Experiments

Date Preprocessing, Metrics, Model Section, and Ensembling

Shan-Hung Wu
shwu@cs.nthu.edu.tw

Department of Computer Science,
National Tsing Hua University, Taiwan

NetDB-ML, Spring 2015

Outline

Outline

Outline

Inputs and Assumptions

- Elements of the training set $\mathcal{X} = \left\{ \left(\boldsymbol{x}^{(t)}, r^{(t)} \right) \right\}_{t=1}^{N}$ are assumed to be i.i.d. and drawn from the same (unknown) joint distribution $F(\boldsymbol{x}, r)$
  - $\boldsymbol{x}^{(t)} \in \mathbb{R}^d$ and d is called the ***input dimension***
- A new pair $(\boldsymbol{x}', r')$ (whose r' is unknown and will be predicted by our model) is also assumed to be drawn from the same distribution
  - If $\boldsymbol{x}'$ is assumed to come from a different distribution, then we call the learning task ***transitive learning***

Features

- As we have seen, data may be raised to another space before fed into a learning algorithm
  - E.g., kernelization
- We call the space and dimension of the raised instances the ***feature space*** and ***feature dimension*** respectively
- Input space $\neq$ feature space

Data Preprocessing

- Examples are usually ***preprocessed*** before becoming the input
- Why preprocessing?
- Real world data are generally
  - Incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data
  - Noisy: containing ***noises*** or ***outliers***
    - Noises, e.g., due to imprecision in recording the inputs or ***latent*** (or ***hidden***) attributes that affect the actual labeling
    - Outliers, e.g., due to errors in labeling examples
  - Inconsistent: different data sources may use different names, scale, precision, etc.

Tasks in Data Preprocessing

- ***Data integration***: using multiple databases, data cubes, or files
- ***Data cleaning***: fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies
- ***Data transformation***: normalization and aggregation
- ***Data reduction***: reducing the volume but producing the same or similar analytical results
  - ***Discretization***: part of data reduction, replacing numerical attributes with nominal ones

Data Integration

- How to correct/merge inconsistent data ?

Data Integration

- How to correct/merge inconsistent data ?
  - No generally good solution
  - Usually rely on domain knowledge or human experts

Outline

Filling Missing Values

- Missing values could be attributes or labels
- How?

Filling Missing Values

- Missing values could be attributes or labels
- How?
- Ignore the instance: usually done when the label is missing
- Use the attribute mean (or majority nominal value) to fill in the missing value
- Use the attribute mean (or majority nominal value) for all samples belonging to the same class
- Predict the missing value by using a learning algorithm: consider the attribute with the missing value as the "label" and run a learning algorithm (usually Bayes or decision tree) to predict the missing value from other attributes

Identify Outliers and Smooth-Out Noises

- How?

Identify Outliers and Smooth-Out Noises

- How?
- Binning (histograms): reducing the number of attribute values by grouping them into intervals (***bins***)
  - Sort the attribute values and partition them into bins
    - Equal-interval (equiwidth) binning: split the whole range of values in intervals with equal size
    - Equal-frequency (equidepth) binning: use intervals containing equal number of values
  - Then smooth by bin means, bin median, or bin boundaries
- Clustering: group values in clusters and then detect and remove outliers (automatic or manual)
- Regression: smooth by fitting the data into regression functions

Outline

Normalization

- Scaling attribute values to fall within a specified range
  - Example: to transform v in $[min, max]$ to v' in $[0, 1]$, apply $v' := (v - min)/(max - min)$
- Scaling by using mean and standard deviation
  - Useful when min and max are unknown or when there are outliers
  - Example: ***Z-normalization***: $v' := (v - mean)/std$
- Why normalization?

Normalization

- Scaling attribute values to fall within a specified range
  - Example: to transform v in $[min, max]$ to v' in $[0, 1]$, apply $v' := (v - min)/(max - min)$
- Scaling by using mean and standard deviation
  - Useful when *min* and *max* are unknown or when there are outliers
  - Example: ***Z-normalization***: $v' := (v - mean)/std$
- Why normalization?
  - To prevent some attributes from ***dominating*** the performance of a learning algorithm
  - E.g., those with wide value ranges

Aggregation

- Combing two or more attributes into a single attribute
  - For example, merging daily sales attributes to obtain monthly sales attributes
- Why aggregation?

Aggregation

- Combing two or more attributes into a single attribute
  - For example, merging daily sales attributes to obtain monthly sales attributes
- Why aggregation?
  - Data reduction
  - If done properly, aggregation can act as scope or scale, providing a high level view of data instead of a low level view
- Forget seasoning is a common pitfall in e-commerce learning tasks

Attribute Construction/Augmentation:

- Replacing or adding new attributes inferred by existing attributes
- Why?

