LESSONS FROM LEARNING THEORY FOR BENCHMARK DESIGN

JOHN LANGFORD

Abstract. We discuss the trade-offs inherent in various choices for benchmark design from a quantitative learning theory perspective.

1. Structure

This discussion/tutorial consists of a few short discussions on the (theoretical) trade-offs of various choices in constructing benchmarks. Most of the results discussed here are "common sense" at high level. However, all of this "common sense" is (to some extent) quantifiable common sense, and occasionally that quantification is useful. These short discussions cover:

(1) Prediction Domains & Loss functions
(2) Prediction settings
(3) Assumption Failures

2. A Choice of Prediction Domains

One natural goal of benchmarks is objective and quantitative evaluation of the possible contestants. The natural objective and quantitative metric for learning domains seems to be a measurement of prediction ability.

This observation alone has a significant impact on the sorts of problems we might consider for a benchmark. In particular "data visualization", "clustering", "dimensionality reduction" and several other problem domains are ruled out. These domains are not (naively, at least[1]) related to prediction.

Prediction ability is sometimes quite difficult to measure, so it is important to understand the ease or difficulty of these measurements for benchmark design. Consider the following common choices:

(1) Classification: Predict a single (or a few) bits.
(2) Regression: Predict a real number (or a few real numbers).
(3) Density Estimation: Predict a probability distribution over some domain.

2.1. **Classification.**

Classification is typically about predicting one or a few bits. Examples include "Is there a tank in this picture?", "Is it raining?", etc...

Formally, we have an input space X, an output space $\{0,1\}$, and the goal of classification is to find a classifier $c : X \to \{0,1\}$ and we wish to find a classifier which is correct as often as possible. In particular, suppose D is some measure over $X \times \{0,1\}$ which expresses the proportion of the time examples occur. Then, we wish to optimize:

[1]Naturally, many of the algorithms for these domains can be used as subroutines for prediction problems.

$$\min_{c} \Pr_{(x,y)\sim D}(c(x) \neq y)$$

where the minimum is (logically at least) taken over all possible classifiers.

A sufficient, but not necessary, condition for verifying the accuracy of a classifier c is a set of m independently drawn classifiers. A standard statement has the following form:

Theorem 2.1. *(Classification Verification) For all classifiers c, for all $\delta > 0$, for all distributions D over $X \times \{0,1\}$:*

$$\Pr_{S\sim D^m}\left(\left|\Pr_{(x,y)\sim D}(c(x) \neq y) - \frac{1}{|S|}\sum_{(x,y)\in S} I(c(x) \neq y)\right| \leq \sqrt{\frac{\ln \frac{2}{\delta}}{2m}}\right) \geq 1 - \delta$$

Here $I()$is 1 when it's argument is true and 0 otherwise.

From this theorem, we see that the proportion of errors is upper and lower boundable with high probability using the observed proportion of the time that the classifier is incorrect on a set of independent examples.

It's worthwhile to note several things here:

(1) The above theorem holds for *all* distributions D and is thus relatively widely applicable. In contrast some competitive results in statistics are relevant only under restrictive assumptions about the form of the distributions.
(2) The theorem is significantly improvable [2]. The result here is the simplest correct statement rather than the tightest.
(3) The above result holds more generally in situations where the amount of loss suffered for any example is bounded to the interval $[0,1]$.

2.2. **Regression.** Regression is similar to classification except that that the space of prediction is the real line $(-\infty, \infty)$ rather than $\{0,1\}$. The standard loss function in regression is l_2 loss. For an example (x,y) and a predictor $f(x)$ the loss is:

$$(f(x) - y)^2$$

Verifying successful regression is harder than verifying successful classification as independence is not a sufficient assumption. To see this consider modifying any distribution D to have a small mass very far from the rest of the distribution. In particular, let $D' = \left(1 - \frac{1}{m^2}\right) D + \frac{1}{m^2}\Delta\left(2^m\right)$ where $\Delta(y)$ is the distribution which picks (x,y') form D and then outputs (x,y) with probability 1. With probability tending towards 1 as m grows, D' and D are indistinguishable given m samples drawn from either D or D'. However, the expected loss rate difference:

$$E_{(x,y)\sim D'}(f(x) - y)^2 - E_{(x,y)\sim D}(f(x) - y)^2$$

tends towards ∞ was m increases. Since D and D' are indistinguishable yet have wildly different l_2 losses, it is not possible to bound the expected l_2 loss given only independence.

One standard additional assumption is that the range of y is limited to some interval, say $[0,1]$. Given this additional assumption, a generalized form of theorem 3.2 holds for $f(x) \in [0,1]$.

