Algorithm Independent Issues

CS 550: Machine Learning

Does the "best" algorithm exist?

- **No Free Lunch Theorem**
  - There exists no learning algorithm that is superior to the other algorithms for all situations
  - If one algorithm seems to outperform another in a particular situation, it is a consequence of its fit to this situation, not the general superiority of the algorithm
  - Thus, there are no context-independent or usage-independent reasons to favor one learning algorithm over another

- *So, how can we understand that a learning algorithm is a good fit to a particular situation?*

Estimating the classification accuracy

- Bias in the estimate: Accuracy on the training samples is often a poor estimator of the accuracy over future samples
  - Likelihood of a model to overfit the training samples is high especially when the model is complex and the training set is small
  - To obtain an unbiased estimate of the future accuracy, the model should be tested on samples that are chosen independently of the training samples and the model
- Variance in the estimate: The measured accuracy can vary from the true accuracy depending on the test samples
  - The expected variance is high especially when the test set is small

Estimating the classification accuracy

- Given a model M and a dataset S containing n samples drawn at random according to a distribution D
  1. What is the best estimate of the accuracy of M over future samples drawn from the same distribution?
  2. What is the probable error in this accuracy estimate?

The **true error** of model M with respect to target function f and distribution D is the probability that M will misclassify a sample drawn at random according to D

$$error_D(M) \equiv \Pr_{x \in D}\left[f(x) \neq M(x)\right]$$

The **sample error** of model M with respect to f and the dataset S

$$error_S(M) \equiv \frac{1}{n}\sum_{x \in S}\delta(f(x), M(x)) \qquad \delta(f(x), M(x)) = \begin{cases} 1 & \text{if } f(x) \neq M(x) \\ 0 & \text{otherwise} \end{cases}$$

How good an estimate of $error_D(M)$ is provided by $error_S(M)$?

Estimating the classification accuracy

- Given no other information, the most probable value of $error_D(M)$ is $error_S(M) = \mathrm{r} / \mathrm{n}$, where r is the number of misclassified samples, and
- With N percent confidence, $error_D(M)$ lies in the interval of

$$error_S(M) \mp z_N \sqrt{\frac{error_S(M)\left(1 - error_S(M)\right)}{n}}$$

z_N should be chosen depending on the desired confidence level

  - If dataset S contains n samples drawn independent of one another, and independent of model M, according to the distribution D, and
  - If $\mathrm{n} \geq 30$ [more accurately, $\mathrm{n}\, error_S(M)\,(1 - error_S(M)) \geq 5$]

Confidence level N%	50 %	68 %	80 %	90 %	95 %	98 %	99 %
Constant z_N	0.67	1.00	1.28	1.64	1.96	2.33	2.58

What does it mean?

10 miscorrect classifications in 100 samples → $error_S(M) = 0.1$
With 95% confidence, the true error lies in the interval of

$$0.1 \mp \underbrace{1.96\sqrt{\frac{0.1\ \ 0.9}{100}}}_{\sim 0.059} \quad \Rightarrow \quad 0.041 \le error_D(M) \le 0.159$$

z_N =1.96 for two-sided 95% confidence interval

100 miscorrect classifications in 1000 samples → $error_S(M) = 0.1$
With 95% confidence, the true error lies in the interval of

$$0.1 \mp \underbrace{1.96\sqrt{\frac{0.1\ \ 0.9}{1000}}}_{\sim 0.019} \quad \Rightarrow \quad 0.081 \le error_D(M) \le 0.119$$

10 000 miscorrect classifications in 100 000 samples → $error_S(M) = 0.1$
With 95% confidence, the true error lies in the interval of

$$0.1 \mp \underbrace{1.96\sqrt{\frac{0.1\ \ 0.9}{100\ 000}}}_{\sim 0.002} \quad \Rightarrow \quad 0.098 \le error_D(M) \le 0.102$$

Binomial Distribution

- Let's consider a coin-tossing experiment to find the probability of obtaining head (let's call this probability p)
- This experiment involves n trials, in each of which we obtain either head (1) or tail (0)
  - Each trial is Bernoulli
  - Thus, the entire experiment follows the Binomial distribution

$$
\begin{aligned}
P(X = r) &= \frac{n!}{r!\,(n-r)!}\, p^r\, (1-p)^{n-r} \\
E[X] &= n\,p \\
Var(X) &= n\,p\,(1-p) \\
\sigma_X &= \sqrt{n\,p\,(1-p)}
\end{aligned}
$$

For sufficiently large values of n, the Binomial distribution is closely approximated by a normal distribution with the same mean and variance. Most statisticians recommend using the normal approximation only when $n\,p\,(1-p) \geq 5$

- *Design an experiment to find the probability p of misclassification*
- *This experiment involves classifying the samples of a randomly drawn set with a size of n*
- *For each sample, we obtain either misclassification (1) or correct classification (0)*

Estimating the classification accuracy

What is the likely difference between the true and the sample error?

