Variance Reduction for AB testing Two Policies

Pushpendre Rastogi

September 27, 2020

AB testing is an economically important estimation problem.[1] One application of AB testing is to test the improvement in the new version of a classifier in comparison to the old version. Another application is to test versions of contextual bandits that decide amongst two pieces of content to show to a customer. In both situations we are comparing two policies which can take 1 out of K actions given an input. A common idea often proposed in such a situation is to only use those inputs (a.k.a. samples) in the AB test for which the two policies take different actions. *In this note I quantify the benefit of this idea.*

Why this problem is interesting? I think quantifying the benefit of this idea is interesting because its not clear apriori that removing inputs where the two systems produce the same output will actually improve efficiency. Consider a situation where two policies A and B can chose between action 1 and 2. Let's say that A and B are deterministic and they agree in their recommendation on 50% of the population, and that the total traffic is split 50/50 between the two policies. So out of 100 samples we'll end up throwing out 50 samples and we'll have 25 samples where policy A prescribed the action to take and 25 samples where policiy B prescribed the action. So whatever benefit we get from using only

[1] Data Science can be divided into four areas. Estimation, Inference, Prediction and Control.

Estimation measures a "population level" quantity from samples. E.g. the problem of AB testing is an "estimation" problem. Research in estimation comes up with frameworks for measuring how well an estimator can perform and applied research comes up with procedures for more efficient estimation. For example the statistical concept of the variance of an estimator measures the quality of an estimator and techniques such as control variates, conditioning, importance sampling, and antithetic variables make an estimator more efficient by reducing its variance.

Inference is the problem of finding out the true parameters underlying a data generating process where the data may either be passively observed, or we may need to design an apparatus for collecting observations that is efficient and inexpensive. Techniques such as Maximum likelihood estimation, Pseudo-likelihood, Bayesian Inference (deterministic or sampling based), and the method of moments are a few statistical inference methods.

Prediction is the problem that is directly studied under supervised learning and weakly-supervised learning. Methods such as Deep learning, Kernel methods, and linear classifiers are used in this area.

And, finally, the problem of control is to design policies for taking actions. Techniques in reinforcement-learning are largely focused on learning policies that work well inside MDPs. Some of the most famous policies in game-theory arise from the study of Nash-equilibrium in zero-sum games. The classical control theory uses laplace transforms to study the control problem arising in electro-mechanical systems governed by low-order differential equations.

“clean” samples in our AB test must be greater than the loss we suffered from throwing out the 50 samples where the two policies prescribed the same action. Its this tension between having less samples but which are “cleaner” which makes this problem interesting.

1 The solution

Formal Statement: Let the input X be sampled from some distribution over the input space $\mathcal{X}$. Let Y_A and Y_B, both in the action space $\mathcal{A}$, be the actual actions taken by the policies, and let $M = \mathbb{I}\{Y_A = Y_B\}$ be the random variable that denotes whether the two policies took the same action or not. Let R_A, R_B be two random variables taking values in $\{0, 1\}$. They denote the reward received by policy A, B respectively for input X. Let D be the random variable which measures the difference between reward received by the two policies, i.e. $D = R_A - R_B; D \in \{-1, 0, 1\}$. Let N be the total number of samples. Let d_i denote the value of D for the i-th sample and m_i be the value that M_i takes. We dont observe d_i in practice, only either r_a or r_b but we can observe m_i. For convenience let $q_i = 1 - m_i$ and $\mathrm{Q} = \sum_{i=0}^{N} q_i$ so Q denotes the total times that the the two policies did not match. Note that Q itself is a random variable.

Note that $E[D|M = 1] = 0$ because given that the two policies being considered took the same action they should receive the same reward. Therefore,

$$E[D] = E[E[D|M]] = P(M = 0)E[D|M = 0] + P(M = 1)E[D|M = 1] \tag{1}$$
$$= P(M = 0)E[D|M = 0] \tag{2}$$
$$= E[\mathbb{I}\{M = 0\}]E[D|M = 0] \tag{3}$$

1.1 First Attempt

If we could observe d_i then we could estimate $E[D]$ via three estimators.

