Questions Arising in Machine Learning

The following questions are relevant:

1. How accurate is the hypothesis for a particular target concept and example-set?

2. How does accuracy vary with the number of examples provided?

3. How fast is the hypothesis constructed for a particular target concept and example-set?

4. How does speed vary with the complexity of the target and the complexity and number of examples?

5. For a given application, is one particular algorithm *better* than another?

Experimental methodology is concerned with Q1, and to some extent, Q5

Computational Learning Theory deals with Q2–4

What is meant by "Accuracy"?

Accuracy is measured according to some probability distribution D over the space of possible examples.

> For illustration, examples might be drawn according to the uniform distribution over the Herbrand base for a given set of constants and a given predicate.

The accuracy of a hypothesis H is simply the probability, according to D, of drawing an example that H misclassifies.

Method 1: Get More Examples

We wish to estimate the accuracy of our algorithm's hypothesis H. If we can obtain m additional labelled examples, we can estimate the accuracy of H based on the binomial distribution. Call an example a *success* just if H classifies it correctly, and suppose in m examples we have $n \leq m$ successes. Then $\frac{n}{m}$ is an *unbiased estimator* of the accuracy of H.

Note: if we used the *same* examples for testing as we used for learning, the estimate of accuracy would be biased in an "optimistic way".

Further Details of Method 1

Suppose that p is the true accuracy of our hypothesis H, and we will draw m new examples. Then n (the number of successes) is distributed according to the binomial distribution $b(m, p)$ with mean p and variance $mp(1-p)$. This additional information allows us to say something about how close our estimate of accuracy is likely to be to the true accuracy. There are *several* ways to make use of this information.

Chernoff Bounds

For $0 \le p \le 1$ and m a positive integer, let X be a random variable distributed $b(m,p)$. Let LE(p,m,r) denote the probability of $X \le r$, and let GE(p,m,r) denote the probability of $X \ge r$. Then for $0 \le \alpha \le 1$:

$$\text{LE}(p,m,(1-\alpha)mp) \le e^{-\frac{\alpha^2 mp}{2}}$$

$$\text{GE}(p,m,(1+\alpha)mp) \le e^{-\frac{\alpha^2 mp}{3}}$$

As an example, suppose $m = 100$, and the number of successes we see in our sample is 90, so our estimated accuracy is .9. What is the probability that the true accuracy is less than .8? This is at most the maximum, over all values of p between 0 and .8 and all values of α between 0 and 1 such that $100p(1+\alpha) \ge 90$, of $e^{-\frac{\alpha^2 mp}{3}}$, which is at most .66.

Chebyshev's Inequality

The probability that any random variable X falls within k standard deviations of the mean is at least $(1 - \frac{1}{k^2})$.

This is somewhat better than Chernoff Bounds. Chernoff Bounds and Chebyshev's Inequality are very general and require no underlying assumptions. They are useful to provide loose probabilistic bounds on the accuracy.

Explicit Computation of Binomial Distribution

Tighter bounds can be found by explicit computation using the binomial distribution. Thus, for the question "What is the probability that the true accuracy is 0.8, and we have obtained a sample giving an estimate of at least 0.9" we can calculate directly:

$$b(x; m, p) = C(m, x)p^x q^{m-x},\ x = 0, 1, 2, ..., m.$$

Assume $p = .8$ and compute the sum of probabilities of $x = 90, 91, ..., 100$. The probabilities found are much tighter.

Problem with Method 1

We often have very limited data: Examples: determining active drugs or protein structure requires time and costly experiments; determining user interests requires time on the part of the user.

So if we had more data for testing, we'd really like to use it for training. But then we lose our unbiased estimator. It turns out we can get tighter, almost unbiased estimates if we do *resampling*. We will consider only one resampling method here.

Method 2: Leave-One-Out Cross-Validation

Suppose we have m examples. We train (learn) using $m-1$ examples, *leaving one out* for testing. We repeat this m times, each time leaving out a different example. The accuracy estimate is the number of successes (correct classifications) divided by m.

Problem with leave-one-out: can be computationally expensive. This motivates our next technique.

Method 2': k-Fold Cross-Validation

k-fold cross-validation is the same as leave-one-out cross-validation, except we only repeat k times, each time training on $m - \frac{m}{k}$ examples and testing on the remaining $\frac{m}{k}$ examples.

Presentation of Results

Results of testing on new data or k-fold cross-validation are tabulated as follows:

		Actual		
		Positive	Negative	
Predicted	Positive	n_1	n_2	n_a
	Negative	n_3	n_4	n_b
		n_c	n_d	m

n_1: number of examples in the test set that are labelled “positive” and are predicted “positive” *etcetera*

$$Accuracy = p = \frac{n_1 + n_4}{m}$$

$$S.d = \sqrt{p(1-p)/m} \quad \text{(not with k-fold c.v.)}$$

Classical statistical tests for independence between “Actual” and “Predicted” values can be applied to the table when testing is done on new data