Model Selection and Assessment

CS4780/5780 – Machine Learning
Fall 2013

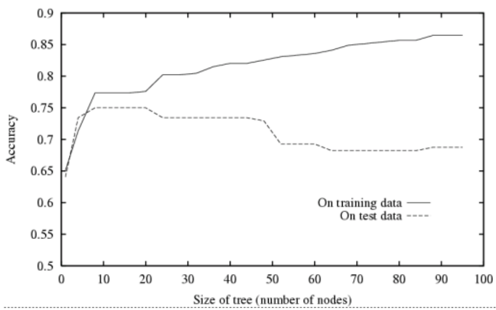
Thorsten Joachims
Cornell University

Reading:
Mitchell Chapter 5
Dietterich, T. G., (1998). Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. Neural Computation, 10 (7) 1895-1924.
(http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.37.3325)

Outline

- Model Selection
  - Controlling overfitting in decision trees
  - Train, validation, test
  - K-fold cross validation
- Evaluation
  - What is the true error of classification rule h?
  - Is rule h_1 more accurate than h_2?
  - Is learning algorithm A1 better than A2?

Overfitting



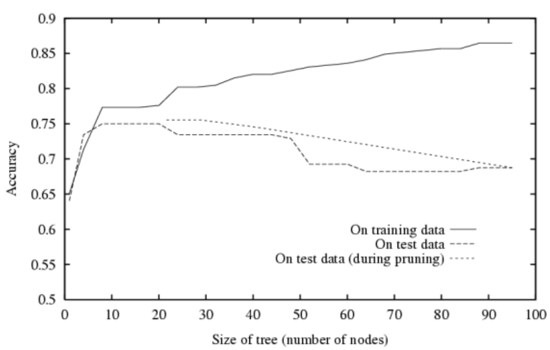
- Note: Accuracy = 1.0-Error

[Mitchell]

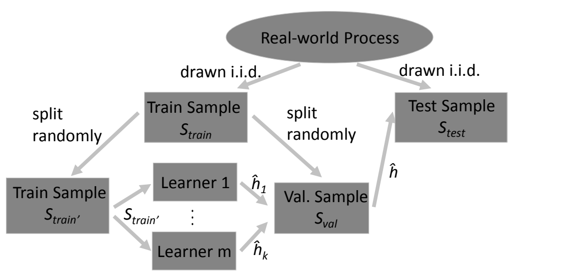
Controlling Overfitting in Decision Trees

- Early Stopping: Stop growing the tree and introduce leaf when splitting no longer "reliable".
  - Restrict size of tree (e.g., number of nodes, depth)
  - Minimum number of examples in node
  - Threshold on splitting criterion
- Post Pruning: Grow full tree, then simplify.
  - Reduced-error tree pruning
  - Rule post-pruning

Reduced-Error Pruning



Model Selection



- **Training:** Run learning algorithm m times (e.g. different parameters).
- **Validation Error:** Errors $Err_{S_{val}}(\hat{h}_i)$ is an estimates of $Err_P(\hat{h}_i)$ for each h_i.
- **Selection:** Use h_i with min $Err_{S_{val}}(\hat{h}_i)$ for prediction on test examples.

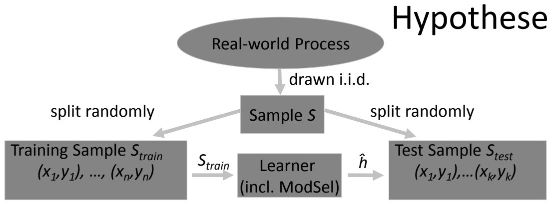
K-fold Cross Validation

- Given
  - Sample of labeled instances S
  - Learning Algorithms A
- Compute
  - Randomly partition S into k equally sized subsets $S_1 \dots S_k$
  - For i from 1 to k
    - Train A on $S_1 \dots S_{i-1} S_{i+1} \dots S_k$ and get $\hat{h}$.
    - Apply $\hat{h}$ to S_i and compute $Err_{S_i}(\hat{h})$.
- Estimate
  - Average $Err_{S_i}(\hat{h})$ is estimate of average prediction error of rules produced by A, namely $E_S(Err_P(A(S_{train})))$

