Refinements and Evaluation

- ROC curves
- Hold out method
- Cross-validation
- The bootstrap
- Bagging
- Boosting

Optimal performance: ROC

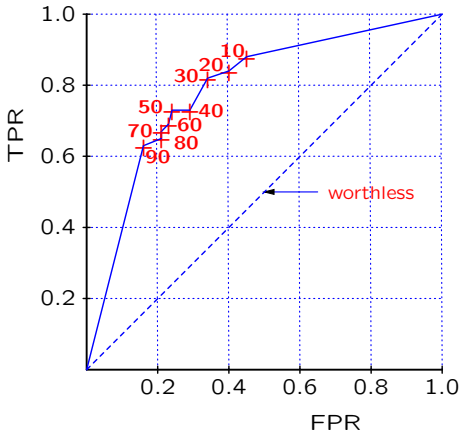
- Output of probabilistic classifier:

$$c_{max} = \arg\max_{C} P(C \mid \mathcal{E})$$

may not yield the best performance

- Alternative: Receiver Operating Characteristic (ROC): determine threshold d, such that

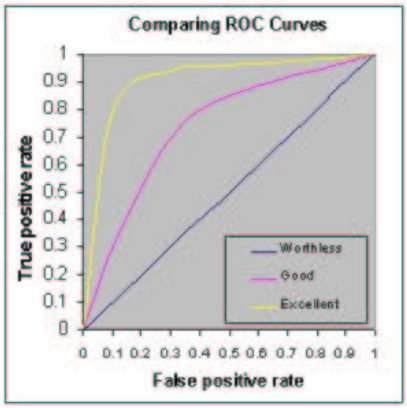
$$C = \begin{cases} c & \text{if } P(c \mid \mathcal{E}) \geq d \\ \neg c & \text{otherwise} \end{cases}$$



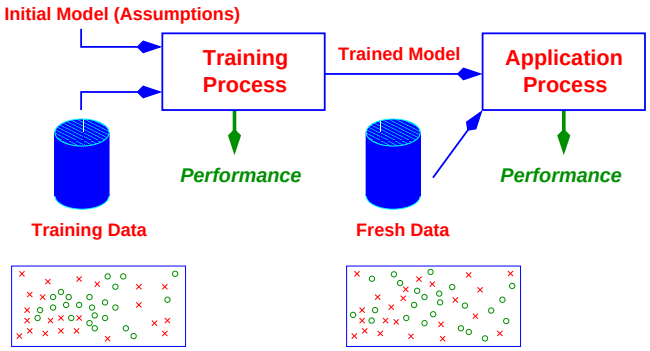
Area under the ROC curve

When comparing various techniques:

- actual performance for particular thresholds (cut-off points) may vary
- area under the ROC curve $A_f = \int_0^1 f(x)\mathrm{d}x$ offers good measure for comparison, with f relationship between FPR and TPR for classifier

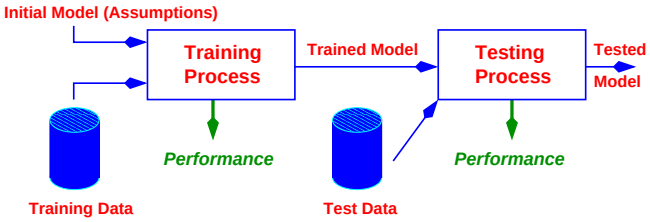


Evaluation problem sketch



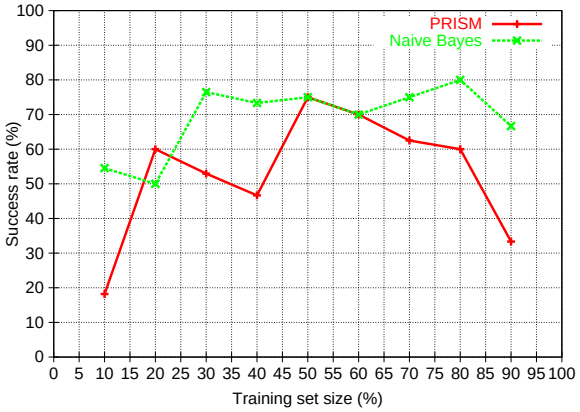
- Fresh data differs from training set (which affects performance)
- Overfitting
- Bias-variance decomposition

Solution 1: holdout method



- Test set and training set disjoint
- Select more (66%?) training instances than test instances
- Holdout: test set
- Problem: what if the dataset is small?

