LINEAR TIME APPROXIMATION OF THE TV DISTANCE BETWEEN PRODUCT DISTRIBUTIONS

KONRAD ANAND, ALISTAIR BENFORD, AND HENG GUO

Abstract. We present a linear time approximation algorithm of the total variation distance between two product distributions. The main algorithm was found using ChatGPT 5.6 Sol Ultra.

1. Introduction

The total variation (TV) distance is an important statistic with wide-ranging applications. However, the computation of TV distances is not easy, even for simple distributions such as product distributions. Unlike the Kullback-Leibler divergence (also known as relative entropy), the TV distance does not tensorise over product distributions. In fact, the exact evaluation for product distributions turns out to be #**P**-hard [BGM+25a].

It is then natural to ask for approximation. Chernoff bounds imply relatively straightforward efficient additive error approximations. (See, e.g. [PTXZ26] and the references therein for additively approximate TV distances in more advanced models.) Instead, we focus on relative errors. In this context, the first efficient (randomised) approximation algorithm was given by Feng, Guo, Jerrum, and Wang [FGJW23], which has inspired a number of follow-up works in various settings [BGM+24, BGM+25b, BFS25, FLY25, FF26, FFYZ26]. Most notably, Feng, Liu, and Liu [FLL24] derandomised the algorithm in [FGJW23].

In this note, we give a linear time approximation algorithm of the total variation distance between two product distributions, improving upon [FGJW23].

Theorem 1. *Let $q, n > 0$ be two integers, and $P = \bigotimes_{i=1}^{n} P_i$ and $Q = \bigotimes_{i=1}^{n} Q_i$ be two product distributions on $[q]^n$, given explicitly by the marginal probabilities of each coordinate. For any $0 < \varepsilon \leq 1$ and $0 < \delta < 1$, there is a randomised algorithm that outputs a nonnegative random variable Z satisfying*

$$\Pr\left[(1-\varepsilon)d_{\mathrm{TV}}(P,Q) \leq Z \leq (1+\varepsilon)d_{\mathrm{TV}}(P,Q)\right] \geq 1-\delta$$

in $O\left(qn\,\varepsilon^{-2}\log\frac{1}{\delta}\right)$ time.

For Theorem 1, real arithmetic operations and generating a uniform real from $(0,1)$ are considered unit-cost, and P and Q are given by the marginals p_{ij} and q_{ij} for $i \in [n]$ and $j \in [q]$. Thus, for fixed ε and δ, the running time of Theorem 1 is linear in the input size. Previously, the algorithm in [FGJW23] states an $O\left(\frac{qn^2}{\varepsilon^2}\log\frac{1}{\delta}\right)$ time, although it can be improved to $O\left(\frac{qn^{1.5}}{\varepsilon^2}\log\frac{1}{\delta}\right)$ time using a bound given by Kontorovich [Kon25]. The deterministic algorithm of Feng, Liu, and Liu runs in time $O\left(\frac{qn^2}{\varepsilon}\log q\log\frac{n}{\varepsilon d_{\mathrm{TV}}(P,Q)}\right)$. Since the ideas of [FGJW23] have been found useful in various other settings, we hope the ideas of Theorem 1 will find wide applications too.

The algorithm in [FGJW23] relies on an estimator based on couplings, and the algorithm in [FLL24] is based on the idea of careful discretisation. Instead, the algorithm in Theorem 1 uses

School of Informatics, University of Edinburgh, Informatics Forum, Edinburgh, EH8 9AB, United Kingdom

very different ideas. Essentially, the improved running time comes from an adaptation of the filtered Monte Carlo technique [Gla93].

In Section 4, we present an alternative implementation of Theorem 1, which runs in time $O\left(nq\log q + \frac{n\log q}{\varepsilon^2}\log\frac{1}{\delta}\right)$. This is faster if $\varepsilon^{-2}\log q + \frac{\log q}{q} = o(1)$. On the other hand, we also present an $\Omega(qn)$ lower bound in Theorem 5.

