Machine learning is (not) child's play[*]

Jonas Latz

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

Abstract

Machine learning describes the generation of an artificial intelligence (AI) through data. The terminology creates an analogy between this process and the learning in intelligent beings: An analogy that has been criticised as inaccurate in the past. Goal of this note is to create a better analogy explaining machine learning – especially the mathematical optimisation process at its foundation. The analogy is given through a children's game that is able to not only accurately describe the machine learning process, but to also show challenges arising within the process, and how these are solved within the state of the art in machine learning. This note acts as a brief introduction to optimisation in machine learning, it proposes and discusses the new analogy to the machine learning process, and it aims to inform teaching practice.

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

1 Introduction

Artificial intelligence is the area of computer and data science studying computers, software, and robots that show behaviour of intelligent beings. In the last decades, artificial intelligences were often constructed by *machine learning*, i.e. in an automated fashion from data. 'Machine learning' is a metaphor: intelligent beings often become (more) intelligent by learning from observational data, so this terminology is an obvious choice to describe the related process in machines. Similarly, 'artificial neural networks' have played a fundamental role in AI and are intuitively understood as related to the neural networks in a brain. The use of analogies of this kind is generally commendable, as it helps to given an intuition behind a complicated process and as it may have supported the advent and popularisation of AI [15, 22]. Analogies of this type that lead to simplified explanations also appear in the physical sciences, such as the hydraulic analogy of electricity [19].

Given the impact AI has on our society, the importance of finding ways to explain it to the society is unquestionable [48]. This necessity is underlined by a recent survey [22] that shows a stark imbalance between use and understanding of AIs in the society. However, the inbuilt analogy of the construction of artificial intelligences with learning of intelligent beings has rightfully been criticised as inaccurate and misleading [46]. Inaccurate analogies

[*]Preprint. Version: December 9th 2025. Available from https://latzplacian.org/wp-content/uploads/2025/12/Latz2025_MLisNotChildsPlay_v1_1.pdf

presented by, e.g., a teacher, may lead to misconceptions in learners and even "limit further knowledge acquisition" [49]. Better analogies for machine learning may be based on genetic or other evolutionary algorithms [12]. Those, however, neither reflect the complexity of machine learning nor represent what may be considered the state-of-the-art.

Goal of this note is to present a more accurate analogy for the process of machine learning as it is used to construct modern artificial intelligences. This analogy constructs a model based on a version of a children's game to describe the process of mathematical optimisation that is used to incorporate data into an artificial intelligence. The model is able to highlight challenges within this optimisation process and to represent solutions proposed to overcome these challenges. Whilst discussing this analogy, the note also acts as a self-contained short introduction to optimisation in machine learning using a combination of technical and intuitive explanations. Moreover, the note shall support the effort of finding simple and accurate analogies and models for processes within artificial intelligence and to stimulate discussions in this area.

We now give a brief introduction to the supervised learning problem and the associated optimisation problem, which will be the basis of our presentation. Then, we outline the remainder of this note.

Supervised learning. The goal in *supervised (machine) learning* [4, 23] is to find a function that can connect an *input* $x \in X$ to its associated but unknown *label* $\ell \in L$. In particular, we assume that there is an underlying probability distribution $\pi = \mathbb{P}((x, \ell) \in \cdot)$ and try to find a function $h : X \to Y$, referred to as *hypothesis* or *(machine learning) model*, that minimises

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

where loss : $L \times L \to [0, \infty)$ denotes a *loss function*: an appropriate way to measure the distance between two labels. $G(h)$ is often referred to as the *risk* or *generalisation error* of the hypothesis h and describes the loss we expect when sampling an input-label pair $(x, \ell) \sim \pi$ and comparing $h(x)$ to ℓ. $G(h)$ is small in case h is able to predict well which label ℓ is likely to belong to the input x and large otherwise.

In practice, we have no access to π and, thus, no access to G either. We can only access a number $N \in \mathbb{N}$ of independent samples $(x_1, \ell_1), \ldots, (x_N, \ell_N) \sim \pi$, the *data*. Hence, we can only find an h that minimises the *empirical risk* or *empirical error* given by

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

Depending on loss and the chosen set of possible hypotheses $\mathcal{H} \subseteq \{h : X \to L\}$, we can sometimes derive inequalities bounding

$$G \leq E + \text{additional terms}$$

with high probability if N is sufficiently large, see, e.g., [1, 10, 28]. If the additional terms are small, this would mean that a small empirical error implies a small generalisation error.

