Machine learning is (not) child's play*

Jonas Latz

Department of Mathematics, University of Manchester
jonas.latz@manchester.ac.uk

Abstract

Artificial intelligences are everywhere, they are immensely popular and often quickly adopted by the society. At the same time, AI literacy remains low. Learning about AI seems easy, *machine learning* is self-explanatory: the machine *learns* intelligent behaviour from data. Unfortunately, this analogy does not go deep enough to describe challenges, complexity, or resource intensity of the computational learning process – all of which are central for a careful judgement of AI outputs and usage at large. The basic learning analogy is rather oversimplifying, misleading, or even dangerous if the anthropomorphic terminology gives the AI too much agency. We initialise a debate around better ways to explain AI processes by proposing a more accurate analogy for machine learning – especially the algorithmic optimisation at its foundation. The analogy is given through a children's game that can accurately describe the machine learning process, show challenges arising within the process, and explain how these are approached or resolved within the state of the art. The analogy is constructed by introducing machine learning challenges and algorithms before carefully mapping those to aspects of the children's party game. This model still anthropomorphises but in a way that is more faithful to the computational process.

key words: artificial intelligence, education, mathematics and society
MSC2020: 97N60, 97P80, 97A40

1 Introduction

The term *Machine learning* describes the automated process of constructing an artificial intelligence (AI) from data. Machine learning is an anthropomorphic metaphor: intelligent beings often become intelligent by learning from observational data, e.g., books, lectures, or videos. Machine learning describes the related process in AIs. Similarly, *artificial neural networks* have played a fundamental role in AI and are intuitively understood as related to the neural networks in a brain. Analogies of this kind give an intuition behind a complicated process; these analogies in particular may very well have supported the advent and popularisation of AI. Simplified explanations through analogies are also a popular tool in the physical sciences, say, the hydraulic analogy of electricity [13].

AI has a growing impact on the society. A recent survey in the UK found that 72% of participants use AI one way or another [14]. Microsoft's Global AI Diffusion measurements

*Preprint. Version: May 18th 2026. Available from https://latzplacian.org/wp-content/uploads/2026/05/Latz2026_MLisNotChildsPlay_v1_2.pdf

paint a similar picture [19]. However, *AI literacy* still remains low. AI literacy is defined as "a set of competencies that enables individuals to critically evaluate AI technologies; communicate and collaborate effectively with AI; and use AI as a tool online, at home, and in the workplace"[18]. In the aforementioned survey [14], 56% of participants felt that it is important to be able to judge accuracy and reliability of AI outputs, but only 26% of those feel confident in doing so. Thus, core competencies of AI literacy are recognised as important by the society, but AI literacy is low.

Whilst the anthropomorphic metaphor *machine learning* helped popularising AI, it does not go deep enough to help with AI literacy. The *learning* in machines is conceptually different to the usual learning of intelligent beings: the analogy is inaccurate [17, 29, 33]. Educators are critical about teaching wrong analogies as they may lead to misconceptions in learners and even "limit further knowledge acquisition" [40]. Indeed, knowing that an AI has *learned* a certain task must not be the end of knowledge acquisition: Firstly, the term *learning* oversimplifies the underlying computational problem that is resource-intensive and so complex that existing algorithms are often not fully reliable. Effective collaboration with and critical evaluation of AI technologies are only possible when knowing about resource cost and reliability. Secondly, the anthropomorphic terminology may be dangerous if it leads to too much agency for the resulting AI [38]. 'The machine has learnt about causes for bridges to collapse' is not the same as 'we have hired a civil engineer' when it comes to infrastructural decision making. This may sound obvious, but distinguishing learning in machines and in intelligent beings may be the core behind the ability to critically evaluate AI technologies.

The goal of this article is to present a more accurate analogy for the algorithmic process behind the learning in a modern AI. Indeed, we make an analogy between optimisation algorithms that are the foundation for machine learning and the simple children's game *pot hitting*. Pot hitting will allow us to describe basic optimisation methods, central machine-learning-specific challenges in optimisation, and how they are approached or resolved in modern AIs. An appreciation of the challenges associated with machine learning is necessary to anticipate shortcomings of AIs, interpret errors that AIs make, to understand the resource-intensity of machine learning, and to distinguish machine learning from learning in intelligent beings.

We first give a brief theoretical background to supervised machine learning and the associated optimisation methods. We then introduce the pot hitting game and connect it to the aforementioned optimisation methods, especially focussing on machine learning challenges and solutions. We conclude by discussing advantages and limitations and give some pointers towards the use of the pot hitting analogy in practice.

2 Supervised learning and optimisation

We focus on optimisation problems in *supervised (machine) learning* [3]. In supervised learning, we are given *input-label data pairs* $(x_1, \ell_1), \dots, (x_N, \ell_N) \in X \times L$. Our goal is to use the data pairs to construct a *model* $h : X \to L$ that is able to predict a label $\ell \in L$ from its associated input $x \in X$, i.e. $h(x) \approx \ell$. The model is what we might call an *artificial intelligence*. Typical supervised learning tasks may be to connect medical symptoms (input) with a matching diagnosis (label) [25] or to detect and recognise pedestrians, road signs, and else (labels) on a video stream (input) to ultimately control a self-driving car [30].

