Pittsburgh Interdisciplinary Mathematics Review
Volume 2, December 2024, Pages 1–25
https://doi.org/10.5195/pimr.2024.38

Mathematics of Machine Learning: An Introduction

Stephan Wojtowytsch

(Communicated by Neil MacLachlan)

1. Introduction

Over the course of just a few decades, machine learning has grown into a force which shapes modern life perhaps as much as the combustion engine or wireless communication. As we drive to work, machine learning algorithms extract license plate numbers from images captured by automatic cameras at busy intersections. At work, they measure our productivity and govern our supply chains. In our personal lives, they power product recommendations on online shopping sites and suggestions on social media. In our homes and on our devices, they recognize our voices and our faces. They process our loan applications and evaluate our medical images. More globally, they stabilize power grids and assist in planning flight routes. There is not a space in our public and personal lives which machine learning has not at least begun to affect.

For a field as ubiquitous, it is fairly poorly represented in our common knowledge. This article aims at giving a high level introduction to some core tasks, ideas and methods of machine learning for readers who are familiar with at least some undergraduate mathematics. Advanced readers may get more from certain sections, but our goal is to present a self-contained picture which requires little knowledge beyond calculus in multiple variables and elementary linear algebra. For readers who have not taken many of the advanced classes, this may also motivate why certain fields of study are of interest in applications.

One big omission in this article are deep neural networks, i.e. the models which underly 'deep learning.' Due to their special importance, they will be discussed in a separate companion article.

1.1. What are Machine Learning, Data Science and Artificial Intelligence?

Broadly speaking, machine learning (ML) is the discipline of building models which automatically recognize patterns within data sets. As such, it has a fluid boundary with other fields such as classical statistics, data science and artificial intelligence. As a distinction, we could say that machine learning is more concerned with computational aspects than classical statistics: We are not content to know that an optimal estimator exists, we also need to know to what accuracy we can approximate it in what time.

Additionally, machine learning primarily concerns itself with high-dimensional data analysis, while the focus of classical statistics is broader.

Data science (DS) is a field focussed on working with data. As such, it employs methods of machine learning, but it may be faced with additional practical challenges. This may involve 'cleaning' data, finding duplicate entries, privacy protections and others. To distinguish between data science and machine learning, we can draw a parallel to the distinction between statistics and probability theory: Both are concerned with the behavior of random objects, but in statistics, we typically start with a dataset, while in probability theory, we make an assumption on the distribution of the random quantity. Similarly, in machine learning we have the luxury of developing methods for problems with certain characteristics without having to get our hands dirty with real data if we so desire – although it is considered good form to test theory on real world examples.

Finally, artificial intelligence (AI) is the field of building systems which display intelligent behavior, ideally by making intelligent decisions. Naturally, this is a field where techniques from machine learning are important to extract information about the problem or previous decisions, but not all AI is built on machine learning models: Sometimes intelligent behavior can at least partly be scripted directly into the model. Generally, AI is more on the engineering side than the science side, focussing on specific projects rather than general principles.

Naturally, these boundaries are vague and somewhat arbitrary, and different authors use terms differently. As the distinction is much less interesting than any of the fields themselves, let us get to it and learn about machine learning.

1.2. The curse of dimensionality

Why is it hard to find patterns in data sets? After all, we live in the age of 'big data'. Datasets with thousands or even millions of datapoints are common.

The main complication comes from the fact that data is typically very high-dimensional: A small color image for instance is described by 3 color channels (RGB) and 256×256 pixels, corresponding to $196,608 = 3 \cdot 256^2$ variables (dimensions). In natural language processing, words from the dictionary of the language are typically 'embedded' as vectors in a space of dimension $d \in [100, 1000]$. Even for the smallest application, each data point is typically described by at least a dozen dimensions.

If we want to find out for example how a function f behaves on a set U, it is a reasonable approach to cover U with a fine net of points $\{x_i, \dots, x_N\}$ and extrapolate $f(x)$ from $f(x_i)$ at a point x_i which is close to x. If $U = (0,1)$ is an interval, we achieve a resolution to the level $\varepsilon > 0$ with approximately $1/\varepsilon$ points:

$$\Big(\varepsilon, 2\varepsilon, 3\varepsilon, \dots, N\varepsilon\Big)$$

where $N = \lfloor 1/\varepsilon \rfloor$ is the largest integer below $1/\varepsilon$. In two dimensions, we need approximately $(1/\varepsilon)^2$ points to achieve a comparable resolution on the square $(0,1)^2$:

$$\Big\{(i\varepsilon, j\varepsilon) : 1 \leq i, j \leq \lfloor 1/\varepsilon \rfloor\Big\}.$$

In general in dimension d, the number of points needed for a grid of fineness ε scales like $(1/\varepsilon)^d$. This exponential dependence on d quickly becomes prohibitive: For a relatively course grid of fineness $\varepsilon = 0.1$ in dimension $d = 100$, we would need more points than there are particles in the observable universe (estimated at about 10^{80}).

So, even when we have millions of datapoints, they are geometrically 'sparse' and there are large regions of our data domain where we have little or no data. This makes it challenging to infer what happens away from the 'few' datapoints we have.

2. Some Flavors of Machine Learning

In machine learning, we are looking to recognize patterns and structure in datasets. Depending on what information we are given about our data, there are different kinds of problems that we may want to try and solve.

2.1. Supervised learning

Supervised learning is the most basic variety of machine learning. Our data comes in pairs (x_i, y_i) where x_i is the data point and y_i is a 'label', usually a real number or vector of real numbers, or a descriptor which we translate into a real number or vector. When there is a finite number of possible labels $\{y^1, \dots, y^k\}$, we speak of classification problems, trying to sort data x into the k different classes. If the label space is continuous (e.g. the real line), we speak of regression tasks.

In a classification problem, the data x might be images, which we can understand as elements in a very high-dimensional space, and the labels $1, \dots, k$ could correspond to k different classes we might sort them into. In a regression problem, we could receive the same set of images, but we might want to predict a score between 0 and 100 on how emotionally impactful they are to a user or how likely they are AI-generated.