In fact, the authors of this note first set out to improve the run-time bounds in [FGJW23], and found a roughly $O(qn^{1.3})$ time algorithm for the product of Bernoulli distributions. Shortly after that, the authors tried ChatGPT 5.6 Sol Ultra, which managed to find the linear-time (and much more elegant) algorithm in Theorem 1 with a single prompt.[1] The prompt is adapted from OpenAI's cycle double cover prompt by only changing the problem statement. However, the key reference [Gla93] did not emerge until after a few more rounds of interaction with ChatGPT. In the note we still use words like "we" or "our", mostly out of habit, although the main ideas actually come from ChatGPT. The proofs are written or rewritten by the authors, who assume all responsibility for errors.

2. The algorithm

Let Ω be the state space, which in our setting is $[q]^n$. We may assume $d_{\mathrm{TV}}(P,Q) \neq 0$, as it is easy to rule out otherwise at the start of the algorithm. The starting point of the algorithm is to rewrite

$$\begin{aligned} d_{\mathrm{TV}}(P,Q) &= \frac{1}{2}\sum_{\omega\in\Omega}|P(\omega)-Q(\omega)| \\ &= \sum_{\omega\in\Omega}\frac{P(\omega)+Q(\omega)}{2}\frac{|P(\omega)-Q(\omega)|}{P(\omega)+Q(\omega)}. \end{aligned} \tag{1}$$

Therefore, if we sample ω from the mixture $\frac{P+Q}{2}$, then $\frac{|P(\omega)-Q(\omega)|}{P(\omega)+Q(\omega)}$ is an unbiased estimator. However, this estimator has a large relative variance, especially when $d_{\mathrm{TV}}(P,Q)$ is small. The main point of our algorithm is to overcome this issue by designing a different estimator.

We use a filtered Monte Carlo idea [Gla93]. The algorithm first draws a hidden fair hypothesis $H \in \{+,-\}$. Then, draw a sample from P if $H = +$ and from Q otherwise. Equivalently, this is drawing from $\frac{P+Q}{2}$. Call the samples $X = (X_1,\ldots,X_n)$. After each observed coordinate, maintain the posterior probabilities

$$\alpha_i = \Pr(H = + \mid \mathcal{F}_i) \qquad \text{and} \qquad \beta_i = 1-\alpha_i,$$

where the filtration $\mathcal{F}_i$ is the σ-algebra generated by $X_1,\ldots,X_i$. It is important that $\mathcal{F}_i$ excludes the randomness of H. If the sample up to coordinate i is $\omega_{\le i}$, then

$$\alpha_i = \frac{P(\omega_{\le i})}{P(\omega_{\le i})+Q(\omega_{\le i})}.$$

The posterior bias $U_i = \alpha_i - \beta_i$ is a bounded martingale. Its terminal absolute value, by (1), has expectation exactly equal to the total variation distance:

$$\mathbb{E}\,|U_n| = d_{\mathrm{TV}}(P,Q).$$

Instead of using $|U_n|$ itself, whose relative variance can be large when the distance is small, the algorithm sums the predictable increases of the submartingale $|U_i|$:

$$C_i := \mathbb{E}\left[|U_i| - |U_{i-1}| \mid \mathcal{F}_{i-1}\right] = \mathbb{E}[|U_i| \mid \mathcal{F}_{i-1}] - |U_{i-1}|.$$

[1] The raw output can be found here: https://chatgpt.com/s/t_6a6e380155a0819184ee91915136108e.

The final estimator is $A := \sum_{i=1}^{n} C_i$. The way we compute A is given by Algorithm 1, where we denote $(x)_+ := \max\{x, 0\}$.

Algorithm 1 The estimator

Draw uniformly at random a sign $H \in \{+, -\}$. ▷ The same sign is used for all coordinates.
Initialize $\alpha \leftarrow \frac{1}{2}$, $\beta \leftarrow \frac{1}{2}$, and $A \leftarrow 0$.
for $i = 1, \dots, n$ **do**
    Before observing coordinate i, compute

$$C_i = \begin{cases} 2\sum_{c=1}^{q} \big(\beta Q_i(c) - \alpha P_i(c)\big)_+, & \text{if } \alpha \geq \beta, \\ 2\sum_{c=1}^{q} \big(\alpha P_i(c) - \beta Q_i(c)\big)_+, & \text{if } \alpha < \beta. \end{cases} \tag{2}$$

    $A \leftarrow A + C_i$.
    if $H = +$ **then**
        Sample X_i from P_i.
    else
        Sample X_i from Q_i.
    end if
    Update simultaneously

$$\alpha \leftarrow \frac{\alpha P_i(X_i)}{\alpha P_i(X_i) + \beta Q_i(X_i)} \qquad \text{and} \qquad \beta \leftarrow \frac{\beta Q_i(X_i)}{\alpha P_i(X_i) + \beta Q_i(X_i)}.$$

end for
return A.