Supervised learning is a major task in machine learning and artificial intelligence. Of course, one may think of linear regression where $\mathrm{loss}(\ell, \ell') = (\ell - \ell')^2$ is a simple square distance and $\mathcal{H}$ is a simple linear space, e.g., $\mathcal{H} = \mathrm{span}(1, x, x^2)$. In modern machine learning, $\mathcal{H}$ often contains artificial neural networks or similarly complicated objects. Artificial neural networks are compositions of linear and non-linear functions: the nonlinear functions are typically predetermined and the linear functions are variable within $\mathcal{H}$; see [18, 21] for details. Typical supervised learning tasks may be to associate medical symptoms (input) with a matching diagnosis (label) [34] or to detect and recognise pedestrians, road signs, and else (labels) on a video stream (input) to ultimately control a self-driving car [38]. In these cases, the hypothesis h would indeed replicate the behaviour of an intelligent being.

Optimisation. A major challenge in supervised machine learning is the identification of a hypothesis h_* that minimises the empirical error E, i.e. h_* is chosen such that $E(h_*) \leq E(h)$ $(h \in \mathcal{H})$. We usually define a *parametric hypothesis* or *parametric model* $H(\cdot;\theta)$ with $\theta \in \Theta := \mathbb{R}^K$, i.e. $\mathcal{H} = \{H(\cdot;\theta) : \theta \in \Theta\}$. Then, we can write the problem of minimising E as a finite-dimensional, unconstrained optimisation problem

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

where

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

The process of solving the problem (OP) is often referred to as the *training* of a machine learning model and at the very basis of all supervised learning. Indeed, when speaking of a computer or software *learning* from data, we should implicitly not think of learning as a human being does it. We should think of a large scale optimisation problem that forces a parametric model to behave as the observed data – assuming that this behaviour generalises to unseen examples. Optimisation problems of this form are very challenging, as we will argue in the remainder of this note.

Outline. The remainder of this note commences with the introduction of some optimisation algorithms (Section 2) that are commonly used to solve (OP), alongside the challenges that these algorithms shall overcome. We then introduce the pot hitting game (Section 3) and explain its relationship to the algorithms used to solve (OP) (Section 4). Therein, we also explain the machine learning challenges mentioned earlier within the pot hitting game and how the game participants may employ them. The note ends with concluding remarks (Section 5).

2 Optimisation in supervised learning

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 converges to the minimiser of $\bar{\Phi}$ or to a stationary point thereof. A point $\theta' \in \Theta$ is *stationary* if the gradient $\nabla\bar{\Phi}(\theta') = 0$.

Gradient descent. The most basic and likely most common optimisation 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

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.

intuitively find a new θ with a smaller function value after every step. In gradient descent, we start at an initial value θ_0 and then iteratively walk in negative gradient direction

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

Here, $\eta_t > 0$ denotes the *stepsize* or *learning rate* at time t. GD can find the minimiser if $\bar{\Phi}$ is convex and if the learning rates are chosen sufficiently small. Otherwise, it will converge to a stationary point or diverge. The function $\bar{\Phi}$ is *convex*, if

$$\bar{\Phi}(\lambda\theta + (1-\lambda)\theta') \leq \lambda\bar{\Phi}(\theta) + (1-\lambda)\bar{\Phi}(\theta') \qquad (\theta, \theta' \in \Theta, \lambda \in [0,1]).$$

Convexity is important as it implies that a stationary θ' is already a minimiser of $\bar{\Phi}$. Thus, the local knowledge that $\nabla\bar{\Phi}(\theta_*) = 0$ provided by gradient descent is sufficient to know that a minimiser θ_* of $\bar{\Phi}$ has been found. We illustrate this in Figure 1, where we see that non-convex $\bar{\Phi}$ may have many stationary points that, for GD, are indistinguishable of the minimiser. We refer to [17, 30] for details on gradient descent and its convergence.