The possibly simplest strategy in supervised learning is to select a class of parametrized functions $h(w,x)$ of parameters ('weights') $w \in \mathbb{R}^p$ and data $x \in \mathbb{R}^d$ and try to minimize the 'mean squared error' between model output and desired label over the dataset

$$L_{MSE}(w) = \frac{1}{2n} \sum_{i=1}^{n} \big|h(w, x_i) - y_i\big|^2.$$

This approach is classical in many applications with a function class of linear functions: $p = d$ and $h(w,x) = w^T x = \sum_{i=1}^d w_i x_i$ or polynomials for a one-dimensional variable x, i.e. $h(w,x) = \sum_{i=1}^{p} w_i x^{i-1}$. In a more modern setting, we could use neural networks here, which we will discuss in a follow-up article.

For classification problems, the mean squared error (MSE) may not be the optimal criterion to minimize. For instance, consider a binary classification problem (i.e. there are only two labels 1 and -1) with four data points

$$(x_1, y_1) = (-16, -1), \qquad (x_2, y_2) = (0, -1), \qquad (x_3, y_3) = (4, 1), \qquad (x_4, y_4) = (8, 1)$$

and an affine linear classifier $h(w,x) = w_1 x + w_2$. Since we know that there are only two labels, we only need to get the sign right: We say that the function $h(w,\cdot)$ predicts label 1 at the point x if $h(w,x) > 0$ and label -1 if $h(w,x) < 0$. We note that

$$\begin{aligned}\nabla L_{MSE}(w) &= \nabla_w \frac{1}{8}\sum_{i=1}^{4} \left|w_1 x_i + w_2 - y_i\right|^2 = \frac{1}{4}\sum_{i=1}^{4}(w_1 x_i + w_2 - y_i)\begin{pmatrix} x_i \\ 1 \end{pmatrix} \\ &= \frac{1}{4}\begin{pmatrix} \sum_{i=1}^4 x_i^2 & \sum_{i=1}^4 x_i \\ \sum_{i=1}^4 x_i & \sum_{i=1}^4 1 \end{pmatrix}\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} - \frac{1}{4}\sum_{i=1}^{4} y_i \begin{pmatrix} x_i \\ 1 \end{pmatrix} = \begin{pmatrix} 84 & -1 \\ -1 & 1 \end{pmatrix}\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} - \begin{pmatrix} 7 \\ 0 \end{pmatrix}.\end{aligned}$$

The minimizing parameters w_1, w_2 can be obtained by setting the gradient to zero, which corresponds to solving the linear system:

$$\begin{pmatrix} 84 & -1 \\ -1 & 1 \end{pmatrix}\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} 7 \\ 0 \end{pmatrix} \qquad \Leftrightarrow \qquad \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \frac{1}{84+1}\begin{pmatrix} 1 & 1 \\ 1 & 84 \end{pmatrix}\begin{pmatrix} 7 \\ 0 \end{pmatrix} = \frac{7}{85}\begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

We see that $h(w,x_2) = h(w,0) = w_2 > 0$, but we know that the label of x_2 should be -1! Based on our classifier $h(w,x)$ we would mislabel one of our known datapoints. The outlying datapoint at -16 (which is far away from the others) has driven us to misclassification. On the other hand, we see that it is possible to find a classifier in the class of affine linear functions which has the correct sign at all known data points, for instance $h((1,-1),x) = x - 1$. For this reason, we sometimes minimize different 'loss functions' of the form

$$L(w) = \frac{1}{n}\sum_{i=1}^{n} \ell\Big(h(w,x_i), y_i\Big)$$

instead. A popular example is the 'logistic loss'

$$L_{log}(w) = \frac{1}{n}\sum_{i=1}^{n} \ell_{log}\Big(h(w,x_i), y_i\Big), \qquad \ell_{log}(h,y) = \log\Big(1 + \exp(-yh)\Big)$$

or its multi-class version, cross-entropy loss.[1] Here, $\log(1+\exp(z)) \geq \log(2)$ if $z \leq 0$ (i.e. the loss is at least $\log 2$ if the label y and the output $h(w,x)$ of the classifier have

[1] Unfortunately, the terminology can be confusing – some authors refer to ℓ as the loss function, while others – especially when it comes to training – refer to L as the loss function.

different signs) but $\log(1+\exp(z)) \approx \exp(z) \ll 1$ if z is large and negative. Thus logistic loss drives $h(w,x_i)$ to have the same sign as y_i at all given data points, if possible, and then increases the magnitude of the output of h. If we drove L_{log} to zero for the example dataset above, after a while we would find a (large) multiple of the 'maximum margin' classifier $x-2$ which changes sign in the middle of the two data-points $x_2 = 0$ and $x_3 = 4$ where the class changes.

In binary classification, we exploit that only the sign of $h(w,x_i)$ matters, not its magnitude, but also for regression problems there are situations where MSE loss is not the best choice. The power $q = 2$ in the loss function $\ell_q(h,y) = |h-y|^q$. is chosen because it makes the computations easiest. Sometimes, we replace it with a smaller number $1 \leq q < 2$ to be more stable against outliers: If there is a single point which really does not fit the pattern, the square amplifies the impact of the largest misfit considerably. Especially if we do not trust the labels fully, $q = 1$ may be the better choice.

Of course, MSE loss also has its advantages. Since it is quadratic, to find the minimizer, we only need to solve a linear equation (usually in many variables). By comparison, logistic loss does not have a minimizer if perfect classification is possible as we can always multiply w by a large positive number and reduce L_{log} slightly. This can make it harder to work with logistic loss, and it is non-trivial to find functions of low loss. This is the problem of optimization.

Another question arises from the use of a small dataset: If we perform well and classify our four data points correctly, do we perform well on the true problem? This is the problem of generalization. It is particularly important when there are many different parameters w for which $h(w,x_i)$ perfectly fits all the labels y_i, for instance because we have few data points, but many variables per data point (and thus also many parameters in the class of linear functions). We will come back to both in the context of deep learning.