The estimator A is nonnegative and unbiased.

Lemma 2. *Let A be as in Algorithm 1. Then, $\mathbb{E}[A] = d_{\mathrm{TV}}(P, Q)$.*

The key property of A is the following second moment bound.

Lemma 3. *Let A be as in Algorithm 1. Then, $\mathbb{E}\, A^2 \leq 2 d_{\mathrm{TV}}(P, Q)^2$, which implies that the relative variance $\frac{\mathrm{Var}(A)}{(\mathbb{E}\, A)^2} \leq 1$.*

Given Lemma 2 and Lemma 3, we can prove Theorem 1.

Proof of Theorem 1. Let $s := \lceil \frac{4}{\varepsilon^2} \rceil$ and r be the smallest odd integer with $r \geq 8 \log \frac{1}{\delta}$. Run s independent estimators using Algorithm 1, and take the mean. Repeat this process r times and output the median of all r means. This is the standard median of means estimator. The approximation guarantee follows from Lemma 2, Lemma 3, Chebyshev's inequality, and the Chernoff bound.

As for the running time, it is not hard to see that the time to generate one estimator A is $O(nq)$, and the theorem holds. □

3. The analysis

The analysis relies on the following lemma, which holds for arbitrary distributions, not only products.

Lemma 4. *Let R and S be probability distributions on a finite set Ω. Let $\alpha, \beta \geq 0$ satisfy $\alpha+\beta = 1$, and $u := \alpha - \beta$. Define*

$$V_{R,S}(u) := \sum_{\omega\in\Omega} |\alpha R(\omega) - \beta S(\omega)| - |u|.$$

Then

$$V_{R,S}(u) = \begin{cases} 2\sum_{\omega\in\Omega} \big(\beta S(\omega) - \alpha R(\omega)\big)_+, & \text{if } \alpha \geq \beta, \\ 2\sum_{\omega\in\Omega} \big(\alpha R(\omega) - \beta S(\omega)\big)_+, & \text{if } \alpha < \beta, \end{cases} \tag{3}$$

and

$$0 \leq V_{R,S}(u) \leq (1 - |u|) d_{\mathrm{TV}}(R, S). \tag{4}$$

Proof. We first assume $u \geq 0$, so $\alpha \geq \beta$. Note that for any numbers $a_1, \ldots, a_m$,

$$\sum_{i=1}^m |a_i| - \sum_{i=1}^m a_i = \sum_{i=1}^m ((a_i)_+ + (-a_i)_+) - \sum_{i=1}^m ((a_i)_+ - (-a_i)_+) = 2\sum_{i=1}^m (-a_i)_+.$$

Therefore, it holds that

$$\begin{aligned} V_{R,S}(u) &= \sum_{\omega\in\Omega} |\alpha R(\omega) - \beta S(\omega)| - (\alpha - \beta) = \sum_{\omega\in\Omega} |\alpha R(\omega) - \beta S(\omega)| - \sum_{\omega\in\Omega} (\alpha R(\omega) - \beta S(\omega)) \\ &= 2\sum_{\omega\in\Omega} \big(\beta S(\omega) - \alpha R(\omega)\big)_+, \end{aligned}$$

which implies (3) and the lower bound of (4). Moreover,

$$\beta S(\omega) - \alpha R(\omega) = \beta\big(S(\omega) - R(\omega)\big) - (\alpha - \beta) R(\omega) \leq \beta\big(S(\omega) - R(\omega)\big).$$

It follows that

$$\begin{aligned} V_{R,S}(u) &\leq 2\beta \sum_{\omega\in\Omega} \big(S(\omega) - R(\omega)\big)_+ \\ &= 2\beta d_{\mathrm{TV}}(R, S) = (1 - u) d_{\mathrm{TV}}(R, S). \end{aligned}$$

The case $u \leq 0$ is symmetric after exchanging R and S. The lemma follows. □

We now analyse the first moment and show Lemma 2.