Size of holdout

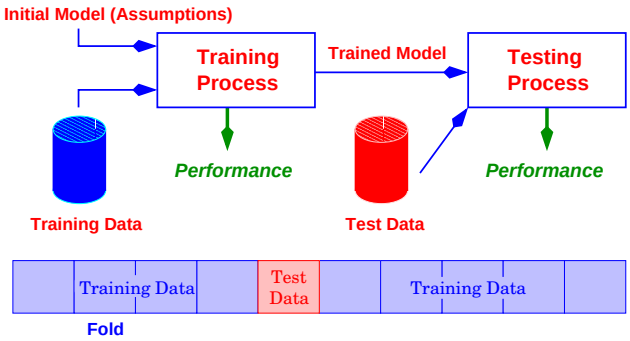


Test set (%) + Training set (%) = 100%

Two methods compared:

- PRISM classification-rule learning algorithm
- Naive Bayesian classifier

Solution 2: cross-validation

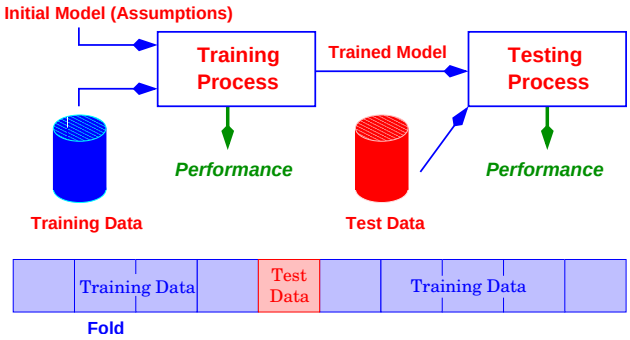


K-fold cross-validation: split up dataset D into K partitions (called folds)

for $k = 1, \ldots, K$:

1. Train model using $K-1$ of the folds (exclude fold k)
2. Evaluate using the remaining fold k (hold-out)
3. Gather performance results

Performance results

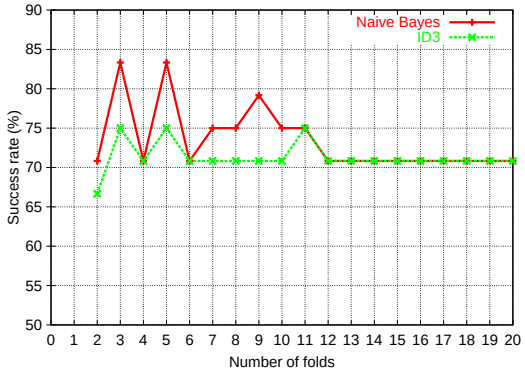


K-fold cross-validation:

- Dataset D, with $N = |D|$
- $\hat{f}^{-k}$ trained model based on dataset D after removal of fold k
- Indexing function $\kappa : \{1, \ldots, N\} \to \{1, \ldots, K\}$ which associates fold $\kappa(i)$ to instance i
- Success rate with L 0-1 loss function:

$$\sigma = \frac{1}{N} \sum_{i=1}^{N} L(c_i, \hat{f}^{-\kappa(i)}(\mathbf{x}'_i))$$

How many folds?