2.2. Unsupervised learning and hybrid settings

Assigning labels y_i to data points x_i can be expensive, time-consuming and sometimes not what we are looking for. For instance, we may simply want to reduce the dimension of our dataset in order to visualize it, or we may search for a good representation of our data which can be used 'downstream' for various supervised learning tasks, potentially with several different sets of labels. We can picture an advertising agency which is trying to identify different groups within a population for a range of campaigns, not just a single assignment.

Two prototypical tasks of unsupervised learning are dimension reduction and clustering. Dimension-reduction allows us to find a simpler representation of our data which requires less storage space and captures the main relational features of our task. In clustering, we for example want to find 'communities' in a social network given only the information who is friends with whom (and possibly how often they liked each others posts etc), but crucially no labels.

The boundary between supervised and unsupervised learning can be fluid. For example, if our data lives in a high-dimensional space $\mathbb{R}^D$, we can treat x_i as both data and label and try to minimize

$$\frac{1}{n}\sum_{i=1}^{n}\left|f(g(x_i)) - x_i\right|^2$$

in classes of functions $g : \mathbb{R}^D \to \mathbb{R}^d$ and $f : \mathbb{R}^d \to \mathbb{R}^D$, i.e. we are trying to 'learn' the identity map on our data domain while passing through a much lower-dimensional space $\mathbb{R}^d$ with $d \ll D$. As in the supervised case, we typically parametrize f, g by elements of Euclidean spaces $\mathbb{R}^{p_f}, \mathbb{R}^{p_g}$. If f, g are linear functions, this is a classical technique known as *principal component analysis* (PCA), whereas we call $f \circ g$ an *auto-encoder* if both f and g are parametrized by neural networks.

There are also many hybrid settings between supervised and unsupervised learning, such as *semi-supervised learning* (we have labels for only a small subset of our datapoints), *active learning* (we have the budget to label a subset of our datapoints which we can choose ourselves), *self-supervised learning* (we have pairs of similar and pairs of dissimilar data, but no further information) or scenarios where we have labels of varying degrees of reliability (labelled by expert vs labelled by lay person).

We will see some examples in this setting in Section 3.

2.3. Reinforcement Learning

Both supervised and unsupervised learning tasks generally start with a dataset. Reinforcement learning is different in that it starts with an 'environment' in which an 'agent' can take actions and observe the environment's reaction. This more closely mimics human learning where for us, the environment would be the physical world and the agent our conscious self. If I take the action 'eat at Primanti's', the environment reacts by changing my state from hungry to full. If I take it immediately again, it reacts by changing my state from full to sick.

Reinforcement learning is a useful framework to think about games of strategy: I move a piece (action), my opponent/the environment moves a piece (reaction). The actions available to me and how the environment reacts depend on the current state of the game board, and often I have no way to predict exactly how the environment will react. My goal is to maximize a reward in the future (I win) where (partly for practical reasons) we value the reward more highly the earlier it occurs: We maximize the expectation of the value function

$$V(a) = \sum_{t=0}^{\infty} \rho^t R_{t+1}$$

over the actions a (moves) available to us. Here $\rho \in (0,1)$ is a discount factor which emphasizes early rewards and R_t is the reward in the t-th time step. In the simplest case, the reward could be 1 for a winning move, -1 for losing and 0 otherwise, but in many cases, there is a more complicated incentive structure in every step (think of investing money and receiving a payout every year, not just at the end). Of course, a lot is hidden in this simple expression: We need to consider not just the immediate payoff of an action a, but also which state it takes us to next...

There is a crucial difference between reinforcement learning and other learning tasks: We initially know nothing about the environment or the actions available to us. We are playing a game without understanding the rules. The initial phase of play is 'exploration': We are trying to figure out what we can do and how the environment reacts. The second phase is 'exploitation': Once we have found a few moves that work well, we want to gravitate towards those. The tradeoff between exploration and exploitation is a main motif in the analysis of reinforcement learning. Once we understand the game, we want to mostly rely on actions which we know to work well, but every once in a while, we should try something new to see if there is an even better option.

Reinforcement learning is simple if there is a small set of states we can be in and actions we can take, but it becomes very complicated if the action and state spaces are large (Go rather than chess, financial markets) or even infinite (self-driving cars). The main advance of the last decade was the use of neural networks to parametrize our strategies (or, in the parlance of reinforcement learning, 'policies').

Of course, many tasks can be approached in various ways. For instance, AlphaGo (the first AI to beat a top human player in the strategy game Go) was trained initially using supervised learning and transitioned to reinforcement learning once it had already become a strong player. In the supervised phase, it learned a probability distribution on the set of actions (conditional on the state of the game) which prioritized moves that strong players would take in a given situation. In the reinforcement phase, it played against other AIs and itself.

The pre-training gives us a 'warm start' for the reinforcement learning process: Rather than starting to learn as a toddler, we start as a teenager with a fairly solid grasp of the world around us.

3. Advanced Mathematics in Machine Learning: An Example

As an example, let us highlight connection between topics which may at first glance be surprising: Partial differential equations, differential geometry, and machine learning. The appearance of deep mathematics of all varieties in machine learning is not by chance: As machine learning grew into the powerful tool it is today, scientists from all fields became fascinated and flocked towards ML to try and understand its promise. Some were curious if ML could solve their problems, others were asking the more foundational questions: Why does it work? Can we guarantee that it works? What are its limitations? And, could we make it work even better?

Mathematicians were generally in the latter camp. As researchers with vastly different areas of expertise developed an interest in machine learning, new methods and techniques developed accordingly. This means that no matter which class you take, it may come in handy somewhere down the road if you decide to pursue ML. Indeed, its interdisciplinary nature within mathematics and beyond is a large part of what makes machine learning so exciting to work in today.

We first review some of the mathematical motivations.