We assess a model h with respect to its ability to *generalise* from the data pairs to unseen combinations of x and ℓ. To define *generalisation*, we assume that any input-label pair (x, ℓ) is

a sample from a probability distribution π. The model h shall then minimise the *generalisation error*

$$G(h) := \mathbb{E}[\text{loss}(h(x), \ell)] := \int_{X \times L} \text{loss}(h(x), \ell)\pi(\mathrm{d}x, \mathrm{d}\ell).$$

Here, $\text{loss} : L \times L \to [0, \infty)$ denotes a *loss function*, which measures the distance between two labels. $G(h)$ is the expected loss between predicted label $h(x)$ and actual label ℓ, assuming that $(x, \ell) \sim \pi$. In practice, we cannot access $G(h)$ since we have only access to the N data pairs and, thus, need to work with the *empirical error* given by

$$E(h) := \frac{1}{N} \sum_{n=1}^{N} \text{loss}(h(x_n), \ell_n).$$

We usually consider models h that are given in a *parametric form* $h =: H(\cdot; \theta)$. Here θ is the *parameter* and identifies the model h; the parameter lives in a *parameter space* $\Theta := \mathbb{R}^K$. We can, for instance, describe the simple task of *linear regression* in this framework by setting $X := L := \mathbb{R}$, $\text{loss}(\ell, \ell') := (\ell - \ell')^2$, $H(x; \theta) := \theta_1 + \theta_2 x$, and $\Theta := \mathbb{R}^2$. In the context of *deep neural networks*, H describes the chosen neural network architecture and θ is the collection of weight matrices and bias vectors [1].

Optimisation The computational challenge underlying machine learning now consists in finding the model parameter θ_* that *minimises* the empirical error, i.e. $E(H(\cdot; \theta_*)) \leq E(H(\cdot; \theta))$ for any $\theta \in \Theta$. We usually write this problem as

$$\min_{\theta \in \Theta} \bar{\Phi}(\theta) \tag{OP}$$

where

$$\bar{\Phi} := \frac{1}{N} \sum_{n=1}^{N} \Phi_n, \qquad \Phi_n(\theta) := \text{loss}(H(x_n; \theta), \ell_n).$$

The process of solving the problem (OP) is often referred to as the *training* of the model H. In the following, we will assume that $\bar{\Phi}$ is continuously differentiable and we discuss four iterative optimisation algorithm for problems of the form (OP). *Iterative* means here that the algorithm constructs a sequence $(\theta_t)_{t=0}^{\infty}$ that shall converge to the minimiser θ_* of $\bar{\Phi}$.

Gradient descent The most basic algorithm for (OP) is *gradient descent* (GD). The underlying idea is that the negative gradient $-\nabla\bar{\Phi}(\theta)$ points into the direction into which the value of $\bar{\Phi}$ descents most as seen from θ. So we should intuitively find a new θ with a smaller function value after every step when iteratively walking in negative gradient direction. Indeed, gradient descent starts at some initial value θ_0 and then proceeds by computing

$$\theta_{t+1} \leftarrow \theta_t - \eta_t \nabla\bar{\Phi}(\theta_t) \qquad (t = 0, 1, \ldots). \tag{GD}$$

Here, $\eta_t > 0$ denotes the *learning rate* at time t. GD converges to θ_* if $\bar{\Phi}$ is convex and if the learning rates are chosen sufficiently small. The function $\bar{\Phi}$ is *convex*, if for any $\theta, \theta' \in \Theta$, the graph of $\bar{\Phi}$ never lies above the line connecting $(\theta, \bar{\Phi}(\theta))$ and $(\theta', \bar{\Phi}(\theta'))$. We illustrate this in Fig. 1. Convexity is a very powerful property to have in optimisation: Note that gradient descent terminates once it finds a $\theta^\dagger$ for which $\nabla\bar{\Phi}(\theta^\dagger) = 0$. Such a $\theta^\dagger$ is called *stationary*.

Figure 1: A convex (left) and non-convex (right) function. Both functions have their minimisers at the red mark. Only for the convex function (left) this minimiser can be determined by finding the stationary point. The non-convex function has several other stationary points. The teal dashed line illustrates the definition of convexity.

For a convex function, we have that $\theta^\dagger = \theta_*$, i.e., any stationary point minimises $\bar{\Phi}$. For non-convex functions, this is not true, as Fig. 1 also illustrates. We refer to [11] for details on gradient descent and its convergence.

We now discuss specific issues when using gradient descent in supervised learning and how they are typically approached or resolved.

