Dear Luc,

Thanks for the discussion last week about calculating the total variation between unknown distributions given only samples.

I've thought more about the problem, and I came up with a really simple estimation algorithm, `estimate_TV`, that appears to yield close-to-unbiased estimate of TV (I'd like to prove that). The algorithm can be justified by imagining the performance of a Bayes classifier, while recalling that no classifier can have an have accuracy better than $\frac{1+TV}{2}$.

I'd like to show you the algorithm for a couple reasons. First, I want to know whether you've seen this before anywhere, and whether you might be able to point me to a reference where I might find similar approaches. I haven't been able to turn up other methods, but I imagine over the centuries many people have come up with methods. Second, I thought, maybe, you'd find it interesting. At first the algorithm seems strange, and I wouldn't have expected it to be correct were it not for an argument based on the performance of a Bayes classifier.

Problem Statement: Given two sets of (respectively) i.i.d samples $\{P_i\}_i$ and $\{Q_i\}_i$, calculate the "statistical distance" (a.k.a. "total variation", "$L1$-distance") between their underlying distributions. Assume that the P_i's and Q_i's are discrete random variables with the same finite support, $\mathcal{X}$. Other than that, we know nothing about the distributions, not even $|\mathcal{X}|$.

Naive approach

Remember the idea of a naive algorithm for calculating TV: look at the sample frequencies for each "bin", x, in the support $\mathcal{X}$, and take a bin's contribution to TV to be $\frac{|p[x]-q[x]|}{2}$, where $p[x]$ is the number of P_i's in that bin. Total up all these contributions accross all the bins, and you have an (inflated) estimator for TV.

Let me quickly recall one way to see why such an estimate is inflated. Consider a bin x whose actual probability masses under the distributions for for P_i and Q_i are equal. Turning up a sample within such a bin is equally likely whether we sample a P_i or Q_i, but there's still a good chance that, among our samples, we won't actually observe an equal number of P_i's and Q_i's in that bin. Under the naive algorithm, such a sampling error will systematically add to TV, and as we tally accross the support in the naive algorithm, such sampling errors accumulate positive bias in our estimate of TV.

Another way to put it is that slightly different counts, $p[x]$ and $q[x]$, in the bin x, among our samples, isn't strong evidence that the underlying distributions have different probability mass at x. Perhaps we should ignore small frequency differences. Or, since sampling error is the source of different counts accross the entire support, maybe we should somehow penalize the running tally whenever we see a bin whose counts are "close". Of course, we need a principled way to introduce such a penalty.

Principled penalties

First, let me write out the naive algorithm in a somewhat unnatural way. Have a look at the algorithm **naive_estimate_tv**. It's not hard to convince oneself that this is just a reformulated version of the naive method already described. Note that we keep a running tally as before, and for each bin, we add the larger of the two counts ($p[x]$ or $q[x]$) to the tally.

Now, to make a estimator that is slightly conservative, we introduce a penalty whenever we see bins for which the counts of P_i's and Q_i's are the same, or *only off by 1*. This is done in the algorithm **estimate_tv**.

Whereas in the naive algorithm, we always added the *larger* count to the running tally, in **estimate_tv**, we discount that amount by a factor, whenever the counts are equal (or nearly so).

But how do we know what the penalties should be? The penalties come by asking "How well would a Bayes algorithm perform in a leave-one-out validation test?". We imagine doing a series of tests in which we withold one sample each from $\{P_i\}_i$ and $\{Q_i\}_i$. We then show one of the witheld samples to the Bayes classifier, and it guesses whether the sample is a P or Q. During the test, it can access all the other samples, and just guesses the most frequently represented class (P or Q) for the bin of the unknown sample.

The algorithm **estimate_tv** computes the expected accuracy of a Bayes classifier accross all possible tests with given sample sets, which arise by witholding all possible pairs, (P_i, Q_j). Calculating the expected accuracy of a Bayes classifier yields an algorithm that applies penalties when the counts in a bin are equal or nearly equal. We know that the classifier *cannot* do better than $\frac{1+TV}{2}$, so the estimate of TV is at worst conservative. But (I think), absent any prior knowledge about the distributions, a Bayes classifier trained on the full sample sets is the best we can make. This will still fall short of the theoretical "best" classifier, but I think that it will still yield a *good* estimator. I ran a bunch of simulations and it does seem to perform well, and slightly conservatively.

Algorithms

Algorithm 1: naive_estimate_tv(p,q)

input : Counts, p and q, where $p[x]$ ($q[x]$) is the number of samples in $\{P_i\}_i$ ($\{Q_i\}_i$) whose value is x.
output: Estimate of the total variation of the distributions underlying $\{P_i\}_i$ and $\{Q_i\}_i$.
begin
  let *score* be 0
  let k be the total number of samples of P, $\sum_{x \in \mathcal{X}} p[x]$ (which is the same as $\sum_{x \in \mathcal{X}} q[x]$)
  for $x \in \mathcal{X}$ **do**
    let *big* be the larger of $p[x]$ and $q[x]$
    add *big* to *score*
  end
  return $\frac{score}{k} - 1$
end

Algorithm 2: estimate_tv(p,q)

input : Counts, p and q, where $p[x]$ ($q[x]$) is the number of samples in $\{P_i\}_i$ ($\{Q_i\}_i$) whose value is x.
output: Estimate of the total variation of the distributions underlying $\{P_i\}_i$ and $\{Q_i\}_i$.
begin
  let *score* be 0
  let k be the total number of samples of P, $\sum_{x \in \mathcal{X}} p[x]$ (which is the same as $\sum_{x \in \mathcal{X}} q[x]$)
  for $x \in \mathcal{X}$ **do**
    let *big* and *small* be the larger and smaller of $p[x]$ and $q[x]$
    if $big = small$ **then**
      add $\left(\frac{small}{k}\right) \cdot big$ to *score*
    else if $big = small + 1$ **then**
      add $\left(\frac{k+small}{2k}\right) \cdot big$ to *score*
    else
      add *big* to *score*
    end
  end
  return $\frac{score}{k} - 1$
end