3.1. Partial Differential Equations

Partial differential equations (PDEs) are one of the most versatile tools of mathematical modeling. In the sciences they are used to express the balance between rates of change of various quantities over space and time, such as the diffusion of heat in a medium, force balances in continuum mechanics or reaction rates in chemistry. They also have curious links to random processes such as Brownian motion and therefore also appear in very different contexts like mathematical finance.

We consider an example from physics, which will guide us below in developing a model for data science. Imagine a soap film spanned by a wire. Mathematically, this we model this as a surface given by the graph of a function $u : \overline{U} \to \mathbb{R}$, where $U \subset \mathbb{R}^2$ is an open set (the region below the film). The wire is given by the 'boundary condition' $u(x) = g(x)$ for some function $g : \partial U \to \mathbb{R}$ (where ∂U is the boundary of U). But what describes the shape of the surface inside the wire?

A fairly good first model is to say that the soap film wants to have area as small as possible (i.e. as small as is compatible with the boundary condition). Knowing calculus, we can calculate the area of the graph of u as

$$A(u) = \int_U \sqrt{1 + \|\nabla u\|^2}\,\mathrm{d}x \approx \int_U 1 + \frac{1}{2}\|\nabla u\|^2\,\mathrm{d}x$$

where we replaced the square root by its first order Taylor approximation at 1. The first expression is more physical, the second one easier to work with. If we have u which minimizes $E(u) = \frac{1}{2}\int_U \|\nabla u\|^2\,\mathrm{d}x$ among all functions with boundary values g, then for all functions ϕ which are zero on ∂U and all $t \in \mathbb{R}$, we know that $e_\phi(t) = E(u + t\phi)$ has a minimum at zero and thus

$$
\begin{aligned}
0 = e'_\phi(0) &= \frac{d}{dt}\bigg|_{t=0} \left(\frac{1}{2} \int_U \|\nabla u\|^2 \,\mathrm{d}x + t \int_U \langle \nabla u, \nabla \phi \rangle \,\mathrm{d}x + \frac{t^2}{2} \int_U \|\nabla \phi\|^2 \,\mathrm{d}x \right) \\
&= \int_U \langle \nabla u, \nabla \phi \rangle \,\mathrm{d}x = \int_U \nabla \cdot \big(\phi \cdot \nabla u\big) - \phi\, \Delta u \,\mathrm{d}x = \int_{\partial U} \phi\, \partial_\nu u \,\mathrm{d}A - \int_U \phi\, \Delta u \,\mathrm{d}x.
\end{aligned}
$$

Here, we used the divergence theorem to convert a volume integral into a boundary integral, much like the fundamental theorem of calculus in one dimension. The boundary integral is zero since ϕ is zero on the boundary, so $\int_U \phi\, \Delta u \,\mathrm{d}x = 0$ for all functions ϕ which are zero on the boundary. By a result known as the 'Fundamental Lemma of the Calculus of Variations', we can conclude that u solves the PDE $\Delta u = 0$ in U and $u = g$ on ∂U. This partial differential equation is *linear* in u since the 'energy' E is quadratic. This partial differential equation has a unique solution, so rather than minimizing E in an infinite-dimensional function class, we only have to solve a linear PDE, which is much like solving a linear system of equations in linear algebra. Efficient techniques of numerical linear algebra for this purpose have been studied for many decades. Working directly with the square root, we would deal with the much more complicated PDE $\nabla \cdot \big(\nabla u / \sqrt{1 + \|\nabla u\|^2}\big) = 0$.

Let us briefly note that the geometry of U has massive influence on the expected behavior. For instance, if the boundary condition of u is $+1$ on one side of a handlebar shape, -1 on the other and (in a slight departure from our previous model) unknown in between, then the function u is mostly constant on the two blobs and transitions where the figure is thin in between. This way, the set where the gradient is large has small measure. If U were a disk, on the other hand, there is no 'cheap' way for a short boundary. We will come back to these ideas below in the context of data science.

3.2. Differential Geometry

Differential geometry is one of the fascinating fields which start as the purest of mathematics and only later turn out to be essential in applications. As a field distinct from general geometry, it can be traced back roughly to Carl Friedrich Gauss, who considered a statement on the curvatures of surfaces in $\mathbb{R}^3$ his finest accomplishment or *theorema egregium* ("Remarkable Theorem", 1827). Following Gauss' work, his PhD student Bernhard Riemann studied abstract n-dimensional 'surfaces' which were not embedded in any ambient space. This was an entirely esotheric pursuit until it became the language of general relativity in the early twentieth century.

The objects of differential geometry are so-called 'manifolds': Spaces which look like the flat Euclidean space locally, but which can have a more complicated global structure. Imagine, for instance, the surface of a sphere or a doughnut (a 'torus') or the chassis of a car. All of these look essentially like the plane to an ant, but more globally, they have a very different shape (topology) from the Euclidean $\mathbb{R}^2$. Even locally, unless we zoom in to an infinitesimal length scale, we can see a geometric difference: Curvature.

Since manifolds locally look like a slightly distorted version of Euclidean space, we can develop a geometric theory of manifolds which allows us to carry over important notions of vector calculus: Gradients, divergences and Laplacians. We can therefore repeat the arguments of the previous section more abstractly and solve PDEs on manifolds. One big difference arises: There are manifolds such as the sphere or the torus which do not have a boundary! In many cases, this makes things easier for us in practice (at the expense of a more abstract underlying framework).

3.3. Application in semisupervised learning

More recently, differential geometry entered the stage in data science where we observe that real data has a sort of low-dimensional structure in a much higher-dimensional ambient space. For instance, if we were to choose the color values for the pixels of an image independently, we would create some form of white noise, not a real image of anything. Sensible images occupy a vanishingly small region within the set of all images of a certain resolution. The 'manifold hypothesis' in ML posits that the set of sensible data in various applications is well-described by a smooth manifold of moderately large dimension in a much higher-dimensional ambient space, e.g. a seventeen-dimensional 'surface' in a seven hundred-dimensional space.

