

Momentum Matters: Comparing Optimisers on Two Moons

Georgios Kiritsis

October 15, 2026

Abstract

We train one small neural network on the two-moons problem with three optimisers: stochastic gradient descent, SGD with momentum, and Adam, each over 5 random seeds. On 400 held-out points their mean accuracies span 1.4 percentage points (the best, Momentum, reaches $(96.0 \pm 0.3)\%$), and they differ even more in speed: the fastest, Momentum, brings the training loss below 0.1 in 26 ± 3 epochs. Every figure, table and number in this paper is produced by `notebooks/experiment.py` and updates when the notebook changes.

1 Introduction

Choosing an optimiser is one of the first decisions in any machine-learning experiment, and one of the easiest to get wrong by habit. Plain stochastic gradient descent [3] is simple and well understood; momentum [2] and Adam [1] promise faster progress at the cost of extra state and extra hyperparameters. This note compares the three on a problem small enough to rerun in under a second, so the comparison can be checked rather than taken on trust.

Section 2 describes the model and the three update rules, Section 3 reports what happened, and Section 4 says what we take from it.

2 Method

Data. The two-moons problem consists of two interleaving half circles with Gaussian noise added to every point. We draw 400 points for training and 400 for testing.

Model. The classifier is a network with one hidden layer of $h = 16$ tanh units and a sigmoid output, $2h + h + h + 1 = 65$ parameters in all, trained with the cross-entropy loss

$$\mathcal{L}(\theta) = -\frac{1}{N} \sum_{i=1}^N \left[y_i \log p_{\theta}(x_i) + (1 - y_i) \log (1 - p_{\theta}(x_i)) \right]. \quad (1)$$

Optimisers. For each of the 5 seeds, all three start from the same weights and see the same mini-batches of 32 points for 100 epochs; we report the mean and standard deviation over seeds. Writing $g_t = \nabla_{\theta} \mathcal{L}(\theta_t)$ for the mini-batch gradient and η for the learning rate, SGD takes the step

$$\theta_{t+1} = \theta_t - \eta g_t, \quad (2)$$

momentum accumulates a velocity,

$$v_{t+1} = \mu v_t + g_t, \quad \theta_{t+1} = \theta_t - \eta v_{t+1}, \quad (3)$$

with $\mu = 0.9$, and Adam rescales each coordinate by running estimates of the first and second moments of the gradient,

$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}}, \quad (4)$$

with the default decay rates $\beta_1 = 0.9$ and $\beta_2 = 0.999$ of [1].

3 Results

Figure 1 shows the training loss of the three runs and Table 1 summarises them. The update rules of Equations 2–4 differ most in how fast they make progress: Momentum crosses a training loss of 0.1 after 26 ± 3 epochs, and an optimiser with “–” in the last column of the table never got there in some run.

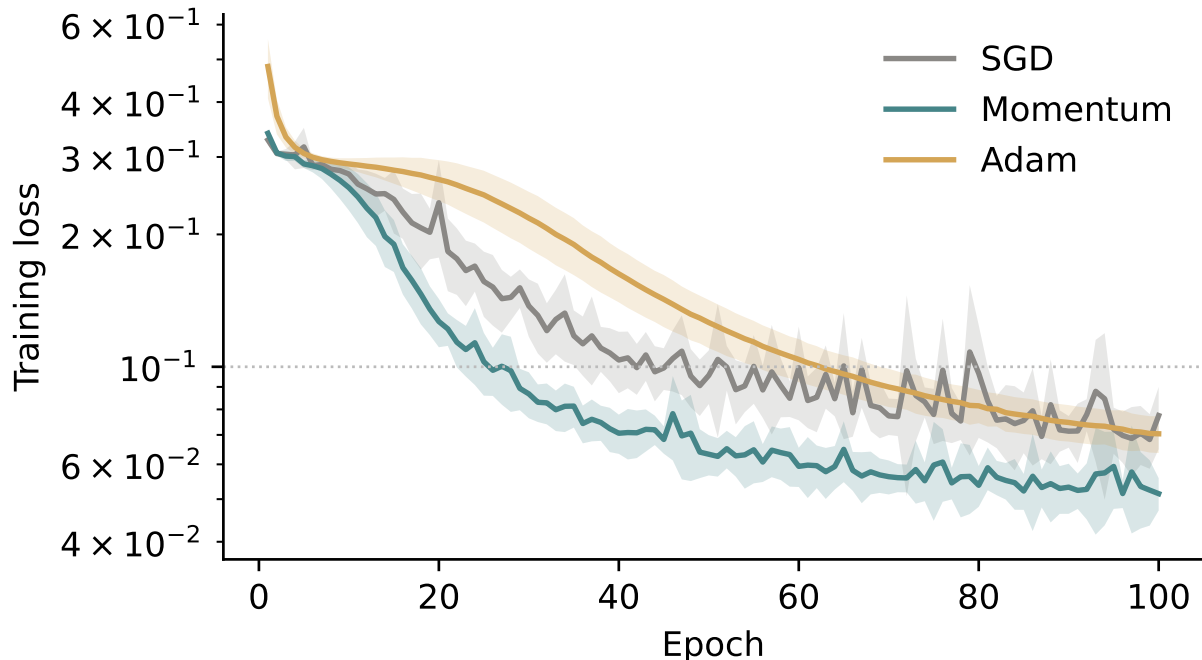


Figure 1: Training loss per epoch on a logarithmic scale: mean over 5 seeds, with a band of one standard deviation. The dotted line marks 0.1.