Big data The efficient computation of gradients through backpropagation (on GPUs) enables the use of gradient-based optimisation in the training of large-scale deep neural networks. However, gradient descent may become too computationally intensive in case the number of data pairs N is large. When computing the gradient of $\bar{\Phi}$, we actually compute

$$\nabla\bar{\Phi} = \frac{1}{N}\sum_{n=1}^{N}\nabla\Phi_n,$$

which means that we actually need to compute N gradients per iteration of gradient descent. This can be resolved with *stochastic gradient descent* (SGD), a method first proposed by Robbins and Monro [26]. The idea is here to not step into the direction of the negative gradient of $\bar{\Phi}$, but to replace it by a randomly chosen Φ_n. Thus, we train the model with respect to one randomly chosen data pair at a time instead of all of the data pairs simultaneously. We obtain the following algorithm

$$\theta_{t+1} \leftarrow \theta_t - \eta_t\nabla\Phi_{n(t)}(\theta_t), \qquad (t = 0, 1, \ldots), \tag{SGD}$$

where $n(0), n(1), \ldots$ are independent and identically distributed (i.i.d.) uniform random variables drawn from $[N] := \{1, \ldots, N\}$. Instead of just one data pair, we can also choose a batch of M of the Φ_n's and obtain *batch gradient descent* (BGD)

$$\theta_{t+1} \leftarrow \theta_t - \frac{\eta_t}{M}\sum_{n\in A_t}\nabla\Phi_n(\theta_t), \qquad (t = 0, 1, \ldots), \tag{BGD}$$

where $A_0, A_1, \ldots \subseteq [N]$ are independent sets of M samples drawn uniformly without replacement from $[N]$. SGD and BGD can be shown to converge to a minimiser of $\bar{\Phi}$ if the sequence of learning rates η_t are reduced to 0 over time and if the $\Phi_1, \ldots, \Phi_N$ are *strongly convex*. We refer to [11] for the definition of strong convexity and convergence results. If η_t is chosen to be constant, the algorithms SGD and BGD can still reach a stationary regime in a probabilistic sense, see [9, 16]. Since both, SGD and BGD work with data subsets, we refer to them as *subsampling methods*.

Non-convex In the context of deep neural networks, the losses $\Phi_1, \dots, \Phi_N$ are usually non-convex [7]. As discussed previously, GD does not necessarily find the minimiser in this case. Indeed, it will find a stationary point of $\bar{\Phi}$ and then terminate, but this stationary point does not need to be the minimiser of $\bar{\Phi}$. SGD and BGD do not necessarily terminate at a stationary point of $\bar{\Phi}$, but will be very slow to escape [8]. Global optimisation methods that can deal naturally with non-convexity, such as simulated annealing, are likely computationally too expensive in machine learning. *Momentum methods* are popular alternatives to the methods mentioned above: Where GD can be understood as a system describing the motion of a massless particle, momentum methods model this particle as massive. Thus, if the particle retains kinetic energy, it can escape stationary points of $\bar{\Phi}$. The most basic of these momentum methods is *Polyak's Heavy Ball method* (HBM) [22, 23] given by

$$\theta_{t+1} \leftarrow \theta_t - \alpha_t \nabla \bar{\Phi}(\theta_t) + \beta_t(\theta_t - \theta_{t-1}) \qquad (t = 0, 1, \dots), \tag{HBM}$$

where $\alpha_t, \beta_t > 0$ appear to be similar to learning rates, but also control particle mass and friction in the system and, e.g., $\theta_{-1} = \theta_0$. Momentum methods are usually not guaranteed to minimise non-convex functions, but they can search the parameter space more efficiently. Combinations of momentum and subsampling are possible and popular, see, e.g. [28, 37]. The *Adam method* [15] is a particularly popular combination of momentum, stochastic gradient descent, and adaptive learning rates.

High-dimensional The parameter space $\Theta = \mathbb{R}^K$ in modern deep neural networks is very high dimensional, i.e., K is very large. Artificial neural networks used in image classification may have $K \approx 10^8$ [32]. The Large Language Model ChatGPT has a parameter spaces with at least $K \approx 10^{11}$ [5]. Higher dimensional parameter spaces have an increased memory requirement and an increased cost when computing gradients. If the computational burden of computing and storing full gradients is too large, *coordinate descent* methods can be employed. *Coordinate descent* (CD) [39] proceeds in the following way

$$\theta_{t+1} \leftarrow \theta_t, \qquad \theta_{t+1}^{(k(t))} \leftarrow \theta_t^{(k(t))} - \eta_t \nabla \bar{\Phi}^{(k(t))}(\theta_t) \qquad (t = 0, 1, \dots), \tag{CD}$$

where $(k(t))_{t=0}^{\infty}$ is a sequence of randomly or deterministically chosen parameter indices in $[K]$, $\theta^{(k)}$ denotes the *kth component* of $\theta \in \Theta$, and, accordingly, $\nabla \bar{\Phi}^{(k)}$ is the *partial derivative* of $\bar{\Phi}$ in the k-th direction. Hence, we update component after component, following the parts of the gradient responsible for those components. Blocked subsets of components, similar to batch gradient descent, can also be used.

Overfitting Overfitting is a common issue in supervised learning. It describes the situation in which we find a parameter θ_* that minimises the empirical error $E(H(\cdot; \theta_*))$, but for which the generalisation error $G(H(\cdot; \theta_*))$ is not small. This means that the hypothesis represents the observational data $(x_1, \ell_1), \dots, (x_N, \ell_N)$ well, but does not *generalise* well to unseen data $(x, \ell) \sim \pi$. This usually happens in situations in which the model H is highly flexible. Then, the chosen model is prone to fitting aspects of the labels that are not actually explainable by the inputs, such as measurement noise or mislabelled data. We give an example for overfitting in polynomial regression in Fig. 2. Overfitting can be prevented by choosing less flexible models H or, e.g., regularisation [20]. In the context of deep neural networks, we often rely on *implicit regularisation*. We reach implicit regularisation, for instance, by employing a

stochastic optimisation method, but keeping the learning rate constant [4, 35]. As mentioned previously, SGD or BGD may not converge to the minimiser of $\bar{\Phi}$, but find a stationary regime in a probabilistic sense. This probabilistic state appears to be better at preventing overfitting. We illustrate these different ways to resolve overfitting also in Fig. 2.