Text Classification Example: "Corporate Acquisitions" Results

- Unpruned Tree (ID3 Algorithm):
  - Size: 437 nodes Training Error: 0.0% Test Error: 11.0%
- Early Stopping Tree (ID3 Algorithm):
  - Size: 299 nodes Training Error: 2.6% Test Error: 9.8%
- Reduced-Error Tree Pruning (C4.5 Algorithm):
  - Size: 167 nodes Training Error: 4.0% Test Error: 10.8%
- Rule Post-Pruning (C4.5 Algorithm):
  - Size: 164 tests Training Error: 3.1% Test Error: 10.3%
  - Examples of rules
    - IF vs = 1 THEN - [99.4%]
    - IF vs = 0 & export = 0 & takeover = 1 THEN + [93.6%]

Evaluating Learned Hypotheses

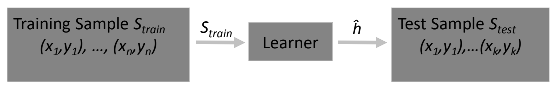


- Goal: Find h with small prediction error $Err_P(h)$ over $P(X,Y)$.
- Question: How good is $Err_P(\hat{h})$ of $\hat{h}$ found on training sample S_{train}.

- **Training Error:** Error $Err_{S_{train}}(\hat{h})$ on training sample.
- **Test Error:** Error $Err_{S_{test}}(\hat{h})$ is an estimate of $Err_P(\hat{h})$.

What is the True Error of a Hypothesis?

- Given
  - Sample of labeled instances S
  - Learning Algorithm A
- Setup
  - Partition S randomly into S_{train} (70%) and S_{test} (30%)
  - Train learning algorithm A on Strain, result is $\hat{h}$.
  - Apply $\hat{h}$ to S_{test} and compare predictions against true labels.
- Test
  - Error on test sample $Err_{S_{test}}(\hat{h})$ is estimate of true error $Err_P(\hat{h})$.
  - Compute confidence interval.



Binomial Distribution

- The probability of observing x heads in a sample of n independent coin tosses, where in each toss the probability of heads is p, is

$$P(X = x|p,n) = \frac{n!}{x!(n-x)!} p^x (1-p)^{n-x}$$

- Normal approximation: For $np(1-p)>=5$ the binomial can be approximated by the normal distribution with
  - Expected value: $E(X)=np$ Variance: $Var(X)=np(1-p)$
  - With probability δ, the observation x falls in the interval

$$E(X) \pm z_\delta \sqrt{Var(X)}$$

δ	50%	68%	80%	90%	95%	98%	99%
z_δ	0.67	1.00	1.28	1.64	1.96	2.33	2.58

Text Classification Example: Results

- Data
  - Training Sample: 2000 examples
  - Test Sample: 600 examples
- Unpruned Tree:
  - Size: 437 nodes Training Error: 0.0% Test Error: 11.0%
- Early Stopping Tree:
  - Size: 299 nodes Training Error: 2.6% Test Error: 9.8%
- Post-Pruned Tree:
  - Size: 167 nodes Training Error: 4.0% Test Error: 10.8%
- Rule Post-Pruning:
  - Size: 164 tests Training Error: 3.1% Test Error: 10.3%

Is Rule h_1 More Accurate than h_2? (Same Test Sample)

- Given
  - Sample of labeled instances *S*
  - Learning Algorithms A_1 and A_2
- Setup
  - Partition *S* randomly into S_{train} (70%) and S_{test} (30%)
  - Train learning algorithms A_1 *and* A_2 on S_{train}, result are $\hat{h}_1$ and $\hat{h}_2$.
  - Apply $\hat{h}_1$ and $\hat{h}_2$ to S_{val} and compute $Err_{S_{test}}(\hat{h}_1)$ and $Err_{S_{test}}(\hat{h}_2)$.
- Test
  - Decide, if $Err_P(\hat{h}_1) \neq Err_P(\hat{h}_2)$?
  - Null Hypothesis: $Err_{S_{test}}(\hat{h}_1)$ and $Err_{S_{test}}(\hat{h}_2)$ come from binomial distributions with same *p*.