Subsampling techniques. A large number of data points N makes it difficult to apply the gradient descent method in practice. When computing the gradient of $\bar{\Phi}$, we actually compute

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

which is the mean of N gradients and computationally infeasible if N is large. *Stochastic gradient descent* (SGD), first proposed by Robbins and Monro [35], can help in this situation. 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. 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 Φ_n, 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 η_t converges to the minimiser and if the $\Phi_1, \ldots, \Phi_N$ are *strongly convex.* We refer to [17] for the definition of strong convexity and convergence results. If $\eta_t > 0$ is chosen to be constant, the stochastic processes SGD and BGD can converge to a stationary distribution, see [13, 25, 27]. Since both, SGD and BGD work with random subsets of the dataset, we refer to them as *subsampling methods.*

Momentum. In the context of artificial neural networks, the potentials $\Phi_1, \ldots, \Phi_N$ are usually non-convex [8]. As discussed previously, GD will in such a case find a stationary point that is not necessarily a minimiser of $\bar{\Phi}$. It actually gets stuck at stationary points and stops searching Θ for better model parameters. SGD and BGD do not necessarily get stuck, but will be very slow to escape [11]. Global optimisation methods that can deal naturally with non-convexity, such as simulated annealing [2], 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) [31, 32] given by

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

where $\alpha_t, \beta_t > 0$ appear to be similar to learning rates, but also control particle mass and friction in the system. Momentum methods are usually not globally convergent, but can search the space more efficiently. Combinations of momentum and subsampling are possible and popular, see, e.g. [24, 37, 44]. The *Adam method* [26] is a particularly popular combination of momentum, stochastic gradient descent, and adaptive learning rates.

High dimensions. The parameter space $\Theta = \mathbb{R}^K$ in modern artificial neural networks is very high dimensional, i.e., K is very large. Artificial neural networks used in image classification may have $K \approx 10^8$ [41]. The Large Language Model ChatGPT has a parameter spaces with at least $K \approx 10^{11}$ [6]. Higher dimensional parameter spaces have an increased memory requirement and an increased cost when computing gradients. In the context of convex optimisation, the speed of convergence in GD, SGD often does not depend on the dimension K – the conditioning of the problem is more central. If the computational burden of computing and storing full gradients is too large, *coordinate descent* methods can be employed. *Coordinate descent* (CD) [47] proceeds in the following way

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

where $(k(t))_{t=0}^{\infty}$ is a sequence of randomly or deterministically chosen parameter indices in $\{1, \ldots, K\}$, $\theta^{(k)}$ denotes the *kth component* of $\theta \in \Theta$, and $\partial_k \bar{\Phi}$ 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 an issue in supervised learning, especially in the context of artificial neural networks. In the setting of supervised learning introduced in Section 1, overfitting may be seen as a situation in which we find a parameter θ_* such that the empirical error $E(H(\cdot;\theta_*))$ is minimised, but the generalisation error $G(H(\cdot;\theta_*))$ is not small. This means that the hypothesis represents the observational data $(x_1, \ell_1), \ldots, (x_N, \ell_N)$ well, but does not *generalise* to unseen data $(x, \ell) \sim \pi$. This usually happens in situations in which the hypothesis space $\mathcal{H}$ is very rich. 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 Figure 2.

Overfitting can be prevented by choosing sparser hypothesis spaces $\mathcal{H}$ or, e.g., regularisation [20]. In the context of artificial 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 [5, 42]. With a constant learning rate, SGD or BGD do not converge to the minimiser of $\bar{\Phi}$, which might likely be overfitting. Instead, it may converge to a random stationary state. We illustrate the different ways to resolve overfitting also in Figure 2. We should note, however, that also gradient descent itself is good at finding regularised solutions, see, e.g., [9].

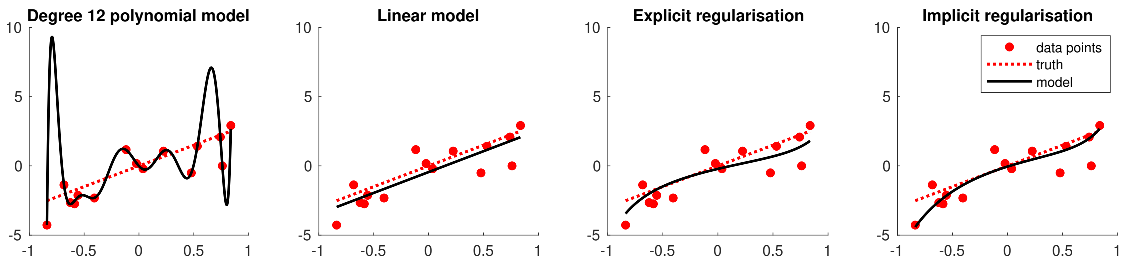

Figure 2: The solution to a linear regression problem using (from left to right) a hypothesis space containing degree 12 polynomials, a hypothesis space containing linear functions, degree 12 polynomials Tikhonov regularised, and degree 12 polynomials implicitly regularised using SGD. The polynomials 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 on 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 this note, 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'.

