Evaluating hypotheses

Issues

- How predictive is the model we learned.
- Error on the training data is not a good indicator of performance on future data. This error may be easily reduced to 0, however we need generalizations of our data.
- Solution: split data into training and test set
- However, to create a good model we need a large training set and to evaluate its performance we need a large test set as well. So, what we really need is lots of preclassified data.
- We need also statistical reliability of estimated differences in performance (significance tests)
- Performance measures:
  - Number of correct classifications
  - Accuracy of probability estimates
  - Costs assigned to different types of errors
- To improve the classifier accuracy we may combine multiple models
- We can measure the classifier predictiveness by applying Minimum Description Length Principle (MDL).

Error rate

- Natural performance measure for classification problems.
- Success: instance class is predicted correctly.
- Error: instance class is predicted incorrectly.
- (Observed) Error rate: proportion of errors made over the whole set of instances tested.
- Resubstitution error: error on the training set (too optimistic measure!).
- True error rate: the actual error rate on the whole population (usually estimated because in most cases the whole population is not available).

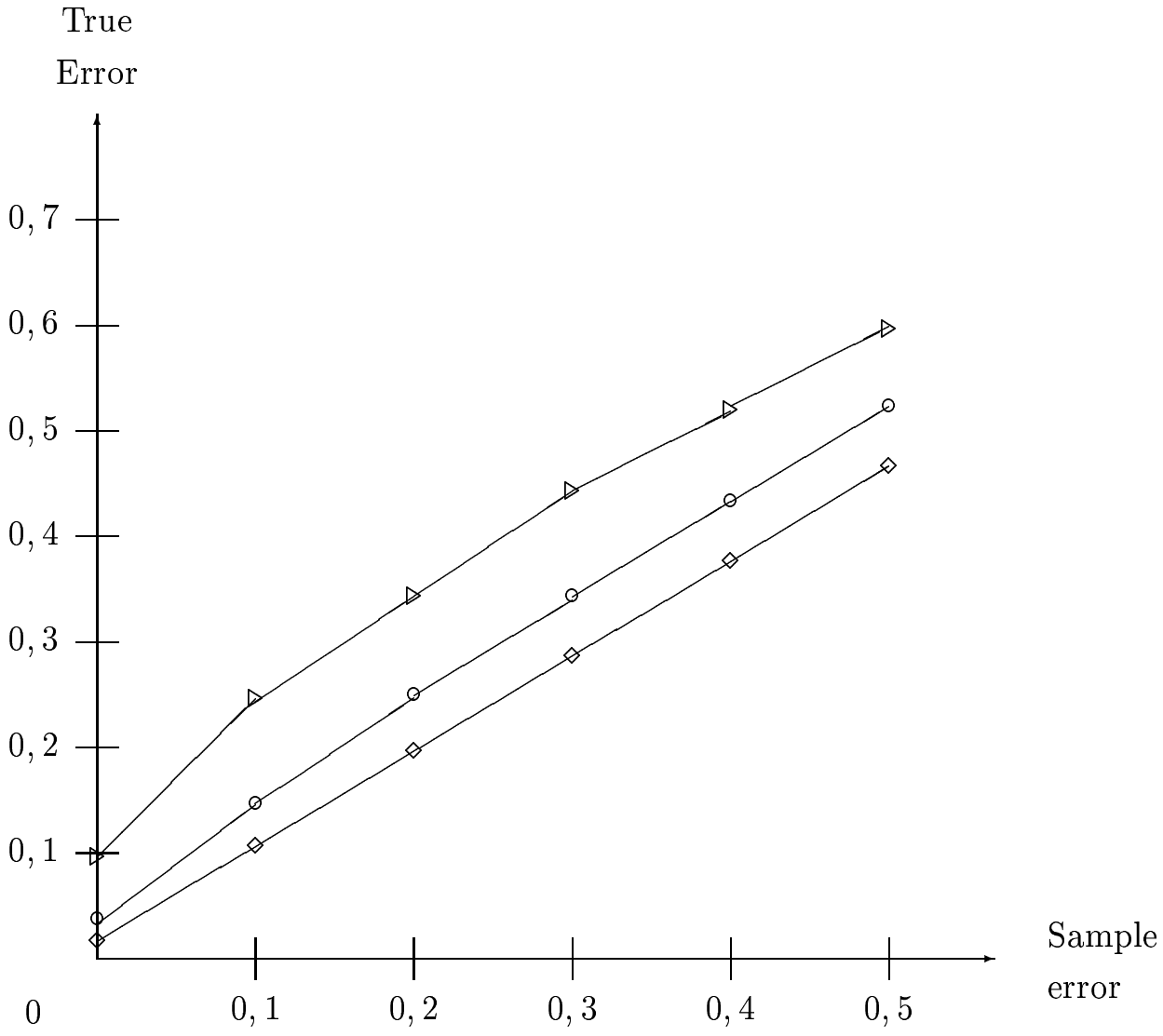
Testing

- Test set: a set of instances that have not been used in the training process.
- Assumption: training set and test set are both representative samples of the same larger population.
- Some classifiers work in two steps (often iteratively):
  - Step 1: learning the basic structure.
  - Step 2: optimizing parameters that are used in learning.
- The test set must not be used in any way in the training process (even for parameter tuning, in Step 2).
- There are may be another independent set of instances to be used for optimizing parameters (validation set). That is, we split the set of known data into three: training set, validation set and test set.
- Holdout procedure: the method of splitting data into training and test set. Dilemma: the balance between training and test set.

Predicting performance (true success/error rate)

- Testing just estimates the probability of success on unknown data (data, not used in both training and testing).
- How good is this estimate? (What is the true success/error rate?) We need confidence intervals (a kind of statistical reasoning) to predict this.
- Assume that success and error are two possible outcomes of a statistical experiment (normally distributed random variable).
- Bernoulli process: We have made N experiments and got S successes. Then, the observed success rate is P=S/N. What is the true success rate?
- Example:
  - N=100, S=75. Then with confidence 80% $P \in [0.691, 0.801]$.
  - N=1000, S=750. Then with confidence 80% $P \in [0.732, 0.767]$.

Predicting true error rate