1. $error_S(M) = \mathrm{r} / \mathrm{n}$ is an unbiased estimator of $error_D(M) = \mathrm{p}$

   since $E[\hat{p}] - p = 0$

2. Derive a confidence interval for $error_D(M)$
   - The derivation is quite tedious for the Binominal distribution
   - If p follows a normal distribution, the measured p will fall the following interval N% of the time

$$\mu_p \mp z_N \sigma_p$$

N% confidence interval for p is an interval that is expected to contain p with N% probability

Thus, $error_D(M)$ will fall the following interval with N% confidence

$$error_S(M) \mp z_N \underbrace{\sqrt{\frac{error_S(M)\left(1 - error_S(M)\right)}{n}}}$$

Multiplying a random variable by constant n multiplies the variance by n^2

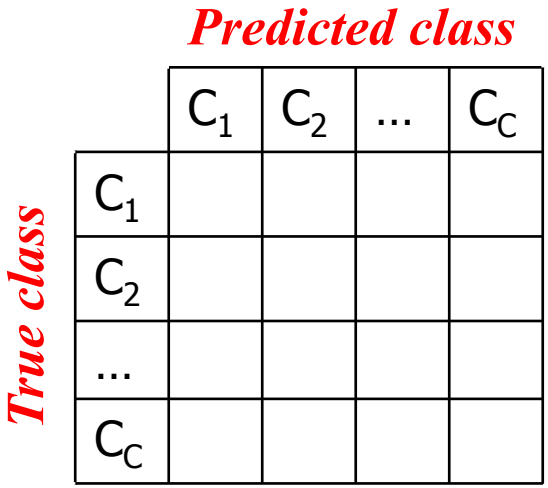
This gives two-sided bounds with N% confidence (α significance level where α = 1 – N%)

For one-sided bound with the same confidence, use $z_{(1-2\alpha)}$

Classification error/accuracy

- Sometimes, we may want to consider class-based accuracies in addition to the overall accuracy
  - Especially when the class distributions are unbalanced

- Confusion matrix



Classification error/accuracy

- For two-class classifications (e.g., for diagnostic systems)

Recall
Hit rate
True pos rate

$$Sensitivity = \frac{TP}{TP + FN}$$

$$Specificity = \frac{TN}{TN + FP}$$

True neg rate

$$\Pr ecision = \frac{TP}{TP + FP}$$

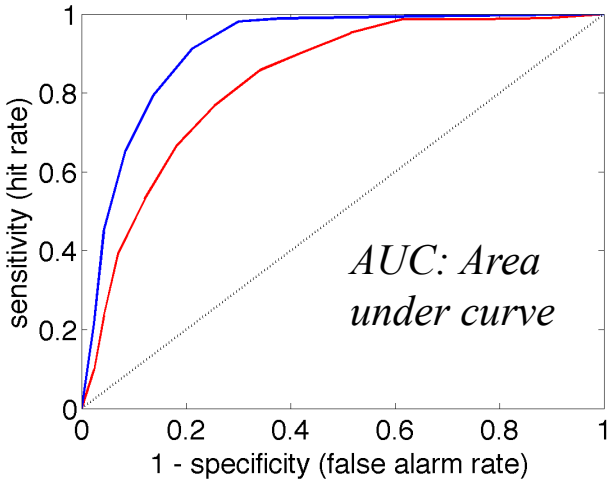
$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$F\text{ - }score = 2\,\frac{\Pr ecision\ \mathrm{Re}call}{\Pr ecision + \mathrm{Re}call}$$

$$Dice\ index = \frac{2\ TP}{2\ TP + FP + FN}$$

		Predicted +	*Predicted* −
True	+	TP	FN
	−	FP	TN

ROC (receiver operating characteristics) curves



Some practical issues

How to form a test set(s)?