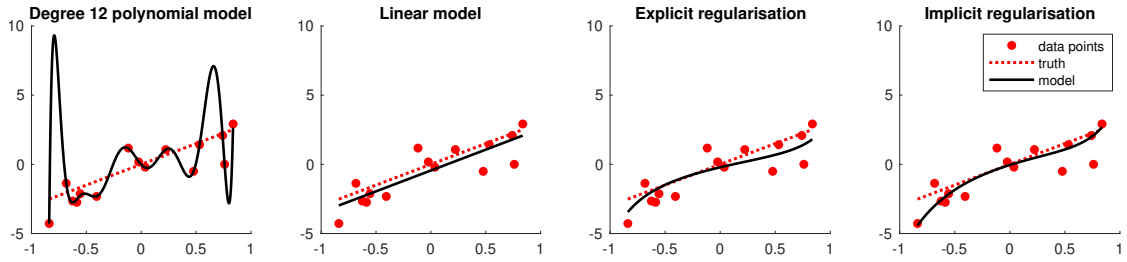

Figure 2: The solution to a linear regression problem using (from left to right) a hypothesis space containing degree 12 polynomials, a hypothesis space containing linear functions, degree 12 polynomials Tikhonov regularised, and degree 12 polynomials implicitly regularised using SGD. The polynomials on their own overfit the data and do not approximate the true linear function well; using a simpler model or regularisation helps to prevent overfitting.

3 Pot hitting – a children's game

Pot hitting is a simple children's game that requires two participants, which we refer to as the *player* and the *director*. The game is usually played inside in a room but at least in an enclosed area: the *pitch*. A *pot* is placed upside down onto the pitch, covering a treasure (e.g. sweets). Player and director act as a team, trying to recover the treasure that they share in the end of the game. During this recovery process, the player is blindfolded and given a wooden *stick* or maybe a wooden spoon. The stick shall help the player to identify the pot. Indeed, the game ends once the player hears the *clunk* sound of the wooden stick hitting the pot. The director is not blindfolded, but may only support the player with verbal cues. What these verbal cues can be is subject to the rule set that the participants agree upon. The director may only give feedback to steps (say, 'hot' or 'cold' depending on whether the player gets closer to or farther away from the pot) or even only enter the game to prevent the blindfolded player from injury. For the purpose of the following discussion, we assume that precise directions are communicated as well a sense of distance, such as 'a small step straight ahead', 'a bit to the left', or 'a large step towards 2 o'clock'. We summarise the game in a flow chart in Fig. 3.

Pot hitting is particularly popular in German-speaking countries, where it is known as *Topfschlagen* [36]. Similar games exist in various other countries: *pin the tail on the donkey* [6] is a popular game in the English-speaking world; Bhattacharjee and Boro [2] report a related ethnic game played by the Bodos.

4 The pot hitting perspective in optimisation

Pot hitting is closely related to GD with convex $\bar{\Phi}$. The imaginary pot that shall be identified when solving the problem (OP) is a minimiser θ_* of $\bar{\Phi}$. GD acts as the player, the data informs the director: θ_t describes the position of the player on the pitch Θ at time t, the

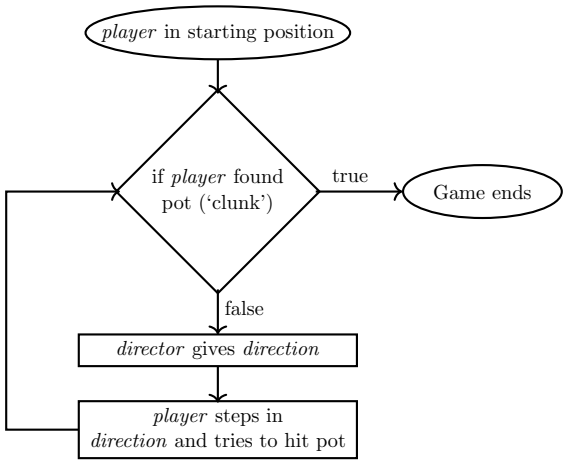

Figure 3: Flow chart for the game of pot hitting

negative gradient $-\nabla\bar{\Phi}(\theta_t)$ is the verbal cue that the director gives to the player at time t. As in pot hitting, the iterate θ_t itself cannot determine where it lies in space relative to θ_*: it is blind and can only access an oracle, the negative gradient $-\nabla\bar{\Phi}$, that points into the direction of smaller function values. Having understood this connection, we now take the challenges from Section 2 and translate them into the game of pot hitting. We then give strategies to overcome these challenges that are motivated by the strategies used in machine learning. Thus, we do not only understand GD within pot hitting, but also the more advanced methods SGD/BGD, HBM, and CD.