Theorem 2.2. *(Regression Verification) For all functions f satisfying $\forall x: \ f(x) \in [0,1]$, for all $\delta > 0$, for all distributions D over $X \times [0,1]$:*

$$\Pr_{S\sim D^m}\left(\left|E_{(x,y)\sim D}(f(x)-y)^2 - \frac{1}{|S|}\sum_{(x,y)\in S}(f(x)-y)^2\right| \leq \sqrt{\frac{\ln\frac{2}{\delta}}{2m}}\right) \geq 1-\delta$$

This theorem is fairly general: it holds for arbitrary bounded distributions D and bounded functions f. The theorem is more general than it might first appear because for any known bounded interval $[a,b]$, we can rescale the above result by $(b-a)^2$ to construct a bound.

It is possible to do some analysis with a weakening of the boundedness assumption. In particular statements of the form "in the limit as the number of independent samples goes to ∞, the observed loss rate converges to the true loss rate" holds under the much weaker additional assumption that the marginal distribution of y has a finite variance. This form of statement is not very useful for benchmarking purposes because the convergence may occur only after an arbitrarily large number of examples.

Regression is often analyzed under the assumption that there exists a function f such that $y = f(x) + \vartheta$ where ϑ is distributed according to the normal distribution:

$$P(\vartheta) \propto e^{-\frac{\vartheta^2}{2\sigma^2}}$$

where σ^2 is a variance. Under this assumption, a statement qualitatively similar to theorem 2.2 holds. In fact, qualitatively similar statements can be made whenever the distribution of noise falls off according to an exponential. The difficulty with applying these results in practice is that precise knowledge of the distribution of noise may be quite hard to acquire (and therefore unreliable). In contrast, acquiring precise knowledge of an upper or lower bound may be somewhat easier and therefore more reliable.

2.3. **Density Estimation.** In density estimation the goal is typically to construct some probability density function $p(x)$ with a small expected log loss: $E_{x\sim D}\ln\frac{1}{p(x)}$. The expected log loss has an information theory interpretation: it is the expected number of bits required to specify x given $p(x)$. When x ranges over an infinite domain, people often use the equivalent term: $E_{x\sim D}\ln\frac{D(x)}{p(x)} = \mathrm{KL}(D||p)$ which can be finite even when $p(x)$ is a density.

Just as for regression, more assumptions than independence are required in order to verify success in density estimation. There are several tricks commonly used:

(1) Change the loss function to be a bounded loss function. This can be done by thresholding $(\min\{\ln\frac{1}{p(x)}, 50\})$ or choosing an entirely different loss function. Results similar to theorem 2.2 can hold with these modifications.
(2) Give up on proving guarantees for some finite number of samples, and instead work with asymptotic guarantees. These results are not directly applicable to benchmark design.
(3) Assume that $D(x)$ has some particular parametric form.

In considering distribution-free estimates of the log loss, it is important to keep a simple example of two distributions indistinguishable with m samples in mind. Let:

$$D(x) \sim N(0,1)$$

and let:

$$D_m(x) = \text{uniform on } m^{2+\delta} \text{ samples drawn from } D(x)$$

Theorem 2.3. *(Distribution indistinguishability) For every $\delta > 0$, for every halting turing machine $T : X^m \to \{0,1\}$*

$$\lim_{m\to\infty} \Pr_{S\sim D^m}(T(S)=1) - \Pr_{S\sim D_m^m}(T(S)=1) = 0$$

In other, there is no machine capable of distinguishing D and D'_m given just m samples. This is important here because $E_{x\sim D} \ln \frac{D(x)}{p(x)}$ can differ radically from $E_{x\sim D_m} \ln \frac{D_m(x)}{p(x)}$. In particular, if $p(x)$is a density and the samples come from D_m, then the expectation is infinite. On the other hand, when the samples come form D, it can easily be finite.

2.4. **Prediction Domain Conclusion.** The discussion above has some implications for benchmark design. Aside from focusing on prediction problems, we see that some domains of prediction problems are inherently easier to evaluate than other domains. In particular, domains where errors on each individual example have a bounded loss are tractable with only an independence assumption over a set of evaluating examples. Of the domains discussed above, classification performance is (typically) more easily evaluated than regression which is more easily evaluated than density estimation.

It is important to keep in mind that all estimates of prediction ability are subject to some (unverifiable) assumption. In particular, with no assumption, we can not rule out the possibility that the world knows your prediction algorithm and intentionally breaks it for the future. The independence assumption is one way to force the future world to behave in a manner consistent with the past world.

3. A Choice of Settings

Let's consider classification as a prototypical bounded-loss learning setting. Even when we specialize to classification, there are several varieties of classification settings, each of which has implications on the ease or difficulty of benchmarking. We'll discuss each of these settings next and how results for these settings relate to benchmark choices.