Attribute Construction/Augmentation:

- Replacing or adding new attributes inferred by existing attributes
- Why?
- E.g., for social networking data where each instance represents a node in a social graph, it is good to create attributes for each node summarizing the structure of its two or three hops ***ego network***

Outline

Reducing the Number of Instances

- Sampling
- But not suitable for all tasks
  - E.g., identifying terrorists

Reducing the Number of Attributes

- Data cube aggregation: applying roll-up, slice, or dice operations
- Removing irrelevant attributes: attribute selection
  - Filtering and wrapper methods; e.g., forward/backward attribute selection
- Principle component analysis (numeric attributes only)
  - Searching for a lower dimensional space that can best represent the data

Reducing the Number of Attribute Values

- Discretization: round the values to their "representative" ones
  - Can be stored/processed using sparse representations
- Unsupervised discretization (labels are not used)
  - Binning (histograms): reducing the number of attributes by grouping them into intervals (bins)
    - Eequiwidth or equidepth
  - Clustering: grouping values in clusters
- Supervised discretization
  - Discretization based on ***concept hierarchies***

Concept Hierarchies based on Class Boundaries

- Three steps:
  - Sort values
  - Place breakpoints between values belonging to different classes
  - If too many intervals, merge intervals with equal or similar class distributions
  - Repeat the above steps to create a concept hierarchy

Information-based Concept Hierarchies (1)

- Information in a class distribution:
  - Denote a set of five values occurring in instances belonging to two classes (+ and -) as $[+,+,+,-,-]$; that is, the first 3 belong to "+" tuples and the last 2 - to "-" tuples
  - Then, $\mathit{Info}([+,+,+,-,-]) = -(3/5) * \log(3/5) - (2/5) * \log(2/5)$
    - log's are base-2
  - 3/5 and 2/5 are relative frequencies (probabilities)
- Information after a split
  - $\mathit{Info}([+,+],[+,-,-]) = (2/5) * \mathit{Info}([+,+]) + (3/5) * \mathit{Info}([+,-,-])$
  - 2/5 and 3/5 are weight coefficients

Information-based Concept Hierarchies (2)

- Method:
  - Sort the values
  - Calculate information in all possible splits
    - No need to consider split points between values belonging to the same class as it will increase information
  - Choose the split that minimizes information
  - Apply the same to the resulting intervals until some stopping criterion is satisfied
    - E.g., there's no split that leads to enough reduction in information

Outline

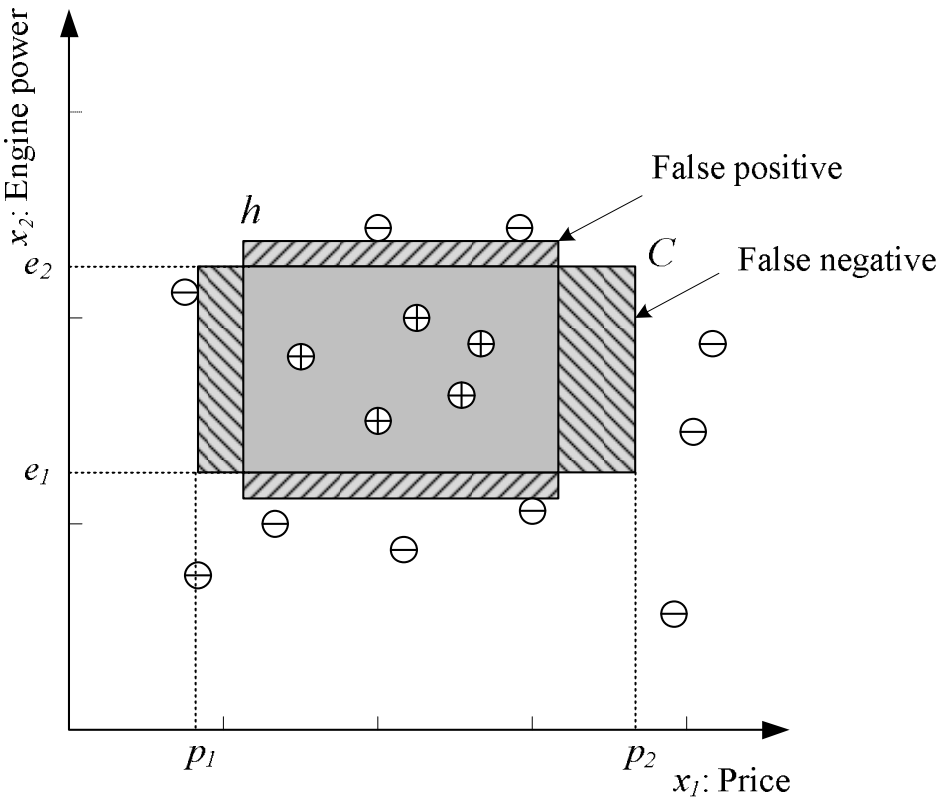
Outline

False Positives and Negatives (1/2)

- A negative instance which is wrongly predicted as positive is called the ***false positive***; and a positive instance which is wrongly predicted as negative is called the ***false negative***
  - They are all errors, why distinguished?