Pot hitting is particularly popular in German speaking countries, where it is known as

Topfschlagen [43]. Similar games exist in various other countries: 'pin the tail on the donkey' [7] is a popular game in the English-speaking world; Bhattacharjee and Boro [3] report a related ethnic game played by the Bodos.

4 The pot hitting perspective in optimisation

We now explain how the game pot hitting described in Section 3 relates 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, the 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.

Throughout this section we now present the challenges within machine learning optimisation given in Section 2 within the game of pot hitting. We then give strategies to overcome these challenges that are motivated by those strategies introduced above. Thus, we not only translate GD into the pot hitting game, but also the other methods introduced above.

Step sizes. 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 Figure 3.

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. In the basic version of GD, we probably would not allow for that. We would assume that the director determines the current cue based on the current position of the player and independently of, e.g., the past.

Learning rates that depend on the past are popular in practice, see, e.g., [14, 26]. 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 suggest a 'very small step' at a point at which the player has overestimated a 'small step' previously.

Big data. If the function $\bar{\Phi}$ is given as mean of many functions $\Phi_1, \ldots, \Phi_N$ 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 potentials 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

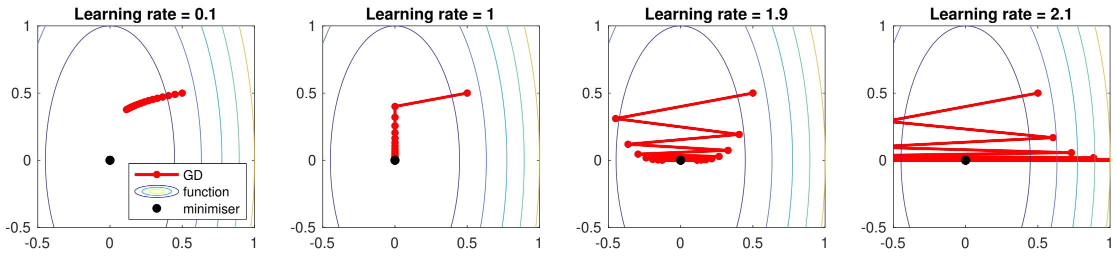

Figure 3: 15 GD steps minimising $\bar{\Phi}(\theta) = 0.5\theta_1^2 + 0.1\theta_2^2$ with different learning rates η. A learning rate that is too small will lead to slow convergence; a learning rates that is too large will prevent convergence.

or millions. SGD and BGD solve this issue by listening to only one or few directors at a time. 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 Figure 4: when the learning rate is constant, the player keeps on wandering around the pitch. With a decreasing learning rate, the player appears to eventually stay very close to the minimiser. At this point, they may just remove their blindfold and pick up the pot.

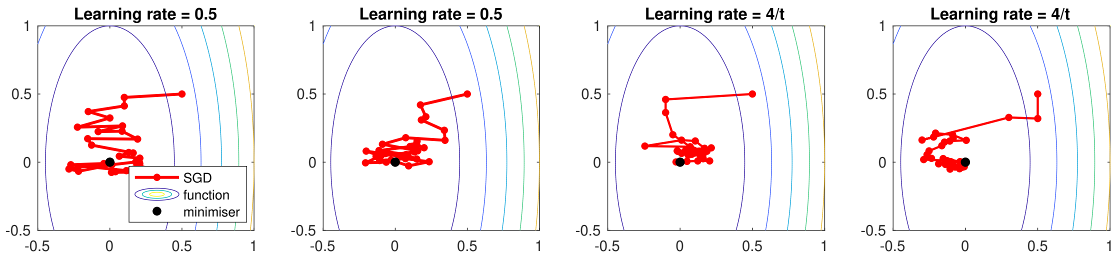

Figure 4: 30 SGD steps minimising $\bar{\Phi}(\theta) = \sum_{n_1,n_2=-10}^{10} 0.5(\theta_1 - n_1/20)^2 + 0.1(\theta_2 - n_2/20)^2$ with different constant and decreasing learning rates η_t. When reducing the learning rate, we appear to converge to the minimiser, a constant learning rate leads to a circling around the minimiser.

Non-convexity. In convex problems, there is only one pot placed onto the pitch – multiple minimisers would contain identical treasures and be 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. 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 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.

We illustrate the momentum-based method in a convex setting in Figure 5 and in a non-convex setting in Figure 6. 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.