E1) $\dfrac{1}{N}\sum_{i=1}^{n} d_i$	E2) $\dfrac{P(M=0)}{\mathrm{Q}/N}\dfrac{1}{N}\sum_{i=1}^{n} d_i q_i$	E3) $\dfrac{\mathrm{Q}}{N}\dfrac{1}{\mathrm{Q}}\sum_{i=1}^{n} d_i q_i = \dfrac{1}{N}\sum_{i=1}^{n} d_i q_i$

E1 has variance $\mathrm{Var}(D)/N$. E3 has variance $\mathrm{Var}(DQ)/N$.

$$\mathrm{Var}(D) = E[\mathrm{Var}(D|Q)] + \mathrm{Var}[E(D|Q)] \tag{4}$$
$$= \mathrm{Var}(D|Q = 1)P(Q = 1) + \mathrm{Var}(D|Q = 0)P(Q = 0) + \big(E(D|Q = 1) - E(D|Q = 0)\big)^2 P(Q = 0)P(Q = 1) \tag{5}$$
$$= \mathrm{Var}(D|Q = 1)P(Q = 1) + \mathrm{Var}(D|Q = 0)P(Q = 0) + \big(E(D|Q = 1)\big)^2 P(Q = 0)P(Q = 1) \tag{6}$$
$$\mathrm{Var}(DQ) = E[\mathrm{Var}(DQ|Q)] + \mathrm{Var}[E(DQ|Q)] \tag{7}$$
$$= \mathrm{Var}(D|Q = 1)P(Q = 1) + \big(E(D|Q = 1)\big)^2 P(Q = 0)P(Q = 1) \tag{8}$$
$$= \mathrm{Var}(D|Q = 1)P(Q = 1) \tag{9}$$

This shows that Var($E1$) is strictly greater than Var($E3$). E2 has an additional factor which acts like a multiplicative control-variate so its variance will be even lower than E3. Alas, we can not actually implement these estimators because d_i is never observed.

1.2 The Solution

Since we can only observe either y_{ai} or y_{bi} but not d_i therefore we can only compute:

$$\frac{1}{N_A}\sum_{i=1}^{N_A} y_{ai} - \frac{1}{N_B}\sum_{i=1}^{N_B} y_{bi} \tag{10}$$

But this just means that the variance will be double the population variance for the true bernoulli rewards.

1.3 Simulation

Consider the following code where I simulate two policies that pick actions 0 with probabilities $0.6, 0.55$ respectively. The reward prob of action 0 is 0.6 and for action 1 reward probability is 0.55. Obviously policy A - policy B $= 0.52 - 0.51 = 0.01$.

```python
import numpy as np
prA0 = 0.6
prA1 = 0.4
piA = 0.6
piB = 0.55
erA = prA0 * piA + prA1 * (1-piA)
vrA = erA * (1 - erA)
erB = prA0 * piB + prA1 * (1-piB)
vrB = erB * (1 - erB)
M = int(1e5)
for N in [int(1e2), int(1e3)]:
    def bern(p):
        return np.random.rand(M, N) < p
    print(’N’, N)
    aA = bern(piA)
    aB = bern(piB)
    rA = bern(prA0) * aA + bern(prA1) * (1-aA)
    rB = bern(prA0) * aB + bern(prA1) * (1-aB)
    print(f’{rA.mean():.4f}, {erA:.4f}, {np.var(rA):.4f}, {vrA:.4f}’)
    print(f’{rB.mean():.4f}, {erB:.4f}, {np.var(rB):.4f}, {vrB:.4f}’)
    tmp = rA[:, :N//2].mean(1) - rB[:, N//2:].mean(1)
    var1 = np.var(tmp)
    print(’Var E1’, np.abs(0.01 - np.mean(tmp)), var1)
    q = (aA != aB)
    tmp = (rA * q)[:, :N//2].mean(1) - (rB * q)[:, N//2:].mean(1)
    var2 = np.var(tmp)
    print(’Var E3’, np.abs(0.01 - np.mean(tmp)), var2)
    print(f’relative savings in samples {100*(1 - var2 / var1):.2f}%’)
    print()
```

Results

```
N 100
0.5200, 0.5200, 0.2496, 0.2496
0.5099, 0.5100, 0.2499, 0.2499
Var E1 0.00040560000000000075 0.009962259488640001
Var E3 0.0004542000000000001 0.007365961702359999
relative savings in samples 26.06%

N 1000
0.5200, 0.5200, 0.2496, 0.2496
0.5099, 0.5100, 0.2499, 0.2499
Var E1 0.00017972000000000474 0.001001855060721601
Var E3 0.00017492000000000028 0.0007403123629936004
relative savings in samples 26.11%
```

We can see that the Estimator 3 has 25% less variance than Estimator 1 because it conditions on the decisions. Also note that the above estimators are realistic because they do not assume access to D. Instead they assume that policy A and policy B is tried on distinct instances.