False Positives and Negatives (1/2)

- A negative instance which is wrongly predicted as positive is called the ***false positive***; and a positive instance which is wrongly predicted as negative is called the ***false negative***
  - They are all errors, why distinguished?
  - Depending on applications, they may not be equally serious
  - E.g., spam filtering, cancer detection, etc.



False Positives and Negatives (2/2)

- Unfortunately, in practice we don't know C
- We therefore estimate the false positive/negative rate by a ***testing set***
  - Remove certain examples in the training set and put them into the testing set
  - Examples in the testing set do not participate in the training process
  - After training, use the classifier to predict the labels of the instances in the testing set and compare with their actual label to obtain the ***confusion matrix***:

	Predicted Class		
True Class	Positive	Negative	Total
Positive	tp	fn	p
Negative	fp	tn	n
Total	p'	n'	T

Performance Measures

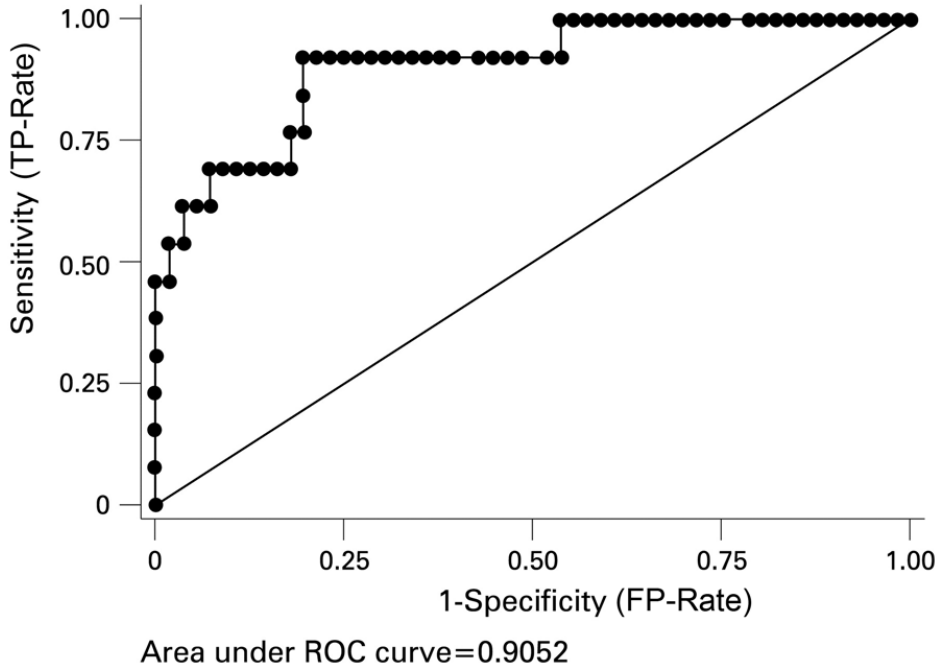
Name	Formula
Error	$(fp+fn)/T$
Accuracy	$1-error$
FP-Rate	fp/n
FN-Rate	fn/p
Precision	tp/p'
Recall (TP-Rate)	tp/p
Sensitivity (TP-Rate)	tp/p
Specificity	tn/n

ROC Curves (1)

- If a classifier gives soft values (e.g., $[-1, 1]$) rather than the hard ones $\{-1, 1\}$, its performance varies with a threshold θ
  - Instances with scores larger/smaller than θ is predicted as positive/negative respectively
- The ***Receiver Operating Characteristics (ROC) curve*** measures the performance of a classifier at different thresholds
  - Rank the T instances from the highest to the lowest score
  - For each threshold $\theta \in \{0, 1, \cdots, T\}$, predict those instances before (inclusive)/after (exclusive) θ as positive/negative respectively, and then calculate tp_θ and fp_θ
  - Connect the pairs (tp_0, fp_0), (tp_1, fp_1), $\cdots$, (tp_T, fp_T) and we obtain an ROC curve