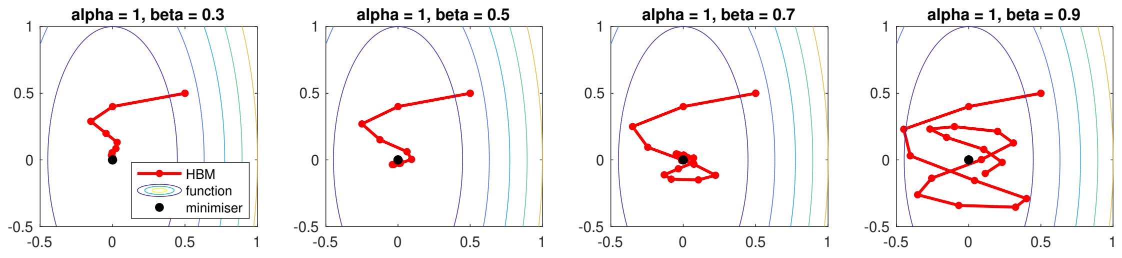

Figure 5: 15 HBM steps minimising $\bar{\Phi}(\theta) = 0.5\theta_1^2 + 0.1\theta_2^2$ with $\alpha_t = 1$, and various β_t. A larger β_t leads to a higher mass or a lower frictions implying that the momentum from past movement is kept longer.

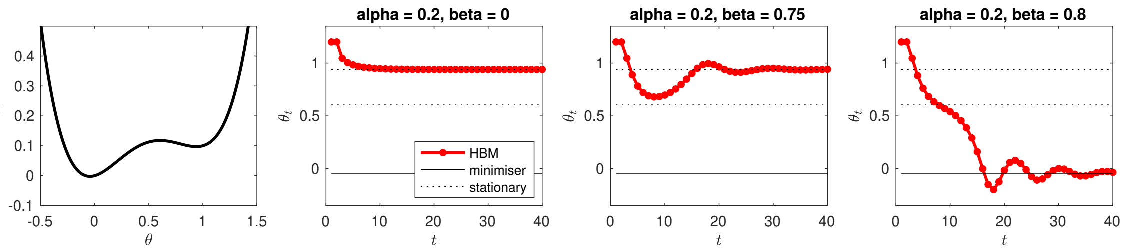

Figure 6: A plot of the non-convex function $\bar{\Phi}(\theta) = \theta^2(\theta-1)^2 + 0.1\theta$ (left) and 40 HBM steps minimising $\bar{\Phi}$ with $\alpha_t = 0.2$, and various β_t (right three plots). $\beta_t = 0$ is equivalent to GD and neither this method nor (HBM) with $\beta_t = 0.75$ can escape the stationary point at ≈ 0.9. With the larger $\beta_t = 0.8$, the method escapes and finds the global minimiser at ≈ 0.

High dimensionality. 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 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 Figure 7.

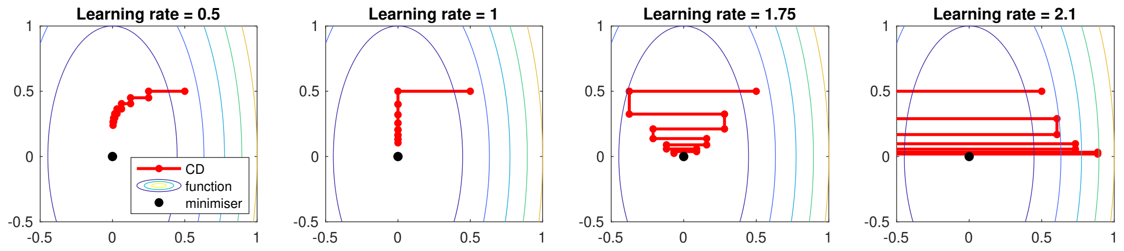

Figure 7: 15 CD steps minimising $\bar{\Phi}(\theta) = 0.5\theta_1^2 + 0.1\theta_2^2$ with various learning rates. The plots illustrate the rook-like movements only in axis-directions. Again, learning rates that are too short may lead to a slow convergence behaviour, while too long learning rates may prevent convergence all along.

Overfitting. Implicit regularisation appears in stochastic optimisation for instance when we do not reduce the learning rates to 0 over time. In that case, we are likely to not 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. 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 Figure 4.

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, but not to reduce the stepsize. Then, they well cover the area around the pot well without actually stopping. Similarly, when avoiding overfitting, we aim to get close to the minimiser of $\bar{\Phi}$, but actually need to avoid it to obtain an implicity regularised solution.