Proof of Lemma 2. Recall the definitions of α_i, β_i, and U_i. In Algorithm 1, at coordinate i, the probability of drawing $c \in [q]$ is $\alpha_{i-1} P_i(c) + \beta_{i-1} Q_i(c)$. Thus, by Bayes's rule,

$$\alpha_i = \frac{\alpha_{i-1} P_i(c)}{\alpha_{i-1} P_i(c) + \beta_{i-1} Q_i(c)}, \qquad \beta_i = \frac{\beta_{i-1} Q_i(c)}{\alpha_{i-1} P_i(c) + \beta_{i-1} Q_i(c)}.$$

Note that this is consistent with the update rule in Line 11. Thus, inductively, the values α and β at the end of iteration i are exactly α_i and β_i.

Since $\mathbb{E}\left[|U_i| \mid \mathcal{F}_{i-1}\right] = \sum_{c=1}^q |\alpha_{i-1} P_i(c) - \beta_{i-1} Q_i(c)|$, identifying $R = P_i$ and $S = Q_i$ in (3) yields $C_i = \mathbb{E}\left[|U_i| \mid \mathcal{F}_{i-1}\right] - |U_{i-1}|$, where C_i is the value in Algorithm 1. Taking expectations and telescoping gives $\mathbb{E}\, A = \mathbb{E}\, |U_n|$. For every ω so that $P(\omega) + Q(\omega) > 0$,

$$|U_n(\omega)| = \frac{|P(\omega) - Q(\omega)|}{P(\omega) + Q(\omega)}.$$

The probability of drawing ω is $\frac{P(\omega)+Q(\omega)}{2}$. Therefore,

$$\mathbb{E}\, A = \sum_{\omega\in\Omega} \frac{P(\omega) + Q(\omega)}{2} \frac{|P(\omega) - Q(\omega)|}{P(\omega) + Q(\omega)} = d_{\mathrm{TV}}(P, Q).$$ □

Finally, we complete the proof by analysing the second moment.

Proof of Lemma 3. Let $P_{i:n} = \bigotimes_{j=i}^{n} P_j$ and $Q_{i:n} = \bigotimes_{j=i}^{n} Q_j$. Since P and Q are product distributions, conditioned on $\mathcal{F}_{i-1}$, the remaining experiment consists of distinguishing $P_{i:n}$ versus $Q_{i:n}$ with current prior bias U_{i-1}. By telescoping conditional expectations and applying (4),

$$\mathbb{E}\left[\sum_{j=i}^{n} C_j \,\middle|\, \mathcal{F}_{i-1}\right] = V_{P_{i:n},Q_{i:n}}(U_{i-1}) \le d_{\mathrm{TV}}(P_{i:n}, Q_{i:n}) \le d_{\mathrm{TV}}(P, Q), \tag{5}$$

where the last inequality follows because projecting on the $i, \dots, n$ coordinates cannot increase the TV distance.

Next we rewrite

$$A^2 = 2\sum_{i=1}^{n} C_i \sum_{j=i}^{n} C_j - \sum_{i=1}^{n} C_i^2.$$

The key point is that C_i is completely determined by $\mathcal{F}_{i-1}$. Thus,

$$\mathbb{E}\left[C_i \sum_{j=i}^{n} C_j \,\middle|\, \mathcal{F}_{i-1}\right] = \mathbb{E}\left[C_i \mid \mathcal{F}_{i-1}\right] \mathbb{E}\left[\sum_{j=i}^{n} C_j \,\middle|\, \mathcal{F}_{i-1}\right]. \tag{6}$$