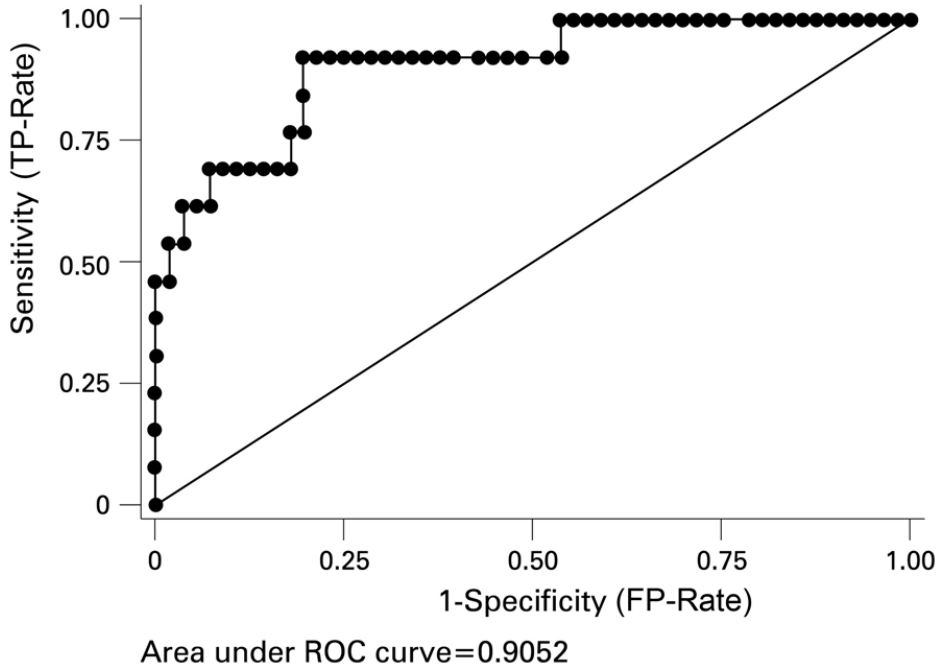
1	
1	
0.87	θ
0.64	$\Downarrow$
$\vdots$	
−0.88	
−0.93	
−1	

ROC Curves (2)



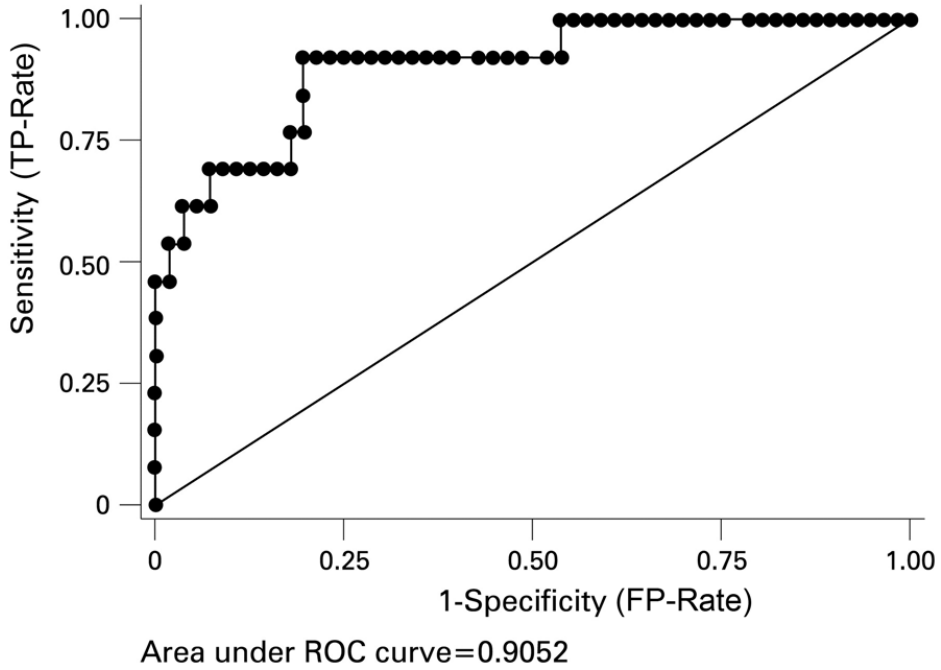
- What does the diagonal line means?

ROC Curves (2)



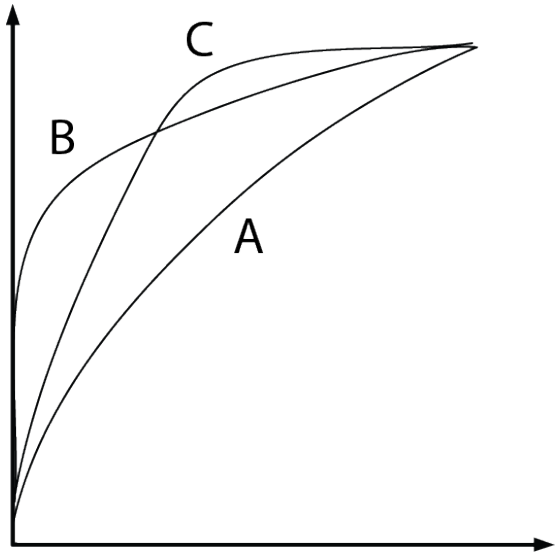
- What does the diagonal line means?
  - The ROC curve of pure guesses
- How should the line given by a good classifier look like?