Given a finite collection of datapoints $x_1,\dots,x_n$ in $\mathbb{R}^d$, we can imbue them with a structure as a mathematical object known as a 'graph': The points x_i are known as 'vertices' and two points x_i, x_j can be connected by an 'edge' e_{ij}. Fairly common ways to construct a graph from a dataset in $\mathbb{R}^d$ would be to connect all points, connect points if they are closer than a cut-off distance ε, or connect every point to its k nearest neighbors for some $k \ll n$. We can then also associate a 'weight' w_e to an edge e which should be large if the points are close and small if they are far apart (for instance $w_{e_{ij}} = \exp(-\|x_i - x_j\|^2)$).

The edge weights provide us with an analytic structure on the graph, and we can define 'differential operators' such as the 'graph-Laplacian' of a function u on the vertices as

$$\Delta u(x_i) = \sum_{j\neq i} w_{ij}\Big(u(x_j) - u(x_i)\Big)$$

where the sum goes over all indices j such that x_i and x_j are connected by an edge. In a special case, this gives us a familiar object! If the vertices are equidistant on a line $x_i = ih$ with a small length-scale $h > 0$ and the weights w_{ij} are $1/h^2$ if $j = i \pm 1$ and zero otherwise, then

$$\Delta u(x_i) = \frac{u\Big((i+1)h\Big) - u(ih)}{h^2} + \frac{u\Big((i-1)h\Big) - u(ih)}{h^2} = \frac{u\Big((i+1)h\Big) - 2u(ih) + u\Big((i-1)h\Big)}{h^2}$$

is the standard difference quotient approximation of the second derivative of u at x_i! The same works on a square grid in two (or higher) dimensions, where we obtain a difference quotient approximation of the Laplacian. More generally, if we assume that the data points x_i from which we build our graph are sampled independently from a probability distribution on a manifold, we can show in a precise way that the graph Laplacian converges to a differential operator on the manifold as the number of samples approaches infinity. In the simplest case, the limiting object is the natural 'Laplacian' on the manifold, but in general, it will also depend on the probability distribution.

Now, let us return to the semi-supervised learning setting: We have many data points x_i, but we only know the labels y_i for a few of them. We are looking for a function u on the set of data points which predicts a label y in such a way that if x_j is close to x_i and the label of x_i is known to be y_i then also $u(x_j)$ is close to $u(x_i) = y_i$. In between known points, the function should not be too wild and oscillatory. We are looking for a function which is as 'flat' as the known labels permit it to be – very much like a soap film with a wire at the boundary. There are many ways to specify what kind of function u we are looking for, but one of the easiest is to enforce that the square gradient should not be large 'on average'. This is exactly analogous to our PDE example.

For analytical purposes (but not necessarily numerical implementations), we can assume that all vertices are connected by setting the weights to zero if we create additional connections this way. We then try to set up an 'energy' to be minimized:

$$E(u) = \frac{1}{4} \sum_{i,j} w_{ij} \Big| u(x_i) - u(x_j) \Big|^2$$

subject to $u(x_i) = y_i$ at all vertices where we know the label. Heuristically, we minimize the square gradient of u on the graph subject to a 'boundary condition' given by the known labeled data. The weights tell us how important it is that $u(x_i)$ and $u(x_j)$ should be close together – recall that the weights are large if $\|x_i - x_j\|$ is small.

For simplicity, abbreviate $u_i := u(x_i)$. Then, if u minimizes E and x_I is not one of the labeled pints, we find that

$$
\begin{aligned}
0 = \partial_{u_I} E(u) &= \frac{1}{2}\sum_{i,j} w_{ij}(u_i - u_j)^2 = \frac{1}{2}\sum_{i,j} w_{ij} 2(u_i - u_j)\big(\delta_{iI} - \delta_{jI}\big) \\
&= \frac{1}{2}\sum_{j} w_{Ij}(u_I - u_j) - \frac{1}{2}\sum_{i} w_{Ii}(u_i - u_I) = \sum_{j} w_{Ij}(u_I - u_j) = -\Delta u(x_i).
\end{aligned}
$$

Thus also on a graph, minimizing a suitable average of the 'square gradient' (here: the squared difference of u at neighboring points), we can instead solve a partial differential equation involving the graph Laplacian. In practice, this 'PDE' boils down to solving a system of linear equations:

$$
\begin{cases} u(x_i) &= y_i \quad \text{if } y_i \text{ is known} \\ \Delta u(x_i) &= 0 \quad \text{otherwise.} \end{cases}
$$

In general, these linear systems may comprise thousands of equations, so ideas of abstract linear algebra and numerical linear algebra become crucial ingredients of machine learning in practice. We observe that the fact that we had unlabelled data points was crucial in constructing our predicted labels $u(x_i)$. Even if we do not know the correct answers for all the questions, it may be very helpful to know what we will be asked about.

Finally, let us note that there are other notions of a graph-Laplacian, while there is only one version of the Laplacian in the Euclidean case. For a more detailed introduction to analysis on graphs, see e.g. [Tan24].

3.4. Why do we like PDE-inspired algorithms?

We have seen that we can use ideas from fields like partial differential equations and differential geometry to inspire techniques in machine learning. The question remains: Why would we want to go this route? And, even if the model seems sensible, why do we care about similar models in these more abstract fields?

When considering partial differential equations on domains in $\mathbb{R}^d$, things are conceptually more complicated than when we think about the graph Laplacian: We deal with functions instead of vectors and with differential operators rather than matrices. Still, our life becomes easier in other ways: We don't have to deal with a positive length scale in a graph and can instead consider the homogeneous and isotropic Euclidean space. This allows us to obtain geometric insights into the structure of solutions which can enable a better interpretation of algorithms on graphs.