3.1. **The Batch Classification Setting.** The batch classification setting is the most prototypical. In the batch setting, a learning algorithm $A : (X \times Y)^* \to (X \to Y)$ receives a set of examples $(x,y)^m$and produces a classifier $c : x \to y$. The goal in this setting is to produce a classifier with a small error rate with respect to some (unknown) measure.

Evaluation in this setting is quite easy. You simply reserve a set of examples m_{test} for use as a test set. When these m_{test} examples are drawn independently, theorem 3.2 gives a high probability bound on the error rate of a chosen classifier.

One variant of the batch setting "K-fold cross validation" is in common use. The K-fold cross validation procedure divides the data into K (roughly) equal sized sets, then trains on $K-1$ of the sets while testing on the held-out set. A precise upper bound on the error rate of some classifier is tricky to state. One result is that the classifier which randomizes over the K learned classifiers has an upper bound on the error rate at least as good as 3.2. See [1] for details.

One difficulty with the cross validation method is that it ignores time dependence in data. For example, suppose our goal was predicting if the stock market was going to rise or fall given past data. In this setting, K-fold cross validation makes little sense because (for example) predicting the performance in May given the performance in April and June may result in a highly optimistic estimate of performance.

3.2. **The Online Classification Setting.** In the online classification setting, a learning algorithm A iterates through a loop:

(1) Receive unlabeled example x.
(2) Predict $\hat{y} \in \{0, 1\}$
(3) Receive label y for x, update internal state and go to 1.

The online learning setting is somewhat more difficult to evaluate than the batch classification setting. Part of the difficulty is even knowing what exactly to evaluate.

Do we care about the prediction performance of the last classifier produced? If so, we can freeze the online algorithm state by skipping step 3, and evaluate it just as we would a classifier produced by a batch classification algorithm.

Another intriguing alternative is to use the error rate of the online learning process in order to bound the error rate of an implicit classifier. Suppose the sequence of classifiers implicitly making predictions at step 2 are labeled as $c_1, ..., c_m$. Then, we can bound the error rate of the stochastic classifier $c_{PV} \equiv U(\{c_1, ..., c_m\})$ in a manner similar to the bound for the batch classifier. This progress is called "progressive validation" (see [1] for details) and has the following associated theorem:

Theorem 3.1. *(Progressive Verification) For all classifiers c, for all $\delta > 0$, for all distributions D over $X \times \{0, 1\}$, for all online learning algorithms A:*

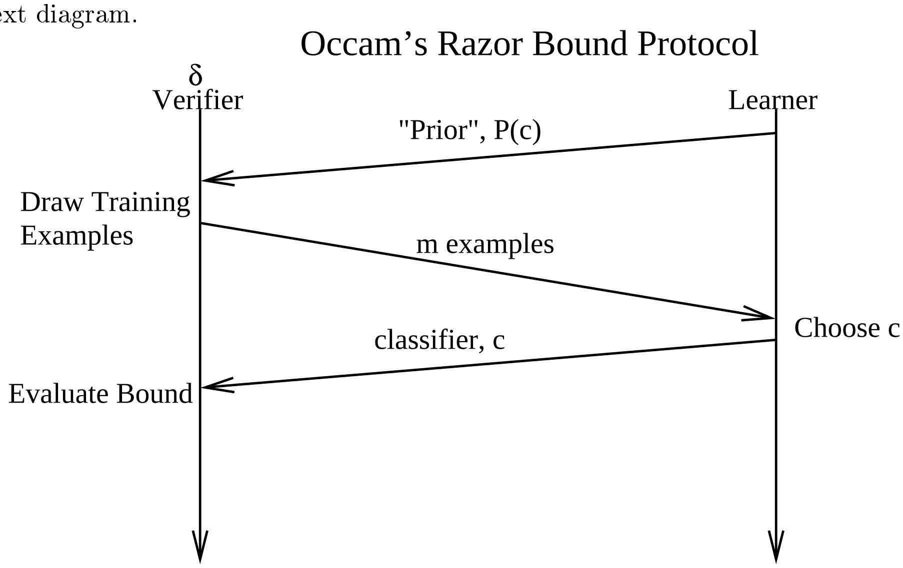
$$\Pr_{S \sim D^m} \left(\left| \Pr_{(x,y) \sim D} (c_{PV}(x) \neq y) - \frac{1}{|S|} \sum_{(x_i, y_i) \in S} I(c_i(x_i) \neq y_i) \right| \leq \sqrt{\frac{\ln \frac{2}{\delta}}{2m}} \right) \geq 1 - \delta$$

The nice thing about this statement is that it "stretches" the available data to allow some portion of the test data to be used as training data. The unnice thing about progressive validation is that the statement is not quantitatively improvable, as in the batch learning setting.