Therefore,

$$\begin{aligned}
\mathbb{E}\, A^2 &= 2\sum_{i=1}^{n} \mathbb{E}\left[C_i \sum_{j=i}^{n} C_j\right] - \sum_{i=1}^{n} \mathbb{E}\, C_i^2 \\
&= 2\sum_{i=1}^{n} \mathbb{E}\left[\mathbb{E}\left[C_i \sum_{j=i}^{n} C_j \,\middle|\, \mathcal{F}_{i-1}\right]\right] - \sum_{i=1}^{n} \mathbb{E}\, C_i^2 \\
&= 2\sum_{i=1}^{n} \mathbb{E}\left[\mathbb{E}\left[C_i \mid \mathcal{F}_{i-1}\right] \mathbb{E}\left[\sum_{j=i}^{n} C_j \,\middle|\, \mathcal{F}_{i-1}\right]\right] - \sum_{i=1}^{n} \mathbb{E}\, C_i^2 && \text{(by (6))} \\
&\le 2d_{\mathrm{TV}}(P, Q) \sum_{i=1}^{n} \mathbb{E}\left[\mathbb{E}\left[C_i \mid \mathcal{F}_{i-1}\right]\right] - \sum_{i=1}^{n} \mathbb{E}\, C_i^2 && \text{(by (5))} \\
&\le 2d_{\mathrm{TV}}(P, Q) \sum_{i=1}^{n} \mathbb{E}\, C_i = 2d_{\mathrm{TV}}(P, Q)\, \mathbb{E}\, A \\
&= 2d_{\mathrm{TV}}(P, Q)^2. && \text{(by Lemma 2)}
\end{aligned}$$

The lemma follows. □

4. Further improved run-time and limits

We first present an alternative way of implementing Theorem 1, which is faster if $\varepsilon^{-2} \log q + \frac{\log q}{q} = o(1)$. The key point is the computation of (2) mainly cares about what $c \in [q]$ makes $\frac{P_i(c)}{Q_i(c)} \ge \frac{\beta}{\alpha}$. We may thus first sort for each i, the ratio $\frac{P_i(c)}{Q_i(c)}$. Suppose the resulting ordering for coordinate i is $x_{i,1}, \dots, x_{i,q}$. Then compute the suffix sum $\sum_{j \ge t} P_i(x_{i,j})$ for every $i \in [n]$ and $t \in [q]$, and similarly for the prefix sum, and for Q_i's. Then, at each iteration i, we use a binary search to locate the threshold, and then use the corresponding prefix or suffix sums to compute C_i. The preprocessing

per coordinate takes time $O(q \log q)$, and the binary search takes time $O(\log q)$. The overall runtime is $O\left(nq \log q + \frac{n \log q}{\varepsilon^2} \log \frac{1}{\delta}\right)$.

On the other hand, it is straightforward that any efficient approximation needs to read a constant fraction of the input at least. Here we include a proof for completeness. Our model is that P and Q are given by the $2qn$ marginals, and that the algorithm queries these marginals.

Theorem 5. *Let $n \geq 1$, $q \geq 2$, $0 < \varepsilon < 1$, and $0 \leq \delta < 1/2$. In the marginal-query model, every randomised algorithm that outputs a multiplicative $(1 \pm \varepsilon)$-approximation to $d_{\mathrm{TV}}(P, Q)$ with probability at least $1 - \delta$ for any two product distributions P and Q on $[q]^n$ must make at least*

$$(1 - 2\delta)n \left\lfloor \frac{q}{2} \right\rfloor$$

queries in the worst case. In particular, its running time is $\Omega(nq)$.

Proof. Let $m := \left\lfloor \frac{q}{2} \right\rfloor$, $N := nm$, and $\eta := \frac{1}{2q}$. Let U be the uniform distribution on $[q]$, and let $P = Q^{(0)} := U^{\otimes n}$. For each $(i, j) \in [n] \times [m]$, define $Q^{(i,j)}$ by taking all marginals other than the i-th one to be uniform and setting

$$Q_i^{(i,j)}(2j - 1) = \frac{1}{q} + \eta, \qquad Q_i^{(i,j)}(2j) = \frac{1}{q} - \eta,$$

with all other entries of $Q_i^{(i,j)}$ equal to $1/q$. For any $(i, j) \in [n] \times [m]$, only one marginal of $Q^{(i,j)}$ differs from P, and hence

$$d_{\mathrm{TV}}(P, Q^{(i,j)}) = d_{\mathrm{TV}}(U, Q_i^{(i,j)}) = \eta.$$

In contrast, $d_{\mathrm{TV}}(P, Q^{(0)}) = 0$.