Sample size: ▷ – 50, ○ – 250, ◇ – 1000

Confidence intervals

If

- The sample S contains n examples, drawn independently of the hypothesis h and each other.
- $n \geq 30$
- The sample error of h is $error_S(h)$

Then

- With probability $N\%$ the true error $error_D$ is in the interval:

$$error_S(h) \pm z_N \sqrt{\frac{error_S(h)(1 - error_S(h))}{n}}$$

- Where

$N\%$	50%	68%	80%	90%	95%	98%	99%
z_N	0.67	1.00	1.28	1.64	1.96	2.33	2.58

Estimating classifier accuracy

- Holdout
  - Reserve a certain amount for testing and use the remainder for training (usually 1/3 for testing, 2/3 for training).
  - Problem: the samples might not be representative. For example, some classes might be represented with very few instance or even with no instances at all.
  - Solution: stratification – sampling for training and testing within classes. This ensures that each class is represented with approximately equal proportions in both subsets
- Repeated holdout. Success/error estimate can be made more reliable by repeating the process with different subsamples.
  - In each iteration, a certain proportion is randomly selected for training (possibly with stratification)
  - The error rates on the different iterations are averaged to yield an overall error rate.
  - Problem: the different test sets may overlap. Can we prevent overlapping?

Cross-validation (CV). Avoids overlapping test sets

- k-fold cross-validation
  - First step: data is split into k subsets of equal size (usually by random sampling).
  - Second step: each subset in turn is used for testing and the remainder for training.
  - The error estimates are averaged to yield an overall error estimate.
- Stratified cross-validation: subsets are stratified before the cross-validation is performed.
- Stratified ten-fold cross-validation
  - Standard method for evaluation. Extensive experiments have shown that this is the best choice to get an accurate estimate. There is also some theoretical evidence for this.
  - Stratification reduces the estimate's variance.
  - Repeated stratified cross-validation is even better. Ten-fold cross-validation is repeated ten times and results are averaged.

Leave-one-out cross-validation (LOO CV)

- LOO CV is a n-fold cross-validation, where n is the number of training instances. That is, n classifiers are built for all possible (n-1)-element subsets of the training set and then tested on the remaining single instance.
- LOO CV makes maximum use of the data.
- No random subsampling is involved.
- Problems
  - LOO CV is very computationally expensive.
  - Stratification is not possible. Actually this method guarantees a non- stratified sample (there is only one instance in the test set).
  - Worst case example: assume a completely random dataset with two classes each represented by 50% of the instances. The best classifier for this data is the majority predictor. LOO CV will predict 100% error (!) rate for this classifier (explain why?).