ROC Curves (2)



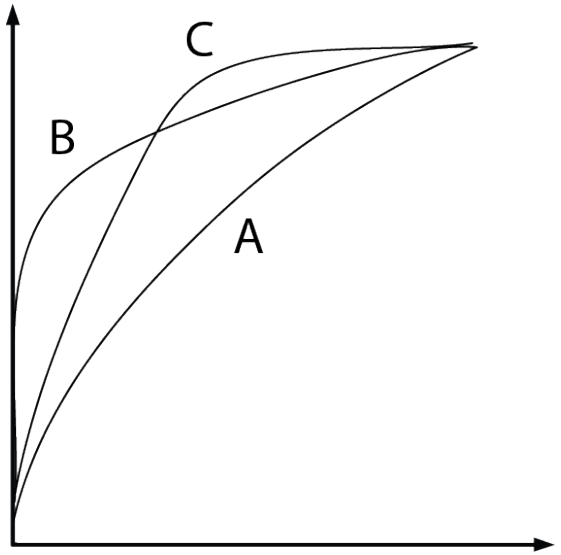
- What does the diagonal line means?
  - The ROC curve of pure guesses
- How should the line given by a good classifier look like?
  - The more a classifier gets closer to the upper-left corner the better

ROC Curves (3)



- Which one is the best?

ROC Curves (3)



- Which one is the best?
  - Classifiers B and C are better than A
  - B and C are preferred under different loss conditions: if you tolerate no more than 15% FP-rate, you should pick B at $\theta = 0.15T$, and 60% TP-rate is best you can get
  - If you tolerate 40% FP-rate, then pick C at $\theta = 0.4T$, which gives 90% TP-rate

AUC

- We can reduce an ROC curve to a single value by calculating the ***Area Under the Curve (AUC)***
  - An ideal classifier has AUC 1, and the pure guess has 0.5
- What does AUC mean?

AUC

- We can reduce an ROC curve to a single value by calculating the ***Area Under the Curve (AUC)***
  - An ideal classifier has AUC 1, and the pure guess has 0.5
- What does AUC mean?
  - AUC is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one [Homework: By partitioning the AUC horizontally]

Outline

Measuring the Regression Performance

- One common measure is the ***coefficient of determination***: $R^2 = 1 - E_{RSE}$
- $E_{RSE} = \frac{\sum_{t=1}^{N} (r^{(t)} - h(x^{(t)}; \theta))^2}{\sum_{t=1}^{N} (r^{(t)} - \overline{r})^2}$ is called the ***relative square error***
  - What doe it mean?

Measuring the Regression Performance

- One common measure is the ***coefficient of determination***: $R^2 = 1 - E_{RSE}$
- $E_{RSE} = \frac{\sum_{t=1}^{N} (r^{(t)} - h(x^{(t)}; \theta))^2}{\sum_{t=1}^{N} (r^{(t)} - \overline{r})^2}$ is called the ***relative square error***
  - What doe it mean?
  - Indicates how good our prediction is as compared to the naive prediction by ***averaging***
  - The smaller the E_{RSE} the better
- A good regression function h should have R^2 close to 1

Outline

Outline

Generalization Performance

- Assuming a hypothesis class $\mathcal{H}$, let $h \in \mathcal{H}$ be the hypothesis trained from the dataset $\mathcal{X} = \{(\boldsymbol{x}^{(t)}, r^{(t)})\}_{t=1}^{N}$ by minimizing the empirical error: $R_{emp}[h] := \frac{1}{N} \sum_{t=1}^{N} l(h(\boldsymbol{x}^{(t)}), r^{(t)})$
  - l is the loss function
- Generalization error of h:
  $R[h] := \int p(\boldsymbol{x}, r) l(h(\boldsymbol{x}), r) d(\boldsymbol{x}, r) = E_{\mathcal{I} \times \mathcal{L}}[l(h(\boldsymbol{x}), r)]$
- Let $h^* := \arg\inf_{g \in \mathcal{H}} R[g]$ and $R^* := \inf_{f: \mathcal{X} \to \mathbb{R}} R[f]$
- Our ultimate goal:
  $$R[h] \to R^*$$
- $R[h] - R^* = R[h] - R[h^*] + R[h^*] - R^*$
  - $R[h] - R[h^*]$ is called the ***estimation error***
  - $R[h^*] - R^*$ is called the ***approximation error***

Model Selection

- We need to pick $\mathcal{H}$ with right complexity to prevent both underfitting and overfitting
- In the context of kernelized and regularized linear models, we need to pick good hyperparameters
  - E.g., γ in the Gaussian RBF kernel, and the coefficient λ of a regularization term
- How to determine good hyperparameters?