For instance, imagine that we are given 'labels' $u = 0$ on the boundary of the unit ball $B_1(0)$ in $\mathbb{R}^d$ and $u = 1$ at the center of the ball. If $d \geq 3$ and u^* is *any* function such that $u(0) = 1$ and $u(x) = 0$ if $\|x\| \geq 1$, then also $u_R(x) := u(Rx)$ satisfies the correct boundary conditions for any $R \geq 1$ and

$$E(u_R) = \int \|\nabla u_R\|^2 \, \mathrm{d}x = \int R^2 \|\nabla u(Rx)\|^2 \, \mathrm{d}x = R^{2-d} \int \|\nabla u(Rx)\|^2 R^d \, \mathrm{d}x = R^{2-d} E(u)$$

by the change of variables formula in d dimensions. The factor R^d arises since we need the determinant of the Jacobian matrix $R \cdot I_{d\times d}$, which is precisely R^d. Clearly $\lim_{R\to 0^+} E(u_R) = 0$ and u_R converges (pointwise) to the function which is 1 at the origin and 0 everywhere else. By considering the specific example

$$u_R(x) = \begin{cases} 1 & \|x\| \leq e^{-R} \\ \frac{\log\|x\|}{\log(e^{-R})} & e^{-R} \leq \|x\| \leq 1 \\ 0 & \|x\| \geq 1 \end{cases} = \begin{cases} 1 & \|x\| \leq e^{-R} \\ -\frac{\log\|x\|}{R} & e^{-R} \leq \|x\| \leq 1 \\ 0 & \|x\| \geq 1 \end{cases}$$

we see that the same is true in dimension $d = 2$ since

$$\int \|\nabla u_R\|^2 \, \mathrm{d}x = \int_{B_1(0)\setminus B_{e^{-R}}(0)} \left\| \frac{1}{R} \log'(\|x\|) \frac{x}{\|x\|} \right\|^2 \mathrm{d}x = \frac{2\pi}{R^2} \int_{e^{-R}}^1 \frac{1}{r^2} \, r \, \mathrm{d}r = \frac{2\pi}{R}$$

approaches zero as $R \to \infty$. In the parlance of applied analysis, we have shown that a single point has 'harmonic capacity zero', i.e. it is too small to prescribe a boundary condition on. We observe something similar on graphs: If the set of labelled points is too small, the 'function' u on the graph is not a smooth sheet spanned by its boundary data, but an almost constant function with a few heavily localized spikes. Insights from PDE theory allow us to develop remedies, such as Poisson learning [CCTS20] or minimizing $\int \|\nabla u\|^p \, \mathrm{d}x$ for $p \gg 2$ instead.

3.5. Unsupervised learning: Embeddings

There are other applications of these 'PDE' ideas in machine learning. For instance, we have claimed above that the shape of a domain influences the way PDEs behave and favor for instance rapid transitions in narrow corridors rather than going through the center of the domain, if possible.

This 'domain dependence' is present even for differential operators like the Laplacian $\Delta = \sum_{i=1}^d \partial_i \partial_i$ which look universal and domain-independent. This is perhaps not surprising – the same microscopic laws govern fluid flows in the ocean, in straight pipes and in porous media, but their behavior is quite different. The presence of physical boundaries has massive influence on the behavior of solutions.

Studying the behavior of the Laplacian operator of a domain or manifold is called *spectral geometry* because we study the eigenvalues and eigenfunctions (the 'spectrum') of the Laplacian operator. This is defined much as in linear algebra for matrices: We look for functions u and real numbers λ such that $\Delta u = \lambda u$ with the condition that $u = 0$ on the boundary of the domain (if non-empty). We can guarantee the existence of these as in linear algebra since the Laplacian is symmetric and negative (semi-)definite:

$$\int_U (\Delta u)\, v \, \mathrm{d}x = -\int_U \langle \nabla u, \nabla v \rangle \, \mathrm{d}x = \int_U u \, \Delta v \, \mathrm{d}x, \qquad \int_U u \Delta u \, \mathrm{d}x = -\int_U \|\nabla u\|^2 \, \mathrm{d}x \leq 0.$$

Similarly, the graph Laplacian is represented by a symmetric and negative semi-definite matrix. The starting point of spectral geometry is the question whether you can 'hear the shape of a drum': The eigenvalues of the Laplacian tell us the natural frequencies of vibration when hitting a membrane of the shape U (using the related 'wave equation', another common PDE).

If we accept the premise that the eigenfunctions of the Laplacian tell us something about the geometry of a space, we can use the idea for data visualization: For instance, given a graph of data points $x_1, \dots, x_n$ in a very high-dimensional space $\mathbb{R}^d$, we can visualize them in much lower dimension k by reporting only the values which the first k eigenfunctions $u_1, \dots, u_k$ (eigenfunctions corresponding to the eigenvalues closest to zero) take at these points:

$$\big(u_1(x_1), \dots, u_k(x_1)\big), \;\; \dots \;\; , \big(u_1(x_n), \dots, u_k(x_n)\big).$$

This yields a k-dimensional representation of our n data points in a way which only depends on the geometry of the dataset. This can be used for visualization (when k is two or three) or as a representation for further downstream tasks. Such an embedding can give us much more insight than just plotting the coordinates directly, for instance. The eigenfunctions corresponding to larger eigenvalues tend to be more oscillatory and less informative.

Differential geometry is important in its own right, not just when paired with PDEs. For instance, dimension reduction techniques which embed data into hyperbolic spaces (manifolds of constant negative curvature) can often embed into much lower dimension than projections into Euclidean spaces. This is true for data with hierarchical structure (plane is a vehicle, Boeing planes are planes, Boeing 737 is a Boeing plane, ...) and even some networks of data (such as social networks).

4. Machine Learning and Society

Every new technology in human history has brought with it societal changes, most of them unforeseen when the technology first became available. The steam engine revolutionized manufacture and transportation. Electricity quite literally illuminates the human race. Computer was a job title before it became a piece of technology. Over the course of less than 20 years, the internet grew from a scientific project into the information highway it is today, and the place where many of us spend a substantial portion of our lives.

Due to technological progress, many professions have changed over the years, others have disappeared while new ones have been created. Predictions have often proved incorrect in the long run. The influential economist John Maynard Keynes predicted in [Key30] that by now, the work week should be a mere 15 hours long due to automation. Needless to say, he was wrong [Kes15].