Learning rates Even in convex problems, GD converges only if the learning rate η_t is chosen to be sufficiently small. If η_t is chosen too large, we usually see oscillations around the minimiser in the algorithm in which case the algorithm can diverge. The gradient may suggest a long step as the target is relatively far from the minimiser and the large learning rate catapults the iterate far beyond the minimiser. Small learning rates will lead to a larger number of steps being necessary to get close to the minimiser. We illustrate this in Fig. 4.

In pot hitting, this problem occurs when player and director have a misaligned understanding of step sizes. A 'small step' might be anywhere below 1 foot ($\sim$ 30cm), a 'large step' could be somewhere between 1 and 3 feet ($\sim 30-90$cm). If step lengths are interpreted as too long, the player may overstep the pot time and time again and never hit it. If step sizes are interpreted too short on the other hand, the player will find the pot, but it may take a very long time. In this case, the director could readjust the step sizes to account for the players misinterpretation. This is not contained in the basic version of GD. We would assume that the director determines the verbal cue only based on the current position of the player and independently of the past.

Learning rates that depend on the past are popular in practice, see, e.g., [15]. They would allow the director to adjust their statements and, thus, guide the player to the pot even if there is a wrong step length adjustment in the beginning. So the director could, for instance, suggest a 'very small step' at a point at which the player has overestimated a 'small step' previously.

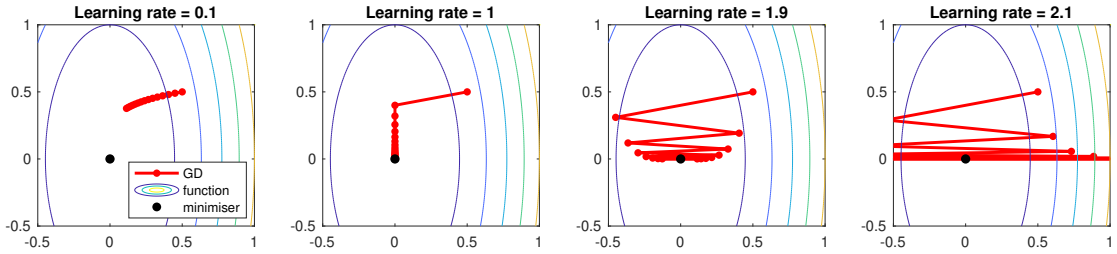

Figure 4: 15 GD steps minimising $\bar{\Phi}(\theta) = 0.5\theta_1^2 + 0.1\theta_2^2$ with different learning rates η. A learning rate that is too small will lead to slow convergence; a learning rates that is too large will prevent convergence.

Big data If the function $\bar{\Phi}$ is given as the mean of many functions $\Phi_1, \dots, \Phi_N$ as in (OP), we should interpret this within pot hitting as an alternative set of rules: there is a total number of N directors that give verbal cues to the player. Each of the directors represents one data set (x_n, ℓ_n) or, equivalently, one of the Φ_n. Thus, each of the directors only sees part of the pitch and the correct direction is the mean of the cues given by the N directors. Asking the N directors for the direction is rather simple, listening to all N directions and computing their mean will be very difficult for the player – especially if N is in the thousands or millions. SGD and BGD solve this issue by listening to only one or few directors at a time, respectively. The directors that the player listens to are picked randomly prior to every step they do.

We have to amend the game in the big data setting even more. Since the player has no access to $\nabla\bar{\Phi}$, they never hear the *clunk*, that would usually be signified by seeing that $\nabla\bar{\Phi} = 0$. However, in SGD/BGD the player would in any case need to employ decreasing step sizes to converge. So rather than searching until they hear a *clunk*, the player would search the space until they have (essentially) stopped moving and should then be right above the pot. Thus, in the big data setting, the player does not actually hit the pot with a stick but walks, say, until the steps become very small. We illustrate this behaviour in Fig. 5: when the learning rate is constant, the player keeps on wandering around the pitch. With a decreasing learning rate, the player eventually stays very close to the minimiser. At this point, they may just remove their blindfold and pick up the pot.

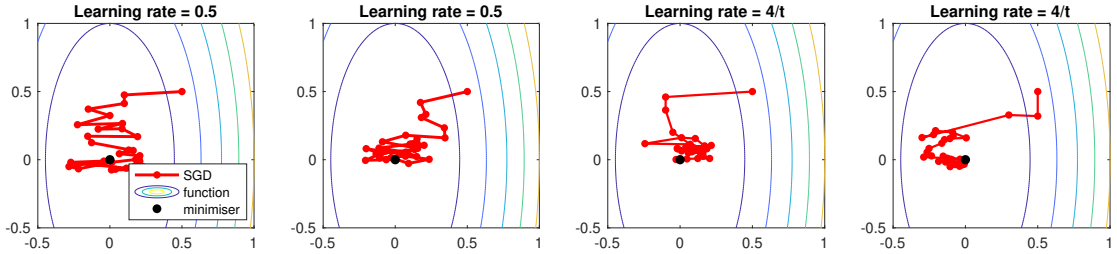

Figure 5: 30 SGD steps minimising $\bar{\Phi}(\theta) = \sum_{n_1,n_2=-10}^{10} 0.5(\theta_1 - n_1/20)^2 + 0.1(\theta_2 - n_2/20)^2$ with different constant and decreasing learning rates η_t. When reducing the learning rate, we appear to converge to the minimiser, a constant learning rate leads to a circling around the minimiser.