Fix the internal randomness of the algorithm, and consider its execution on $Q^{(0)}$. Suppose this execution makes at most T queries. Let $S \subseteq [n] \times [m]$ contain (i, j) whenever the algorithm queries at least one of the cells $Q_i(2j - 1)$ and $Q_i(2j)$. Then $|S| \leq T$. For every $(i, j) \notin S$, the execution log on $Q^{(i,j)}$ is identical to the log on $Q^{(0)}$. It means that the algorithm returns the same value on these two inputs.

Consider a random input that is $Q^{(0)}$ with probability $1/2$ and is a uniformly random $Q^{(i,j)}$ with probability $1/2$. If the algorithm outputs zero on $Q^{(0)}$, then it fails on every $Q^{(i,j)}$ with $(i, j) \notin S$. Its average success probability is therefore at most

$$\frac{1}{2} + \frac{|S|}{2N} \leq \frac{1}{2} + \frac{T}{2N}.$$

If it outputs a nonzero value on $Q^{(0)}$, then it fails on $Q^{(0)}$, so its average success probability is at most $1/2$. Thus, in every case, the average success probability is at most $\frac{1}{2} + \frac{T}{2N}$.

The same bound holds after averaging over the internal randomness. Since the algorithm succeeds with probability at least $1 - \delta$ for any pair of P and Q, we have that $1 - \delta \leq \frac{1}{2} + \frac{T}{2N}$. Thus,

$$T \geq (1 - 2\delta)N = (1 - 2\delta)n \left\lfloor \frac{q}{2} \right\rfloor. \qquad \square$$

Acknowledgement

We thank Weiming Feng for some helpful comments. Part of this project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 947778).

References

[BFS25] Arnab Bhattacharyya, Weiming Feng, and Piyush Srivastava. Approximating the total variation distance between Gaussians. In *AISTATS*, pages 1846–1854, 2025. 1

[BGM+24] Arnab Bhattacharyya, Sutanu Gayen, Kuldeep S. Meel, Dimitrios Myrisiotis, A. Pavan, and N. V. Vinodchandran. Total variation distance meets probabilistic inference. In *ICML*, pages 3776–3794, 2024. 1

[BGM+25a] Arnab Bhattacharyya, Sutanu Gayen, Kuldeep S. Meel, Dimitrios Myrisiotis, A. Pavan, and N. V. Vinodchandran. Total variation distance for product distributions is #P-complete. *Inf. Process. Lett.*, 189:106560, 2025. 1

[BGM+25b] Arnab Bhattacharyya, Sutanu Gayen, Kuldeep S. Meel, Dimitrios Myrisiotis, Aduri Pavan, and N. V. Vinodchandran. Computational explorations of total variation distance. In *ICLR*, 2025. 1

[FF26] Weiming Feng and Yucheng Fu. On approximating the f-divergence between two Ising models. In *ITCS*, volume 362 of *LIPIcs*, pages 59:1–59:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2026. 1

[FFYZ26] Weiming Feng, Yucheng Fu, Minji Yang, and Anqi Zhang. On computing total variation distance between mixtures of product distributions. *arXiv*, abs/2605.03839, 2026. 1

[FGJW23] Weiming Feng, Heng Guo, Mark Jerrum, and Jiaheng Wang. A simple polynomial-time approximation algorithm for the total variation distance between two product distributions. *TheoretiCS*, 2:Paper No. 7, 2023. 1, 2

[FLL24] Weiming Feng, Liqiang Liu, and Tianren Liu. On deterministically approximating total variation distance. In *SODA*, pages 1766–1791, 2024. 1

[FLY25] Weiming Feng, Hongyang Liu, and Minji Yang. Approximating the total variation distance between spin systems. In *COLT*, pages 1974–2025, 2025. 1

[Gla93] Paul Glasserman. Filtered Monte Carlo. *Math. Oper. Res.*, 18(3):610–634, 1993. 2

[Kon25] Aryeh Kontorovich. On the tensorization of the variational distance. *Electron. Commun. Probab.*, 30:Paper No. 32, 10, 2025. 1

[PTXZ26] Eric Price, Kevin Tian, Zhiyang Xun, and Yusong Zhu. Total variation distance estimation in autoregressive models. *arXiv*, abs/2607.19510, 2026. 1