3.3. **The Transductive Classification Setting.** In transduction, the learning algorithm A has access to a set $S = (x, y)^m$ of training examples and a set $x^{m'}$ of unlabeled data. The goal in transduction is to produce a good labeling $y^{m'}$ of *only* the unlabeled data (and notably not for new inputs x). From the viewpoint of a benchmark, it is easy to evaluate performance on a random subset of the m' examples and make predictions about performance across the entire set using theorem 3.2. One thing which is interesting to notice is that the bound is *always* correct since the Independence assumption is explicitly met via the external choice of a random subset. Thus, a bound on the error rate across the m' examples holds in a much stronger sense than the other results discussed here.

The disadvantage of the transduction setting is that it is rare to know in advance exactly which set of instances a prediction is needed for. Since, in the ideal case, good performance on a benchmark immediately implies good performance on real-world problems, this may be a serious concern.

3.4. **The Pure Training Setting.** One alternative to all of the approaches of above is a "pure training" setting. In this setting a learning algorithm A receives a training set $(x, y)^m$ and outputs a classifier $c : X \to \{0, 1\}$. Based upon the *training set performance* and some side information, it is possible to construct a bound on the error rate of a classifier. One example of "side information" might be a commitment to a normalized measure over classifiers $P(c)$ made before the training date is seen. This somewhat more complicated protocol is outlined in the next diagram.



A standard theorem in this setting is slightly more pessimistic and has the form:

Theorem 3.2. *(ORB Verification) For all $\delta > 0$, for all priors $P(c)$ for all distributions D over $X \times \{0, 1\}$:*

$$\Pr_{S\sim D^m}\left(\forall c \ \left|\Pr_{(x,y)\sim D}(c(x) \neq y) - \frac{1}{|S|}\sum_{(x,y)\in S} I(c(x) \neq y)\right| \leq \sqrt{\frac{\ln\frac{1}{P(c)} + \ln\frac{2}{\delta}}{2m}}\right) \geq 1-\delta$$

The verification procedure above is very attractive in the sense that it uses *all* of the data for both learning and verification of learning. Unfortunately, it is sometimes not very tight. Although significant progress has been made recently in making these training set based bounds tight enough for practical use (see [2], [4]), they are not yet tight enough often enough to be relied upon.

One possibly useful alternative is to combined this approach with batch classification (see [3]). A careful combination can allow benefit from good verification performance on the training set while always guaranteeing a reasonably tight bound, as per a test set.

3.5. **Setting Conclusion.** The most straightforward benchmark setting is either transduction (if applicable) or batch learning (otherwise). The online learning setting may be of interest because certain problems inherently fit well. Training set bounds are not mature enough to use on their own, but they may be usefully combined with these other settings.

4. Independence Failures

Since we are discussing prediction performance under the assumption of independently drawn examples, it is important to consider what happens when examples are *not* drawn independently by the world for particular learning problems.

There are two basic forms of failure to be I.I.D.

(1) (Lack of independence) The data is drawn in a correlated fashion from D. In the extreme case, the process generating examples might draw from an independent example, then draw the same example k times, then repeat. When this sort of failure mode occurs, the bounds become too optimistic. Essentially, $m' = \frac{m}{k}$ should be the number of independent draws from D rather than the total number of (dependently drawn) examples m. One standard technique to combat lack of independence is to subsample from the set of test predictions. In particular, if we pick a random subset of size $m'' \ll m'$, with high probability each sample is drawn independently for the above process.

(2) (Lack of identicality) The distribution generating the data shifts over time. A prototypical example of this might be predicting the stock market or learning to drive as you travel from Panama to Alaska. Lack of identicality can result in overoptimistic prediction performance estimates. There are two basic rules of thumb here: 1) Respect the distribution. In particular, do not use cross validation if this is a significant concern. 2) acquire enough data so that we can view this as only a lack of independence.

There is a small amount of research into bounds for nonIID settings, although much work remains to be done.

References

[1] Avrim Blum, Adam Kalai, and John Langford Beating the Holdout: Bounds for KFold and Progressive Cross-Validation. COLT99 pages 203-208.

[2] John Langford, "Practical Prediction Theory for Classification", ICML 2003 tutorial, available at: http://hunch.net/~jl/projects/prediction_bounds/tutorial/tutorial.ps

[3] John Langford, "Combining Train Set and Test Set Bounds", ICML 2002.

[4] Mario Marchand, Mohak Shah, John Shawe-Taylor, and Marina Sokolova The Set Covering Machine with Data-Dependent Half-Spaces, Proceedings of the Twentieth International Conference on Machine Learning (ICML'2003), pp. 520–527,