Non-convex In convex problems, there is only one pot placed onto the pitch: if a convex function has multiple minimisers, those minimisers are right next to each other. So those could be represented by a larger pot. Thus, as soon as the player hears the *clunk* sound, they know that they have reached the pot and the game ends. In non-convex problems, the situation is different: the room is full of pots in different places with a variety of bigger and smaller treasures. Each of the pots representing one stationary point. So when the player hears the *clunk*, it is not clear whether they have reached the pot with the treasure of highest value – the highest value represents the minimiser θ_*. Since they do not know what treasure to expect they will just stop after hearing the first *clunk* and have to be happy with that treasure.

In the pot hitting setting, the use of a momentum method would mean that the pitch has a low friction: say the game is played on a soapy or frozen surface or the player is wearing roller skates. The player can still move on the pitch (without falling), but cannot slow down immediately when hearing a *clunk* from a pot and, thus, may overshoot the pot after hearing the *clunk*. If still close enough to that pot, the director would then point the player back toward that pot. The player will then overshoot again and finally only get to stop after overshooting several more times. However, it is also possible that the velocity of the player is so large when coming across that pot, that they overshoot considerably and that a different pot becomes more convenient to reach. In this case, the director would then guide the player to that more convenient pot. In any case, the player is not stuck with the first pot they hit, but may be able to move toward a pot containing a larger treasure. Of course, there is no guarantee that the method eventually recovers the largest treasure, it will just be able to escape pots in general. We illustrate the momentum-based method in a convex setting in Fig. 6 and in a non-convex setting in Fig. 7. In the convex setting, we can see how the player overshoots, but eventually finds the pot. In the non-convex problem, we can see how a stationary point can be overcome and the minimiser be found.

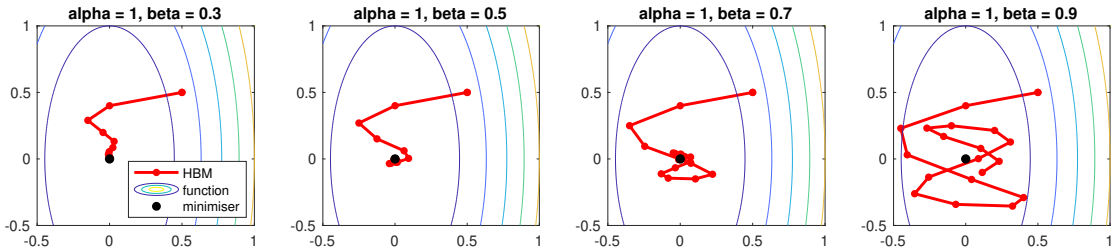

Figure 6: 15 HBM steps minimising $\bar{\Phi}(\theta) = 0.5\theta_1^2 + 0.1\theta_2^2$ with $\alpha_t = 1$, and various β_t. A larger β_t leads to a higher mass or a lower frictions implying that the momentum from past movement is kept longer.

High-dimensional Pot hitting is usually played in two dimensions: the pot is placed on the ground: only a two-dimensional surface needs to be searched. Intuitively, it feels easier to find a pot in one dimension, i.e., the director only needs to tell the player to go left or right, and more difficult to find a pot in three dimensions, where not only a flat angle needs to be specified but also a direction in which to move vertically to hit the pot in the air. When searching a space of millions or billions of dimensions, even communicating all the components of the direction would be challenging. CD refers to the idea of asking for each axis-direction

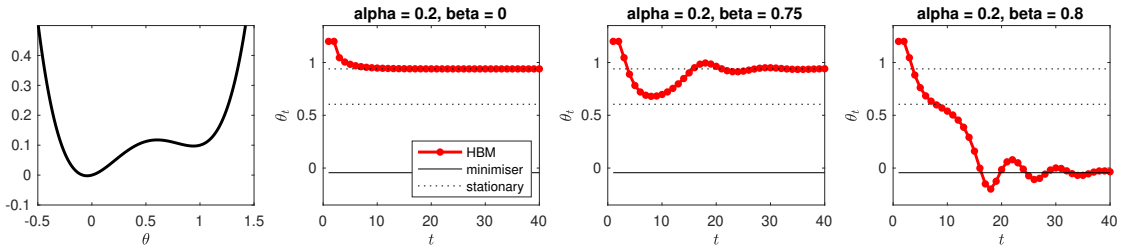

Figure 7: A plot of the non-convex function $\bar{\Phi}(\theta) = \theta^2(\theta-1)^2 + 0.1\theta$ (left) and 40 HBM steps minimising $\bar{\Phi}$ with $\alpha_t = 0.2$, and various β_t (right three plots). $\beta_t = 0$ is equivalent to GD and neither this method nor (HBM) with $\beta_t = 0.75$ can escape the stationary point at ≈ 0.9. With the larger $\beta_t = 0.8$, the method escapes and finds the global minimiser at ≈ 0.

separately: 'how far left or right do I need to go?' or 'do I need to go forwards or backwards' and then only walking in axis-directions, i.e., like a rook on a chess board. We illustrate this behaviour in Fig. 8. Each step is then much easier to make, but the total number of steps needed may be considerably larger.