Three-Way Data Splits

- Idea1: try out all possible combinations of hyperparameters and pick the one which gives the least testing error
- Good idea?

Three-Way Data Splits

- Idea1: try out all possible combinations of hyperparameters and pick the one which gives the least testing error
- Good idea?
  - Problem 1: in practice, we may not have time to try out all possible combinations
    - Global search techniques such as the ***grid search*** can be used speed up try outs
  - Problem2: testing instances are revealed in the training process, so you cannot report the generalization performance of the learned hypothesis anymore
- Idea 2: in addition to the testing set, we can split a ***validation set*** from the training set and then choose the combination that results in the least ***validation error***
  - Testing set is used only for the evaluation of generalization performance

Outline

Problems of Holdout Methods

- We holdout the validation and testing sets for the model selection and performance evaluation respectively
- Drawbacks?

Problems of Holdout Methods

- We holdout the validation and testing sets for the model selection and performance evaluation respectively
- Drawbacks?
- Given a small dataset, we may not afford the "luxury" of setting aside a portion of the dataset from training
- The holdout estimate of error rate will be misleading if we happen to get an "unfortunate" split
- Improvement?

Cross-Validation (1)

- We usually perform ***K-fold cross-validation*** to exploit the labeled data both for training and other holdout tasks
  - Applicable to either model selection or generalization performance evaluation
- For example, for model selection:
  - Split the training set evenly into K subsets (folds)
  - Given a particular combination of hyperparameters, train K hypotheses $h_1, \cdots, h_K$ where each h_i is trained on all but the ith fold
  - Calculate error of each h_i made on the ith fold, and average the errors of h_i's to obtain the ***cross-validation error***
  - Pick the combination of hyperparameters that results in the least cross-validation error
- Similar for generalization performance evaluation

Cross-Validation (2)

- How many folds (K) we need?

Cross-Validation (2)

- How many folds (K) we need?
- The cross-validation error is an average of the estimators of the true errors on different folds
  - The mean square error between each estimator and its true error can be expressed as $(bias)^2 + variance$ (see appendix: Statistics)
- With a large K, the cross-validation error tends to have a small bias but large variance
  - Small bias since each h_i is trained on more examples
  - Large variance because training samples are more similar and the h_i's are more positively correlated
- Conversely, with a small K, the cross-validation error tends to have a large bias but small variance
- Usually, $K = 5$ or 10
- For very small dataset (where error is dominated by bias), we can choose $K = N$, which we call the ***leave-one-out cross-validation***

Outline

Rationale

- There is no single learning algorithm that in any domain always induces the most accurate learner.
- By suitably combining multiple base-learners, the accuracy can be improved.

Outline

Voting

- The simplest way to combine multiple classifiers is by ***voting***, which corresponds to taking a linear combination of the learners:

$$y_i = \sum_j w_j d_{ji} \text{ where } w_j \geqslant 0, \sum_j w_j = 1.$$

- In the simplest case, all learners are given equal weight $w_j = 1/L$ and we have simple voting called ***plurality voting*** that corresponds to taking an average.

Rule	Fusion function $f(\cdot)$
Sum	$y_i = \frac{1}{L} \sum_{j=1}^{L} d_{ji}$
Weighted sum	$y_i = \sum_j w_j d_{ji}, w_j \geqslant 0, \sum_j w_j = 1$
Median	$y_i = \text{median}_j d_{ji}$
Minimum	$y_i = \min_j d_{ji}$
Maximum	$y_i = \max_j d_{ji}$
Product	$y_i = \prod_j d_{ji}$

Table : Classifier combination rules

Expected Value and Variance (1/2)

- Let's assume that d_j are i.i.d. with expected value $E[d_j]$ and variance $Var(d_j)$. When $w_j = 1/L$, the expected value and variance of the output are

$$E[y] = E\left[\sum_j \frac{1}{L} d_j\right] = \frac{1}{L} L E[d_j] = E[d_j]$$

$$Var(y) = Var\left(\sum_j \frac{1}{L} d_j\right) = \frac{1}{L^2} Var\left(\sum_j d_j\right) = \frac{1}{L^2} L \times Var(d_j) = \frac{1}{L} Var(d_j)$$

  - We see that the expected value doesn't change, so the bias doesn't change.
  - But variance, and therefore mean square error, decreases as the number of independent voters, L, increases.

Expected Value and Variance (2/2)

- In the general case where d_j are ***not*** i.i.d.,

$$Var(y) = \frac{1}{L^2} Var\left(\sum_j d_j\right) = \frac{1}{L^2}\left[\sum_j Var(d_j) + 2\sum_j \sum_{i<j} Cov(d_j, d_i)\right]$$

  which implies that if learners are positively correlated, variance increases.