- One separate test set
  - If one is available (e.g., if you use a public dataset), use it as it is
  - If not, randomly split the data into two
    - Consider class distributions
    - Consider dependency between the samples (if any)
      - No dependency should exist in the ideal case
      - However, dependency may exist in practice
        (consider the handwritten characters collected from the same subject)
- Multiple test sets → partition the data many times
  - ***Bootstrapping:*** Draw samples from the dataset with replacement
  - ***K-fold cross validation:*** Form k partitions of the dataset
  - ***Leave-one-out:*** Form partitions, each containing a single sample
    - Leave-one-subject-out may be necessary if there exists dependency between the samples

Some practical issues

How to select model parameters?

- Grid search
  - Determine a set of values for each parameter
  - Consider every combination of the selected values in these sets
  - Select the combination that gives the best accuracy
    - Consider this selection as a part of training, but do not use the training set accuracy as the selection criterion to prevent overfitting
    - For example, you may use k-fold cross-validation on the training set and select the parameter combination that leads to the highest cross-validation accuracy
    - ***Do not use the test set accuracies in any step of this selection***
- After setting the parameters, use the entire training set to train your model
- ***Then, test your model on an independent test set*** which was not used in training or parameter selection

Some practical issues

When the training set is unbalanced

- Some classifiers (e.g., neural networks) may have difficulties in learning the minority class
  - They may favor the majority class
- A common practice to deal with this is to rebalance such a training set artificially
  - ***Oversampling:*** replicate training samples from the minority class(es)
  - ***Undersampling:*** ignore some training samples from the majority class(es)

Some practical issues

When the training set is too small

- Training with noise
  - Virtual training samples can be generated
  - In the absence of problem-specific information
    - Virtual samples can be generated by adding d-dimensional Gaussian noise to true training samples
    - In classification, noise is added to inputs and class labels should be left unchanged
    - In regression, noise could be added to both inputs and outputs
  - This method generally does not improve accuracy
    - For highly local techniques such as the nearest neighbor method

Some practical issues

When the training set is too small

- Manufacturing data
  - We can "manufacture" training samples that convey more information than uncorrelated noise
    - If we have knowledge about the sources of variation among samples
  - E.g., in optical character recognition, we manufacture data by
    - Rotating the images of training samples
    - Performing simple image processing on the images to simulate a bold face character
  - Disadvantage
    - Memory requirements may be large
    - Overall training may be slow

Some practical issues

Scaling/normalizing features

- A classifier may prefer some features over the others
  - When the orders of magnitudes of features are different
  - For example, a neural network adjusts weights in favor of features with smaller magnitudes
    - In fish classification, if mass is measured in grams and length is measured in meters, mass has larger effect than length and the opposite if mass is measured in kilograms and length is measured in millimeters
- We normalize training samples to prevent this problem
  - Samples are shifted so that the average of each feature is 0.0
  - Dataset is normalized so that the variance of each feature is 1.0
  - Test samples must be standardized with the same transformation

How to compare two classifiers?

If you have their results on a single test set

- Use the Mc Nemar's test
  - Form a contingency table
  - Accept the hypothesis that the two algorithms have the same error rate at a significance level α if

$$\frac{\left(\left|e_{01}-e_{10}\right|-1\right)^2}{e_{01}+e_{10}} \le \chi^2_{\alpha,1}$$

Contingency table

No of samples misclassified by both	No of samples misclassified by A but not B } e_{01}
No of samples misclassified by B but not A } e_{10}	No of samples correctly classified by both

Example : Are the following two algorithms same with a significance level of 0.05?

	A	B
100	+	+
12	+	−
25	−	+
30	−	−

$$\frac{\left(\left|25-12\right|-1\right)^2}{25+12} = 3.8919$$

Significance level α	0.20	0.10	0.05	0.02
$\chi^2_{\alpha,1}$	1.64	2.71	3.84	5.41

Chi-square statistics with dof = 1

How to compare two classifiers?

If you have their results on k test sets?

- Paired t-test (parametric test)
  - Assumes that the test set errors for both of the classifiers are normally distributed so their differences are
  - Uses the t-test to check whether or not the mean of these differences is equal to zero (statistically significantly)
- Wilcoxon signed-rank test (nonparametric test)
  - Ranks the differences in errors (ignoring the signs) and sums the ranks for the positive and negative differences (corresponding to the 1^{st} and 2^{nd} classifiers)
  - Claims that the difference between the classifiers is statistically significant if the smaller of the sums is smaller than the critical value defined for the Wilcoxon test

How to compare multiple classifiers?

- ANOVA -- analysis of variance (parametric test)
- Friedman test (nonparametric test)

  - One should be careful about selecting the critical values for the given significance levels
  - When multiple classifiers are to be compared, significance levels should be lowered for example using Bonferroni correction