Optimiser	Learning rate	Final loss	Test acc. (%)	Epochs to 0.1
SGD	0.6	0.077 ± 0.012	94.7 ± 0.7	43 ± 8
Momentum	0.1	0.051 ± 0.004	96.0 ± 0.3	26 ± 3
Adam	0.01	0.070 ± 0.007	96.0 ± 0.3	63 ± 8

Table 1: Final training loss, test accuracy, and the first epoch at which the training loss fell below 0.1, as mean $\pm$ standard deviation over 5 seeds (“–” if some run never got there).

Test accuracy follows from that. The mean accuracies of the three optimisers span 1.4 points, with Momentum ahead at $(96.0 \pm 0.3)\%$. Where the standard deviations overlap the ranking should not be read as significant, and an optimiser that has not finished converging is being compared on speed as much as on the quality of its solution. Figure 2 shows the decision boundary one of those runs learned.

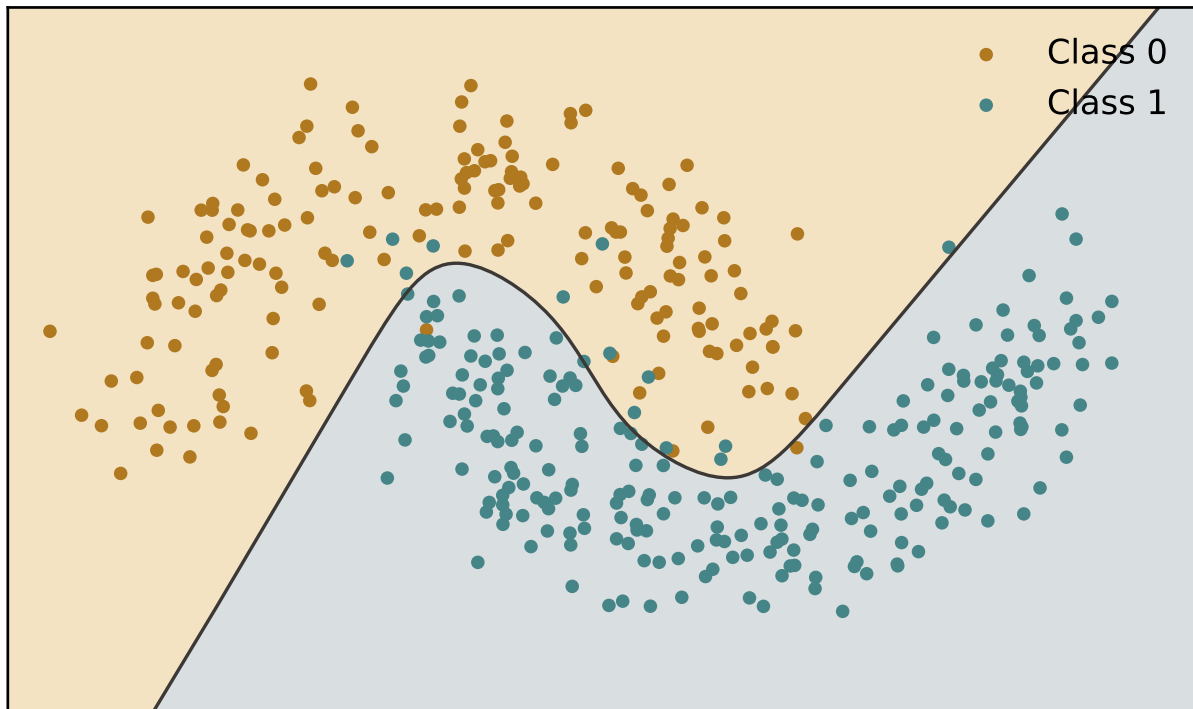


Figure 2: Decision boundary of one Momentum run, with the test points.

4 Conclusion

On this problem, with a fixed budget of 100 epochs, the choice of optimiser decided how much of the training got done within it. That is a statement about one small network and one dataset; the notebook makes it cheap to find out how far it carries, and Appendix A records exactly how these numbers were produced.

References

- [1] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.
- [2] Boris T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.
- [3] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

A Reproducibility

Table 2 is written by the notebook each time it runs.

Notebook	<code>notebooks/experiment.py</code>
Python	3.14.6
Packages	numpy 2.5.1, matplotlib 3.11.2
Random seed	0, 1, 2, 3, 4
Git commit	<code>df78e9c + uncommitted changes</code>
Last run	2026-10-15 10:15

Table 2: The run that produced the results in this paper.