Counting the cost

- Different types of classification errors often incur different costs.
- Example: predict cancer. Compare the cost of predicting "no" when the actual classification is "yes" and predicting "yes" when the actual classification is "no". Obviously the first error is much more costly.
- Confusion matrix:

Actual \ Predicted	yes	no
yes	True Positive (TP)	False Negative (FN)
no	False Positive (FP)	True Negative (TN)

- Total error = (FP+FN)/(TP+FP+TN+FN)
- Recall - precision (information retrieval):
  - Precision (retrieved relevant / total retrieved) = TP / (TP+FP)
  - Recall (retrieved relevant / total relevant) = TP / (TP + FN)

Bayesian evaluation

- E - data
- $H = \{H_1, H_2, ..., H_n\}$ - hypotheses
- $H_{best} = argmax_i \ P(H_i|E)$
- Bayes theorem:

$$P(H_i|E) = \frac{P(H_i)P(E|H_i)}{\sum_{i=1}^{n} P(H_i)P(E|H_i)}$$

Occam's Razor

The best hypotesis is the simplest (shortest) one. Example (grammars):

- Data:
  $E^+ = \{0, 000, 00000, 000000000\}$
  $E^- = \{\varepsilon, 00, 0000, 000000\}$
- Hypotheses:
  $G_1 : S \to 0|000|00000|000000000$
  $G_2 : S \to 00S|0$
- Best hypothesis: G_2

Formal criteria for simplicity

- Bayesian approach: need of large volume of experimental results (statistics).
- Algorithmic (Kolmogorov) complexity of an object (bit string): the length of the shortest program of Universal Turing Machine, that generates the string. Problems: computational complexity.
- Information-based approches: Minimum Description Length Principle (MDL). Most often used in practice.

Minimum Description Length Principle (MDL)

- Bayes Theorem:

$$P(H_i|E) = \frac{P(H_i)P(E|H_i)}{\sum_{i=1}^{n} P(H_i)P(E|H_i)}$$

- Take a $-\log$ of both sides of Bayes:

$$-\log_2 P(H_i|E) = -\log_2 P(H_i) - \log_2 P(E|H_i) + C$$

- Information in message A (minimal length of A in bits):
  $\log_2 P(A) = I(A) = L(A)$
- Then: $L(H|E) = L(H) + L(E|H)$
- MDL: The hypothesis must reduce the information needed to encode the data, i.e.

$$L(E) > L(H) + L(E|H)$$

- The best hypothesis must maximize *information compression*:

$$L(E) - L(H) - L(E|H)$$

Measuring information compression ($E = E^+ \cup E^-$)

$L(H) = ?$

- $H = R_1 \vee R_2 \vee ... \vee R_m$
- R_i: if $t_1 \wedge t_2 \wedge ... \wedge t_{k_i}$ then $+$
- ts - total number of tests t_i (attribute value pairs)
- Then:

$$L(R_i) = -\log_2 \frac{1}{\binom{ts}{k_i}} = \log_2 \binom{ts}{k_i}$$

$$L(H) = \sum_{i=1}^{m} \log_2 \binom{ts}{k_i}$$

$L(E) = ?$

- Apply the same technique for E (both E and H use the same representation).

$L(E|H) =$? **(encoding exceptions)**

- Find the confusion matrix:

Actual \ Predicted by H	$+$	$-$
$+$	tp	fn
$-$	fp	tn

- H covers $tp + fp$ examples: tp from E^+ and fp from E^-
  H does not cover $tn + fn$: tn from E^- and fn from E^+
- fp examples out of $(tp + fp)$ examples can be chosen in $\binom{tp+fp}{fp}$ different ways.
- fn examples out of $(tn + fn)$ examples can be chosen in $\binom{tn+fn}{fn}$ different ways.
- Then the code length of the exceptions (fp and fn), $L(E|H)$ is:

$$L(E|H) = \log_2 \binom{tp+fp}{fp} + \log_2 \binom{tn+fn}{fn}$$

Encoding entropy

- For each rule R_i in H calculate:

$$e_i = -\frac{p_i}{n_i} * \log_2 \frac{p_i}{n_i} - \frac{n_i - p_i}{n_i} * \log_2 \frac{n_i - p_i}{n_i},$$

- Where p_i is the number of positive examples covered by R_i, and n_i is the total number of examples covered by R_i.
- Then

$$L(H|E) = \sum_i |R_i| + e_i * n_i.$$

- Where $|R_i|$ is the number of tests (attribute-value pairs) in rule R_i.
- Since different encodings are used for $L(H)$ and $L(E|H)$, the above formula can be used to compare hypotheses only, but not as a measure of compression.