Outline

Bagging

- ***Bagging*** is a voting method whereby base-learners are made different by training them over slightly different training sets.

1. Generating L slightly different samples from a given sample is done by bootstrap, where given a training set $\mathcal{X}$ of size N, we draw N instances randomly from $\mathcal{X}$ ***with replacement***
   - Because sampling is done with replacement, it is possible that some instances are drawn more than once and that certain instances are not drawn at all.
2. When L samples $\mathcal{X}_j, j = 1, \ldots, L$, are generated, the base-learners d_j are trained with these L samples in $\mathcal{X}_j$.

Outline

Boosting

- In bagging, generating complementary base-learners is left to chance and to the unstability of the learning method.
- In boosting, we actively try to generate complementary base-learners by training the next learner on the mistakes of the previous learners.
- The original boosting algorithm combines three weak learners to generate a strong learner.
  - A ***weak learner*** has error probability less than 1/2, which makes it better than random guessing on a two-class problem, and a ***strong learner*** has arbitrarily small error probability.

The Original Boosting Algorithm

1. Given a large training set, randomly divide it into three.
2. Use $\mathcal{X}_1$ to train d_1 and feed $\mathcal{X}_2$ to d_1.
3. Use all instances misclassified by d_1 and also as many instances on which d_1 is correct from $\mathcal{X}_2$ to train d_2. Then feed $\mathcal{X}_3$ to d_1 and d_2.
4. Use the instances on which d_1 and d_2 disagree to train d_3.
5. During testing, given an instance, give it to d_1 and d_2. If they agree, that is the response, otherwise the response of d_3 is taken.

The Original Boosting Algorithm

1. Given a large training set, randomly divide it into three.
2. Use $\mathcal{X}_1$ to train d_1 and feed $\mathcal{X}_2$ to d_1.
3. Use all instances misclassified by d_1 and also as many instances on which d_1 is correct from $\mathcal{X}_2$ to train d_2. Then feed $\mathcal{X}_3$ to d_1 and d_2.
4. Use the instances on which d_1 and d_2 disagree to train d_3.
5. During testing, given an instance, give it to d_1 and d_2. If they agree, that is the response, otherwise the response of d_3 is taken.

- The disadvantage is that it requires a very large training sample.

AdaBoost (1/2)

- We can instead use ***AdaBoost*** which uses the same training set over and over and thus need not be large.
- The idea is to modify the probabilities of drawing the instances as a function of the error.
- Let p_j^t denote the probability that the instance pair (x^t, r^t) is drawn to train the jth base-learner and let ϵ_j denote the error rate of d_j on the dataset used at step j.

AdaBoost (2/2)