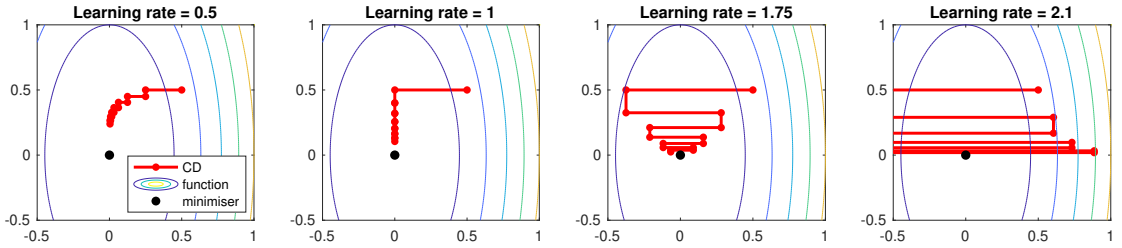

Figure 8: 15 CD steps minimising $\bar{\Phi}(\theta) = 0.5\theta_1^2 + 0.1\theta_2^2$ with various learning rates. The plots illustrate the rook-like movements only in axis-directions. Again, learning rates that are too short may lead to a slow convergence behaviour, while learning rates that are too long may prevent convergence all along.

Overfitting Implicit regularisation appears in stochastic optimisation, for instance, when the learning rate η_t is not reduced to 0 over time. In that case, we are likely not to converge to the minimiser of $\bar{\Phi}$, but may find a more robust solution that may not overfit the data in the supervised learning problem. This solution may be in the area surrounding the pot (at least in a convex setting). In pot hitting this means that we purposefully walk towards the pot, but implicitly try to avoid actually hitting it by consistently stepping over it instead of reducing the stepsize and stopping right in front of it. This behaviour is illustrated in the left two parts of Fig. 5.

The reasoning could be the following: Maybe the player heard someone walking across the pitch, tripping over the pot, and distributing the treasure around it. So now the player knows that there may be little of the treasure left under the pot, so they need to search the area around the pot and pick up the sweets one by one. One way to find the area surrounding the pot could be to let the many directors guide the player towards the pot in an SGD/BGD-sense, but not to reduce the stepsize. Then, they will cover the area around the pot well without actually stopping. Similarly, when avoiding overfitting, we walk toward the minimiser of $\bar{\Phi}$,

but actually need to find a regularised solution close-by.

5 Discussion and conclusions

We have described the pot hitting game as an analogy for the optimisation process underlying machine learning – presenting it as a more elaborate alternative to the metaphorical analogy contained in the term 'machine learning' itself. We have explained a number of challenges underlying machine learning optimisation and illustrated them in *hard mode* versions of pot hitting: big data, non-convexity, high-dimensionality, as well as a version that may prevent overfitting. Each of them portraying the difficulty in machine learning optimisation, but also showing how these challenges are approached in practice.

The analogy is surprisingly deep and accurate. It implicitly conveys the difficulty and resource intensity of the machine learning process, especially when realising that the training of modern AIs encounters a combination of big data, non-convexity, and high-dimensionality. Importantly, it does so by putting the player into the sometimes easy, sometimes almost helpless position of the iterate in the optimisation method. In this way, the algorithm is anthropomorphised, but, firstly, as a struggling player in a close-to-impossible game, rather than with idealising terminology, such as 'learning' or 'intelligence'. Secondly, the player can as well be seen as the machine learning engineer that is employing an optimisation method to train an AI. Hence, pot hitting may be a faithful model, rather than an anthropomorphism.

Pot hitting is simple enough to be understood by anyone within a few minutes. The pot hitting perspective may work well as a thought experiment to explain machine learning to curious people with a missing background in the computer or mathematical sciences or to give an intuition whilst teaching machine learning to a STEM audience. However, it is also a game that can for a large part be played and experienced in practice. Playful learning of this type is effective in both children and adults [24] – especially to improve motivation, engagement, and creativity. Caveats are the high-dimensional and the momentum setting: Even three dimensions might be difficult to realise in practice: the increment in complexity when going from $K = 1$ to $K = 2$ can be motivated and observed though; having a blindfolded player walking on a low friction surface implies a high risk of injury – so the momentum setting is not advisable. Hence, even if played, a part of the pot hitting experiments discussed here need to end in thought experiments [10, 31].

The pot hitting analogy can be extended to other modalities of machine learning, such as unsupervised learning [3] or generative machine learning [12]. These modalities also rely on non-convex, high-dimensional optimisation with respect to big data and are approached with similar algorithms. It is more difficult to represent other central challenges in AI within the analogy that go beyond the computational learning process, such as alignment [21], interpretability [27], and ethical questions [34].

Acknowledgements

The author is grateful for comments made by Tatiana Alessandra Bubba, Lukas Latz, and Timothy Waite who read a draft of this manuscript.

References

[1] Y. Bengio, I. Goodfellow, A. Courville, et al. *Deep learning*, volume 1. MIT press Cambridge, MA, USA, 2017.

[2] S. Bhattacharjee and F. Boro. A Study on the Little Known Ethnic Games of the Bodos and Karbis of Assam. *Journal of the Anthropological Survey of India*, 67(2):185–201, 2018.

[3] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[4] G. Blanc, N. Gupta, G. Valiant, and P. Valiant. Implicit Regularization for Deep Neural Networks driven by an Ornstein-Uhlenbeck like Process. In *Conference on Learning Theory*, pages 483–513. PMLR, 2020.