Bayesian Belief Networks (BN)

- Define joint conditional probabilities.
- Combine Bayesian reasoning with causal relationships between attributes.
- Also known as bayesian networks, probabilistic networks.

BN are defined by

- Directed acyclic graph, with nodes representing random variables and links – probabilistic dependence.
- Conditional probability tables (CPT) for each variable (node): specifies all $P(X|partents(X))$, i.e. the probability of each value of X, given every possible combination of values for its parents.

Reasoning with BN

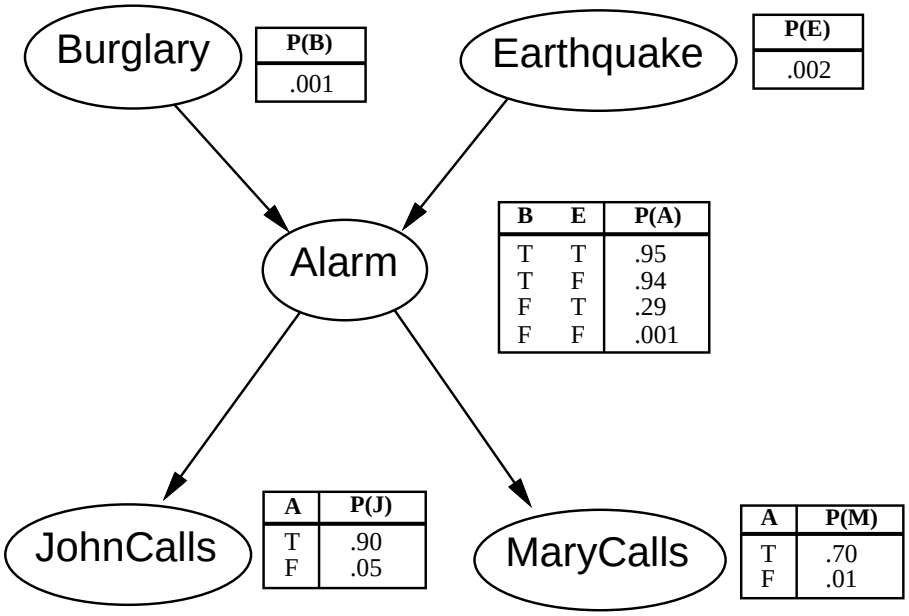
- Given the probabilities at some nodes (inputs) BN calculates the probabilities in other nodes (outputs).
- Classification: inputs - attribute values, output - class value probability.
- There are mechanisms for training BN from examples, given variables and network structure, i.e. creating CPT's.

Example

- I'm at work, neighbor John calls to say my alarm is ringing, but neighbor Mary doesn't call. Sometimes it's set off by minor earthquakes. Is there a burglar?
- Variables: Burglary (B), Earthquake (E), Alarm (A), JohnCalls (J), MaryCalls (M)
- Structure:
  Burglary $\rightarrow$ Alarm
  Earthquake $\rightarrow$ Alarm
  Alarm $\rightarrow$ JohnCalls
  Alarm $\rightarrow$ MaryCalls
- Conditional Probability tables (CPT):

B	E	$P(A)$	$P(\neg A)$
T	T	0.95	0.05
T	F	0.94	0.06
F	T	0.29	0.71
F	F	0.001	0.999

Network topology reflects causal knowledge



(for short probability of false is not given, rows must sum to 1)

Calculation of joint probabilities

$P(J, M, A, \neg B, \neg E) =$
$P(J|A) * P(M|A) * P(A|\neg B \wedge \neg E) * P(\neg B) * P(\neg E) =$
$0.9 * 0.7 * 0.001 * 0.999 * 0.998 = 0.000628.$

Reasoning with BN

- Using complete joints distribution or other more efficient methods.
- Diagnostic (from effect to cause):
  - $P(B|J) = 0.016$
  - $P(B|J \wedge M) = 0.28$
  - $P(A|J \wedge M) = 0.76$
- Predictive (from cause to effect):
  - $P(J|B) = 0.85$
  - $P(M|B) = 0.66$
- Other
  - intercausal $P(B|A)$
  - mixed $P(A|J \wedge \neg E)$