Artificial intelligence may well be the latest in a long sequence of world changing inventions, and as much as any discussion of its impact may be speculation, we must at least try to consider the impact of these advances as they occur, and try to anticipate what their impact might be over time.

At the conference on Neural Information Processing Systems (NeurIPS) in 2017 (one of the largest and most prestigious conferences on machine learning), Ali Rahimi, a research scientist at Amazon, was awarded the 'test of time' award for an article he had published at the same venue ten years earlier. In a very memorable (and somewhat controversial) acceptance speech, he argued:

> *We've made incredible progress. [...] There's a self-congratulatory feeling in the air. We say things like "machine learning is the new electricity." I'd like to offer another analogy: Machine learning has become alchemy.*
>
> *Now, alchemy is okay. Alchemy is not bad. [...] Alchemists invented metallurgy, ways to dye textiles, our modern glass making processes and medications. Then again, alchemists also believed they could cure diseases with leeches and transmute base metals into gold.*
>
> *If you're building photo-sharing systems, alchemy is okay. But we're beyond that now. We're building systems that govern healthcare and mediate our Civic dialogue. We influence elections. I would like to live in a society whose systems are built on top of verifiable, rigorous, thorough knowledge and not on alchemy.*

More recently, chatbots have laid bare the enormous potential and terrible shortcomings of systems built on machine learning: They perform incredibly well on average, but fail catastrophically in specific cases. This is likely not a coincidence: There are settings where we can prove rigorously that it is easy to find a model based on data which performs well on a task on average, but nigh impossible to find one which performs well in the worst case scenario (namely, we hit the curse of dimensionality [CLW23]). Perhaps more insidiously, machine learning can fail us when it works precisely as intended.

4.1. Energy and Materials Cost

Most major advances in artificial intelligence in recent years, at least in experimental research and practical implementation, have come from a small selection of companies and well-funded start-ups which cannot be matched by other places of research. In large part, this is due to the enormous amount of resources which advanced AI models need during training. We will discuss what neural networks are and why they are so computationally intensive in a forthcoming companion article, but for now, let us look at some figures without thinking about where the energy goes.

The two main requirements of AI are computer chips and electricity. GPT-4 – the model underlying the current version of ChatGPT, the follow-up to the one that went viral – is estimated to have about 176 trillion parameters. Even to store the parameters, this requires about 2.75 TB. In order to train such large models, we need to split them up over multiple machines, and most AI computations take part in large computing centers which require electricity not just to run the machines, but also for cooling.

The effects are noticeable also during deployment. A query to ChatGPT consumes about 10 times the energy of a google search in a low estimate [Sac24] and about 25 times according to [New24]. This is not a trivial quantity, and the energy used by companies such as Microsoft, Google and Meta have increased by about one third to two thirds over the last four to five years, largely due to AI. This has even prompted companies to design data centers which generate their own, clean energy.

In a few years, the energy consumption of AI overall is estimated to account for about 0.5% of electricity used worldwide, roughly the equivalent of a small country such as Sweden or Argentina. For comparison, data centers (which power cloud computing and search engines) currently use about two to three times that, and computations for crypto mining use another 0.4%. Much more detailed information can be found in articles such as [McQ23, Erd23, SMW+24, Gel24].[2]

While energy costs are high, at least in parts of the world, chips have been a more serious limiting factor globally. The AI boom has coincided with conflicts around the planet and a global pandemic, both which have heavily affected the semi-conductor industry and lead to the chip shortage following 2020. Certainly, the issue has not been helped by the fact that the same resources – chips and power – have fueled another computationally intensive pursuit: 'mining' for crypto-currencies.

At this time, much of our energy comes from finite energy sources, which are additionally implicated in the destruction of our environment. This, together with the use of rare materials to produce chips has naturally led to environmental concerns about the cost of AI. It is worth remarking, though, that these issues are also felt throughout the industry in monetary costs to the extent that many AI companies are currently not profitable. While companies like Apple, Meta, Microsoft and Google have other sources of revenue to invest in a speculative future industry, most smaller companies

[2] Many academic institutions have an institutional subscription to various newspapers which allows members to access articles for free. At the time of writing, you can access the New York Times that way if you are a student at Pitt.

are kept afloat purely by venture capital.

4.2. So we can solve it. Should we?

Modern AI has been incredibly successful in solving problems which were previously considered intractable: It enabled the first computer program that could beat top human players at the strategy game Go and facilitated significant advances in understanding protein folding.

There are many more harmless or beneficial examples, but also a few that should give us pause: Searching a person on the internet by an image. Generating fake images and movies for propaganda. Deblurring of images. And some technologies may be helpful or harmful, depending on our trust in the actor who uses them: How do we feel about real time face recognition for cameras in public spaces?

Fake images have entered the political discourse in the United States [Vig24] and popular culture [Wik24] in disturbing ways, along with the accusation that real images are fake [EKW+24]. Face recognition has been used in New York to bar lawyers from entering a venue whose owner was involved in litigation with them [HK22] and in Moscow to identify and detain dissidents [STF21].

If a stranger takes a photo of us in the street, do we want them to be able to find our names? Where we study, where we work, what we wrote on the internet as teenagers? [Hil20, Hil22] go into much greater detail on this example.

Beyond Skynet-doomsday scenarios, there are real threats for AI to erode trust and privacy in our society. The ability to generate convincing fake audio and video in a 'post truth' society invites abuse by bad actors, and technology has moved much faster than the legal framework to contain them and harness their power for social good. For now at least, we are relying on scientists themselves to not develop detrimental technology that they could sell to questionable parties for profit, and relying that no questionable characters learn to code. If the rise of AI-aided scams [Tho24] and bullying by deep fake pornography [Kra24] is anything to go by, this may be a naive approach.

The question of ethical behavior by AI scientists is amplified by a drive to make AI accessible, publish code, and enable non-experts to modify models. If AI is accessible to us all, we *all* must behave ethically. Conversely, keeping AI under the control of a small number of companies comes with its own, different set of problems...