    → Binomial Sign Test (McNemar's Test)

Is Rule h_1 More Accurate than h_2? (Different Test Samples)

- Given
  - Samples of labeled instances S_1 and S_2
  - Learning Algorithms A_1 and A_2
- Setup
  - Partition S_1 randomly into S_{train1} (70%) and S_{test1} (30%)
    Partition S_2 randomly into S_{train2} (70%) and S_{test2} (30%)
  - Train learning algorithm A_1 on S_{train1} and A_2 on S_{train2}, result are $\hat{h}_1$ and $\hat{h}_2$.
  - Apply $\hat{h}_1$ to S_{test1} and $\hat{h}_2$ to S_{test2} and get $Err_{S_{test1}}(\hat{h}_1)$ and $Err_{S_{test2}}(\hat{h}_2)$.
- Test
  - Decide, if $Err_P(\hat{h}_1) \neq Err_P(\hat{h}_2)$?
  - Null Hypothesis: $Err_{S_{test1}}(\hat{h}_1)$ and $Err_{S_{test2}}(\hat{h}_2)$ come from binomial distributions with same *p*.

    → t-Test (z-Test) [→ see Mitchell book]

Is Learning Algorithm A_1 better than A_2?

- Given
  - *k* samples $S_1 \dots S_k$ of labeled instances, all i.i.d. from P(X,Y).
  - Learning Algorithms A_1 and A_2
- Setup
  - For *i* from *1* to *k*
    - Partition S_i randomly into S_{train} (70%) and S_{test} (30%)
    - Train learning algorithms A_1 *and* A_2 on S_{train}, result are $\hat{h}_1$ and $\hat{h}_2$.
    - Apply $\hat{h}_1$ and $\hat{h}_2$ to S_{test} and compute $Err_{S_{test}}(\hat{h}_1)$ and $Err_{S_{test}}(\hat{h}_2)$.
- Test
  - Decide, if $E_S(Err_P(A_1(S_{train}))) \neq E_S(Err_P(A_2(S_{train})))$?
  - Null Hypothesis: $Err_{S_{test}}(A_1(S_{train}))$ and $Err_{S_{test}}(A_2(S_{train}))$ come from same distribution over samples *S*.

    → t-Test (z-Test) or Wilcoxon Signed-Rank Test
    [→ see Mitchell book]

Approximation via K-fold Cross Validation

- Given
  - Sample of labeled instances *S*
  - Learning Algorithms A_1 and A_2
- Compute
  - Randomly partition S into k equally sized subsets $S_1 \dots S_k$
  - For *i* from *1* to *k*
    - Train A_1 and A_2 on $S_1 \dots S_{i-1} S_{i+1} \dots S_k$ and get $\hat{h}_1$ and $\hat{h}_2$.
    - Apply $\hat{h}_1$ and $\hat{h}_2$ to S_i and compute $Err_{S_i}(\hat{h}_1)$ and $Err_{S_i}(\hat{h}_2)$.
- Estimate
  - Average $Err_{S_i}(\hat{h}_1)$ is estimate of $E_S(Err_P(A_1(S_{train})))$
  - Average $Err_{S_i}(\hat{h}_2)$ is estimate of $E_S(Err_P(A_2(S_{train})))$
  - Count how often $Err_{S_i}(\hat{h}_1) > Err_{S_i}(\hat{h}_2)$ and $Err_{S_i}(\hat{h}_1) < Err_{S_i}(\hat{h}_2)$