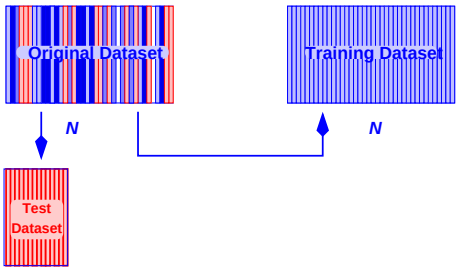
- Common choices: 5 and 10 (5-fold and 10-fold cross-validation)
- Leave-one-out method:
  - $N = K$ (N-fold cross-validation)
  - almost all available data is used
  - easy to implement, no random sampling
  - computationally expensive

Stratification

- Folds not necessarily representative for whole dataset
- Solution:
  - check whether folds are representative
  - if not: select new folds (at random)
- Further elimination of effect of random selection of folds: run cross-validation repeatedly

Result: M-time stratified K-fold cross-validation

Solution 3: bootstrapping



- Yields estimation of uncertainty in learning
- Basic idea:
  1. sample $N = |D|$ times from dataset with replacement: result training set of size N
  2. instances not selected are taken as test set
- Expected size of test set?

Bootstrap: test-set size

- Probability that instance i is not selected:

$$P(i) = 1 - \frac{1}{N}$$

- After taking N samples: $\left(1 - \frac{1}{N}\right)^N$. Note that:
  - $e^x = \sum_{k=0}^{N} \frac{x^k}{k!} + R_N(x)$, i.e.

$$e^{-1} = \sum_{k=0}^{N} (-1)^k \frac{1}{k!} + R_N$$

  - Newton's binomial theorem:

$$\begin{aligned}\left(1 - \frac{1}{N}\right)^N &= \sum_{k=0}^{N} \binom{N}{k} \left(\frac{-1}{N}\right)^k \\ &= \sum_{k=0}^{N} (-1)^k \frac{N(N-1)\cdots(N-k+1)}{k!N^k} \\ &\approx \sum_{k=0}^{N} (-1)^k \frac{1}{k!}\end{aligned}$$

$\Rightarrow$ the size of the test set: $e^{-1} \cdot N \approx 0.37N$

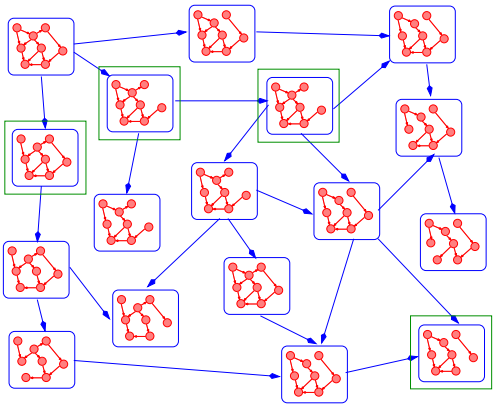
Bootstrap: practical

- Effect of random selection of test set: result may not be representative
- Solution:
  - adjust error rate:

    $$\epsilon = 0.63 \cdot \epsilon_{test} + 0.37 \cdot \epsilon_{training}$$

  - repeat the process a number of times, and compute average outcome
- Usage: in the context of learning multiple models (bagging)

Committee of experts



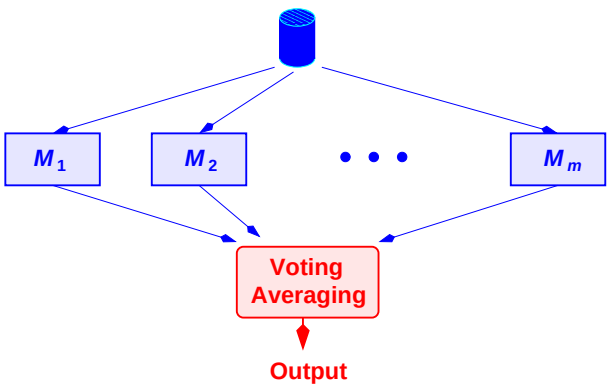
- Hypothesis space
- No single hypothesis fits all possible (or all parts of a) dataset best
- Combining many weak models may give rise to a strong model ("Many hands make light work")