[5] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language Models are Few-Shot Learners. arXiv 2005.14165, 2020.

[6] R. Carlisle. *Pin the Tail on the Donkey*. Sage Publication, 2009.

[7] A. Choromanska, M. Henaff, M. Mathieu, G. Ben Arous, and Y. LeCun. The Loss Surfaces of Multilayer Networks. In G. Lebanon and S. V. N. Vishwanathan, editors, *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pages 192–204, San Diego, California, USA, 09–12 May 2015. PMLR.

[8] H. Daneshmand, J. Kohler, A. Lucchi, and T. Hofmann. Escaping Saddles with Stochastic Gradients. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1155–1164. PMLR, 10–15 Jul 2018.

[9] A. Dieuleveut, A. Durmus, and F. Bach. Bridging the gap between constant step size stochastic gradient descent and Markov chains. *The Annals of Statistics*, 48(3):1348 – 1382, 2020.

[10] I. Galili. Thought experiments: Determining their meaning. *Science & Education*, 18(1):1–23, 2009.

[11] G. Garrigos and R. M. Gower. Handbook of Convergence Theorems for (Stochastic) Gradient Methods. arXiv 2301.11235, 2024.

[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Commun. ACM*, 63(11):139–144, Oct. 2020.

[13] T. B. Greenslade Jr. The Hydraulic Analogy for Electric Current. *The Physics Teacher*, 41(8):464–466, 11 2003.

[14] His Majesty's Government, Department for Science, Innovation & Technology. AI Skills for Life and Work: General Public Survey Findings. `https://www.gov.uk/government/publications/ai-skills-for-life-and-work-general-public-survey-findings/ai-skills-for-life-and-work-general-public-survey-findings`, 2026.

[15] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[16] J. Latz. Analysis of stochastic gradient descent in continuous time. *Statistics and Computing*, 31:39, 2021.

[17] T. P. Lillicrap, A. Santoro, L. Marris, C. J. Akerman, and G. Hinton. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6):335–346, 2020.

[18] D. Long and B. Magerko. What is ai literacy? competencies and design considerations. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–16, New York, NY, USA, 2020. Association for Computing Machinery.

[19] A. Misra, J. Wang, S. McCullers, K. White, and J. L. Ferres. Measuring ai diffusion: A population-normalized metric for tracking global ai usage, 2025.

[20] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. MIT Press, second edition, 2018.

[21] R. Ngo, L. Chan, and S. Mindermann. The alignment problem from a deep learning perspective. In *The Twelfth International Conference on Learning Representations*, 2024.

[22] B. T. Polyak. Gradient methods for the minimisation of functionals. *USSR Computational Mathematics and Mathematical Physics*, 3(4):864–878, 1963.

[23] B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[24] L. Rice. Playful learning. *Journal for Education in the Built Environment*, 4(2):94–108, 2009.

[25] J. G. Richens, C. M. Lee, and S. Johri. Improving the accuracy of medical diagnosis with causal machine learning. *Nature Communications*, 11(1):3923, 2020.

[26] H. Robbins and S. Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400 – 407, 1951.

[27] C. Rudin, C. Chen, Z. Chen, H. Huang, L. Semenova, and C. Zhong. Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistics Surveys*, 16:1–85, 2022.

[28] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[29] A. Saxe, S. Nelli, and C. Summerfield. If deep learning is the answer, what is the question? *Nature Reviews Neuroscience*, 22(1):55–67, 2021.

[30] Q. Sellat, S. Bisoy, R. Priyadarshini, A. Vidyarthi, S. Kautish, and R. K. Barik. Intelligent semantic segmentation for self-driving vehicles using deep learning. *Computational Intelligence and Neuroscience*, 2022(1):6390260, 2022.

[31] F. Sharifi, F. Ahmadi, and M. Meshkat. Investigating the effect of teaching dynamic concepts with the help of thought experiments on the academic progress and cognitive skills of students. *Physics Education*, 60(4):045003, may 2025.

[32] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[33] F. H. Sinz, X. Pitkow, J. Reimer, M. Bethge, and A. S. Tolias. Engineering a less artificial intelligence. *Neuron*, 103(6):967–979, 2019.

[34] B. C. Smith. *The Promise of Artificial Intelligence: Reckoning and Judgment*. The MIT Press, 10 2019.

[35] S. L. Smith, B. Dherin, D. Barrett, and S. De. On the origin of implicit regularization in stochastic gradient descent. In *International Conference on Learning Representations*, 2021.

[36] O. Steinel. *Schülerbuch für den deutschen Aufsatz-Unterricht an den Mittelschulen im Sinne der Schulreform: Bearbeitet von Oskar Steinel und Karl Keppel Für Schüler von 9 bis zu 12 Jahren*. Selbstverlag von Karl Keppel, 1891.

[37] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning*, pages 1139–1147. PMLR, 2013.

[38] D. Watson. The rhetoric and reality of anthropomorphism in artificial intelligence. *Minds and Machines*, 29(3):417–440, 2019.

[39] S. J. Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

[40] K. B. Zook. Effects of analogical processes on learning and misrepresentation. *Educational Psychology Review*, 3(1):41–72, 1991.