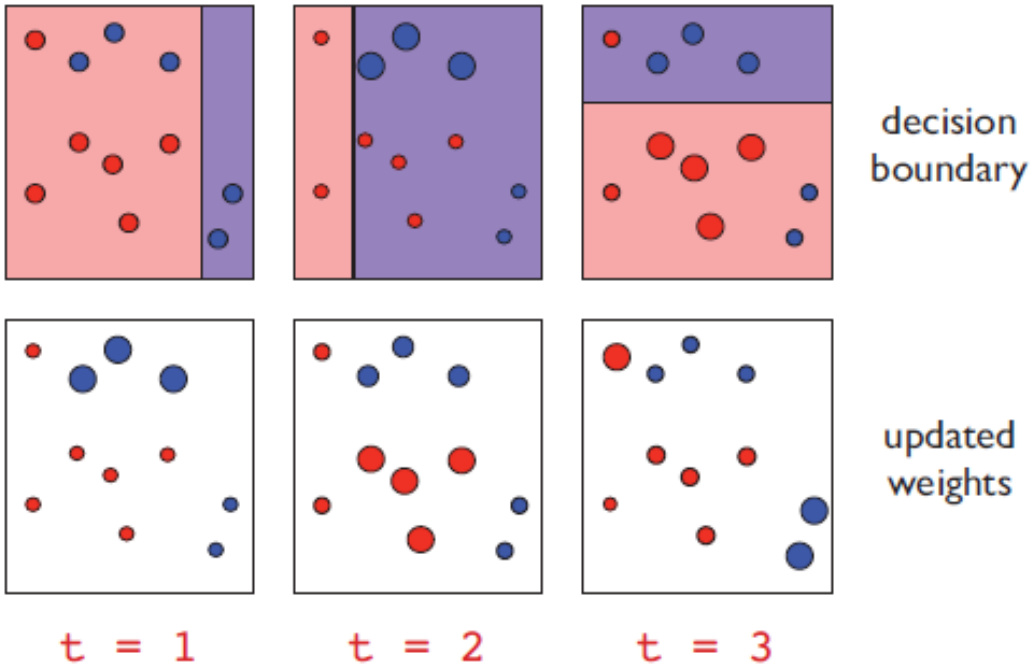
- Training

1. Initialize $p_1^t = 1/N, t = 1, \cdots, N$.
2. Start from $j = 1$:
   1. Randomly draw $\mathcal{X}_j$ from $\mathcal{X}$ with probabilities p_j^t to train d_j.
   2. Since AdaBoost requires $\epsilon_j < 1/2$, we stop adding new base-learners if not.
   3. Define $\beta_j = \epsilon_j / (1 - \epsilon_j) < 1$ and set $p_{j+1}^t = \beta_j p_j^t$ if d_j correctly classifies x^t. Otherwise, $p_{j+1}^t = p_j^t$.
   4. Normalize p_{j+1}^t by $\sum_t p_{j+1}^t$.

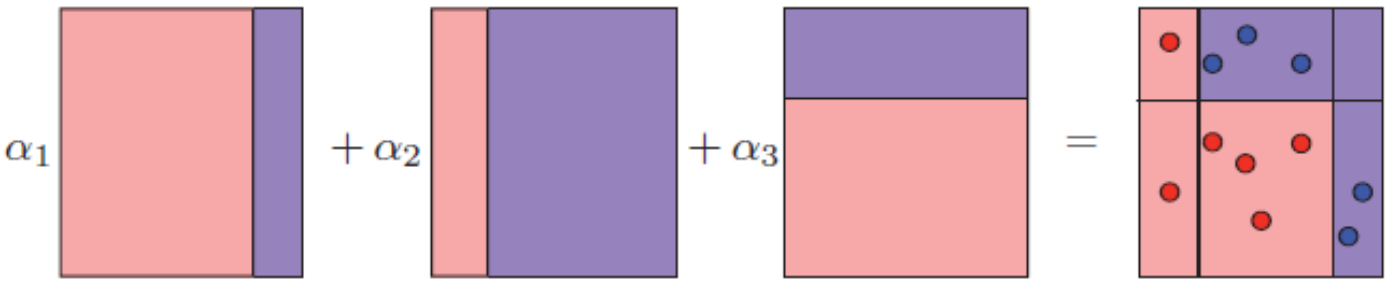
- Testing

1. Given x, calculate $d_j(x)$ for all j.
2. Calculate class outputs, $i = 1, \cdots, K$: $y_i = \sum_j \left(\log \frac{1}{\beta_j} \right) d_{ji}(x)$.

Example


decision
boundary
updated
weights
t = 1
t = 2
t = 3

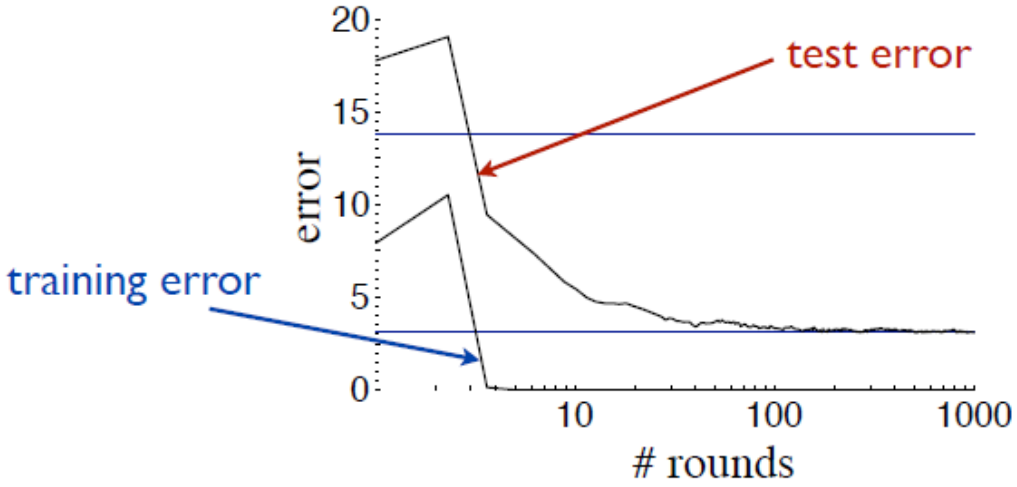
(a)


α_1
$+\alpha_2$
$+\alpha_3$
=

(b)

Large Margin Perspective

- When adding a new base-learner, we increase the probability of drawing a misclassified instance. Thus d_{j+1} focuses more on instances misclassified by d_j.
- Given an instance, all d_j take a weighted vote where $w_j = \log(1/\beta_j)$ is proportional to the base-learner's accuracy.
- It can been shown that AdaBoost can increase the margin, whose aim is similar to that of the SVM.



C4.5 decision trees (Schapire et al., 1998).