Bagging: representation

- Bootstrap aggregating
- Use m bootstrap samples to learn m models M_i: multiple experts; output:

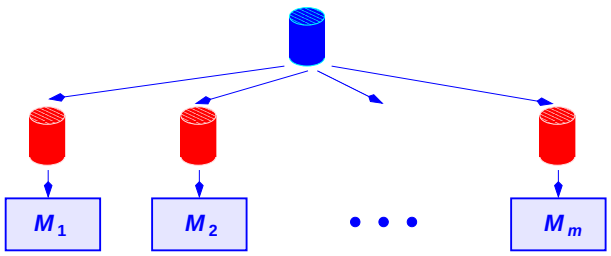
  $$\hat{f}^*_m(\mathrm{x}) = \frac{1}{m} \sum_{k=1}^{m} \hat{f}_k(\mathrm{x})$$

  is Monte-Carlo estimate for true function $f = \lim_{m\to\infty} \hat{f}^*_m$
- Typical use:



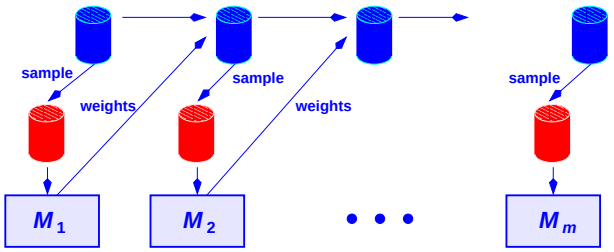
Bagging: learning

- Bias-variance decomposition – bias is fixed for specific technique; try reducing variance of learning
- Random samples ⇒ slightly different models
- Result: expert models for a particular part of the feature space
- Bagging and naive Bayesian networks?



Boosting

- Limitation of bagging: models are learnt completely separately
- Solution: boosting – learn models that complement each other
- Output: weighted combination of model results



Boosting algorithm

- Dataset D with instances $\mathbf{x} \in D$; $\mathcal{M}$ models
- Correctly classified instances improve by
  $\epsilon/(1-\epsilon)$
  where ϵ is the error rate
- $w(\mathbf{x})$ are the weights attached to instances

Boosting($\mathcal{M}, D$)

{
  for each $\mathbf{x} \in D$ **do**
    $w(\mathbf{x}) \leftarrow$ initial-value
  for $k \leftarrow 1, \ldots, m$ **do**
    $E \leftarrow$ Sample(D)
    $M_k \leftarrow$ LearnModel(E)
    $\epsilon \leftarrow$ ErrorRate(D, M_k)
    if $\epsilon \neq 0$ **and** $\epsilon < 0.5$ **then**
      $\mathcal{M} \leftarrow \mathcal{M} \cup \{M_k\}$
      for each $\mathbf{x} \in D$ **do**
        if $\mathbf{x}$ is classified correctly by M_k
        then $w(\mathbf{x}) \leftarrow w(\mathbf{x}) \cdot \epsilon/(1-\epsilon)$
    Renormalise(D)
}

Application multiple models

- Models are combined by weighted votes (classification) or weighted average (regression)
- Contribution of each model M_k:
  $$w_k = -\log \frac{\epsilon_k}{1-\epsilon_k}$$
  Note $w_k \to \infty$ if $\epsilon_k \downarrow 0$ where ϵ_k is the error rate for model M_k
- C: class variable with values c

Classify($\mathcal{M}, \mathbf{x}$)

{
  for each $c \in C$ **do**
    $w(c) \leftarrow 0$
  for $k \leftarrow 1, \ldots, m$ **do**
    if $c =$ MostLikelyClass($M_k, \mathbf{x}$) **then**
      $w(c) \leftarrow w(c) + w_k$
  return highest c
}