5 Concluding remarks

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 machine learning researchers try to overcome these challenges.

Teaching practice. Pot hitting may work well as a thought experiment to explain machine learning to curious people with a missing background in the mathematical or computer 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 [33] – especially to improve motivation, engagement, and creativity. An increment in motivation and engagement was also anecdotally observed by the author when he instructed groups of high school students in pot hitting during a series of outreach talks on AI. 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 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. Thought experiments have a long history in science [16] and can be considered effective [39] when used in teaching.

Some limitations. We have explained different components of optimisation methods in machine learning. These are usually carefully combined in practice – the Adam method [26], for instance, uses elements of SGD, HBM, and adaptive learning rates. The combination of different methods may appear simple in pot hitting, but may lead to complex interactions of the algorithms' effects in practice. Moreover, this note only describes challenges in the optimisation process underlying machine learning and blatantly ignores other issues that machine learning and AIs bring with them, such as: interpretability [36], robustness towards adversarial attacks [40], or ethical issues [29]. Finally, this note focusses on supervised learning and does not discuss other modes of machine learning, such as unsupervised learning [23], generative machine learning [18], or reinforcement learning [45]. Those come partially with similar but also with their own challenges.

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] P. L. Bartlett and S. Mendelson. Rademacher and Gaussian Complexities: Risk Bounds and Structural Results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

[2] D. Bertsimas and J. Tsitsiklis. Simulated Annealing. *Statistical Science*, 8(1):10 – 15, 1993.

[3] 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.

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

[5] 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.

[6] 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.

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

[8] 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.

[9] H.-H. Chou, C. Gieshoff, J. Maly, and H. Rauhut. Gradient descent for deep matrix factorization: Dynamics and implicit bias towards low rank. *Applied and Computational Harmonic Analysis*, 68:101595, 2024.

[10] F. Cucker and S. Smale. On the mathematical foundations of learning. *Bulletin of the American Mathematical Society*, 39(1):1–49, 2002.

[11] 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.

[12] D. Dasgupta and Z. Michalewicz. *Evolutionary Algorithms — An Overview*, pages 3–28. Springer, Berlin, Heidelberg, 1997.

[13] 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.

[14] J. Duchi, E. Hazan, and Y. Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011.

[15] Eurostat. Use of artificial intelligence in enterprises. `https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Use_of_artificial_intelligence_in_enterprises`, 2025.

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

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

[18] 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.

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

[20] P. C. Hansen. *Computational Aspects: Regularization Methods*, chapter 4, pages 53–83. SIAM, 2010.

[21] C. F. Higham and D. J. Higham. Deep Learning: An Introduction for Applied Mathematicians. *SIAM Review*, 61(4):860–891, 2019.

[22] His Majesty's Government, Department for Science, Innovation & Technology. Public Attitudes to Data and AI, Tracker Survey (Wave 4) Report. `https://www.gov.uk/government/publications/public-attitudes-to-data-and-ai-tracker-survey-wave-4/public-attitudes-to-data-and-ai-tracker-survey-wave-4-report#familiarity-with-ai-and-views-of-its-impact`, 2024.

[23] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning.* Springer, second edition, 2021.

[24] K. Jin, J. Latz, C. Liu, and A. Scagliotti. Losing momentum in continuous-time stochastic optimisation. *Journal of Machine Learning Research*, accepted, 2025.

[25] K. Jin, J. Latz, C. Liu, and C.-B. Schönlieb. A Continuous-time Stochastic Gradient Descent Method for Continuous Data. *Journal of Machine Learning Research*, 24(274):1–48, 2023.

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

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

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

[29] V. C. Müller. Ethics of Artificial Intelligence and Robotics. In E. N. Zalta and U. Nodelman, editors, *The Stanford Encyclopedia of Philosophy.* Metaphysics Research Lab, Stanford University, Fall 2025 edition, 2025.

[30] J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer New York, New York, NY, 1999.

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

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

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

[34] 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.

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

[36] 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.

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

[38] 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.

[39] 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.

[40] S. H. Silva and P. Najafirad. Opportunities and challenges in deep learning adversarial robustness: A survey. arXiv 2007.00753, 2020.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction.* MIT press Cambridge, 1998.

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

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

[48] W. Yang. Artificial intelligence education for young children: Why, what, and how in curriculum design and implementation. *Computers and Education: Artificial Intelligence*, 3:100061, 2022.

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