $S_{test}$ drawn according to $D$

Estimate $\Pr[Error_D(L_A(S_{train})) > Error_D(L_B(S_{train}))]$ Using

$Error_{S_{test}}(L_A(S_{train}))$ and $Error_{S_{test}}(L_B(S_{train}))$
Comparing learning algorithms $L_A$ and $L_B$

Estimate the expected value of the difference in errors of $L_A$ and $L_B$ where expectation is taken over training sets $S_{Train}$ drawn according to $D$

$$E_{S_{Train} \subset D} [Error_D(L_A(S_{Train})) - Error_D(L_B(S_{Train}))]$$

We have a limited data set $S$ drawn from an unknown $D$ !!
Comparing learning algorithms $L_A$ and $L_B$

Limited data – Paired t-test

Run $L_A$ and $L_B$ on large training set $S_{Train}$ drawn according to $D$

Test the resulting hypotheses on a large independent test set $S_{Test}$ drawn according to $D$

Estimate

$$\text{Pr}[\text{Error}_D(L_A(S_{Train})) > \text{Error}_D(L_B(S_{Train}))]$$

using

$$\text{Error}_{S_{Test}}(L_A(S_{Train})) \text{ and } \text{Error}_{S_{Test}}(L_B(S_{Train}))$$
Comparing learning algorithms \( L_A \) and \( L_B \)

**Paired t-test**

Partition \( S \) into \( k \) disjoint test sets \( T_1, T_2, \ldots, T_k \) of equal size

For \( i \) from 1 to \( k \) do {

\[
S_{\text{Test}} \leftarrow T_i ; \quad S_{\text{Train}} \leftarrow S - T_i
\]

\[
\delta_i \leftarrow Error_{S_{\text{Test}}} (L_A(S_{\text{Train}})) - Error_{S_{\text{Test}}} (L_B(S_{\text{Train}}))
\]

}

Return \( \bar{\delta} \equiv \frac{1}{k} \sum_{i=1}^{k} \delta_i \)
Comparing learning algorithms $L_A$ and $L_B$

For large test sets, each $\delta_i$ has Normal distribution
$\overline{\delta}$ has Normal distribution if $\delta_i$ are independent
Can we estimate confidence interval for $\overline{\delta}$ as before?

$\delta_i$ are not exactly independent because of sampling from $S$ as opposed to the distribution $D$ (but we will pretend that they are)

We don’t know the standard deviation of this distribution. So we estimate it from sample ..But when the estimated variance is used, the distribution is no longer Normal unless $K$ is large (which typically it is not)
Comparing learning algorithms $L_A$ and $L_B$

Approximate $N\%$ confidence interval for

$$E_{S_{Train} \subseteq S} \left[ Error_D(L_A(S_{Train})) - Error_D(L_B(S_{Train})) \right]$$

is given by

$$\bar{\delta} \pm t_{N,k-1} \psi_{\bar{\delta}}$$

where

$$\psi_{\bar{\delta}} = \sqrt{\frac{1}{k(k-1)} \sum_{i=1}^{k} (\delta_i - \bar{\delta})^2}$$

is the estimate of standard deviation of the $t$ distribution governing $Z_N$ and $t_{N,K-1}$ plays a role analogous to that of $\bar{\delta}$.

As $K \to \infty$, $t_{N,K-1} \to Z_N$ and $\psi_{\bar{\delta}} \to \sigma_{\bar{\delta}}$
Performance evaluation summary

Rigorous statistical evaluation is extremely important in experimental computer science in general and machine learning in particular.

How good is a learned hypothesis?

Is one hypothesis better than another?

Is one learning algorithm better than another on a particular learning task? (No learning algorithm outperforms all others on all tasks – No free lunch theorem)

Different procedures for evaluation are appropriate under different conditions (large versus limited versus small sample) – Important to know when to use which evaluation method and be aware of pathological behavior (tendency to grossly overestimate or underestimate the target value under specific conditions)