4.3. Privacy and Intellectual Property

There are cases where a technology may not directly be harmful, but we venture into murky ethical and legal waters in its creation – just think about electric cars and laptop computers which rely on lithium-ion batteries for which materials are at times mined in ways that displace communities and poison the environment [Gre22] and processed by poorly protected migrant workers [SH23].

Beyond energy and hardware, the thing that machine learning models require is data. The more data we have, the better models we can create. Large Language Models are now at the point where there is not enough text in the English language (and in particular, high quality text).

The New York Times has sued OpenAI for using their articles without consent to train AI models. Meta considered purchasing an entire publishing house to gain access to their material, before deciding that they would not be competitive if they took this time and ostensibly used copyrighted material following OpenAI's 'market precedent'. Both Google and OpenAI are believed to have transcribed spoken word from YouTube videos for training material, potentially violating user agreements. Datasets for image generating tools like DALL-E, Midjourney, and Stable Diffusion have been generated by automated scraping tools for online images and have been found to contain artists' copyrighted works, revenge pornography, and propaganda images of executions by ISIS. At the time of writing and to the best of my knowledge, there is no definitive ruling on whether companies violated the law in the creation of their training data sets.

A much more detailed description of the struggle to create training datasets for large language models can be found in [MKF+24].

4.4. Dataset bias: Quis custodiet collectores datorum?

Clearly, data is needed in machine learning, and it is not easy to find enough of it. At its heart, machine learning uncovers patterns in datasets. Based on its discoveries, we aim to make decisions which will impact the real world. Crucially, we must bear in mind that at no point during its creation or deployment has the machine learning model had access to the real world, and has only ever seen it through the lens of a dataset. By necessity, any collection of data includes the biases, conscious and subconscious, of the person who assembled the dataset and the society which produced it. A large dataset is not necessary a good dataset.

Some of these biases are easy to catch. Early machine learning models trained to classify images of human faces according to gender used blond hair as an indicator of female gender since blonde women were overrepresented in training data taken from online image collections. While efficient on their training data, such biases would make a model vulnerable to deliberate manipulation using hydrogen peroxide and hair dye.

There are many great sources documenting biases based on race and gender in high impact areas such as medicine or criminal justice [Pet85, Kov19, Fri24]. Even expert judgements are, at times, shockingly inaccurate. When present in datasets, these biases are encoded in machine learning models and given a veneer of scientific accuracy. Both medicine and criminal justice have seen applications of machine learning. While seemingly unmoved by human motives, these models are at best as objective as the people involved in their creation.

Humans can be fantastically irrational and biased creatures. A priori, a computer has no preconceptions, yet a computer models taught by humans to replicate human assessments has all the fallibility of a human and all the hidden pitfalls of a black box model trained using undisclosed methods.

4.5. Explainability: errare artificiale est

We have discussed, on a high level, several machine learning models. Some of them are 'explainable': If the output of a linear map $h(x) = w^T x$ is large and positive, we can try to figure out which variables had a big input and look at their signs. Neural networks on the other hand do not enjoy this benefit and are still considered a 'black box' for most purposes. We can admire their superhuman average performance on many tasks, but in certain settings, we want to know *why* the network makes a certain prediction: Receiving a medical diagnosis, we may ask how the model or doctor came to its conclusion. Any model mostly considers statistical correlation, but at least in simple ones, we can understand which factors play a role.

It is comparatively easy to train neural networks to perform well on average (they are 'L^2-close' to the target function), but very hard to make them perform well in the worst case ('L^∞-close') in the sense that we hit the curse of dimensionality here. For some applications, 'good on average' is sufficient: More cannot be expected in financial investing, for instance. There is no strategy that *always* succeeds. But the possibly catastrophical failure of ML models can be terrifying in settings with potentially life-altering implications such as self-driving cars, medical diagnostics or court rooms.

Naturally, humans err. Human drivers are certainly not perfect, and medical doctors are not nearly as reliable as we want them to be. Like machine models, humans learn from examples and in most situations merely reproduce behavior they previously observed. In their favor, we can often understand how and why humans fail in specific ways. The issue with machine learning are its unpredictable failures: Nobody wants to be in a car that suddenly stops on the highway for a reason that no human can understand.

As of yet, even various industries has been skeptical of integrating AI. Models are not sufficiently reliable for widespread deployment in all areas, and it is unclear whether or when they will overcome these issues, as we are hitting the limits of available data. Even when they could be used successfully, the high cost of expertise, materials and energy does not necessarily allow for competitive pricing. The article [DV24] covers the difficulties facing the AI industry in greater detail.

4.6. Adversarial examples, adversarial stability and vulnerability

The issue of explainability worsens if we include problems of deliberate manipulation of data. Adversarial examples were first popularized in [SZS+13], where the authors showed that it is possible to perturb an image in a way that is imperceptible to humans, but which fools a neural network into thinking an image of a panda shows an airliner.

The issue is rooted in the fact that realistic data lies on a moderately low-dimensional subset of a much higher-dimensional space. Assume, for simplicity, that all data points lie on a d-dimensional subspace V of $\mathbb{R}^D$, for instance $V = \{x \in \mathbb{R}^D : x^{d+1} = \cdots = x^D = 0\}$. Here, the x^j refer to the coordinate entries, not to be confused with the data points x_i. Clearly, the final $D-d$ variables of the data should not contribute to the label y which a learned model $h(w,\cdot)$ assigns to a data point x, since the data give us no information about them. But, such an independence may not be learned: When we find w such that $h(w,x_i) = y_i$ for all data pairs (x_i, y_i), in general, we will find that the last $D-d$ entries of $\nabla_x h(w,x)$ are non-zero! What we find there depends on the training algorithm, but in general, $\partial_{x^j} h(w,x)$ will be a random quantity of with expectation zero and positive variance. Now, if D is much larger than d, then the $D-d$ entries of even a small size constitute a sizeable contribution to the vector $\nabla_x h$. When we are given a 'realistic' data point x – i.e., $x^{d+1} = \cdots = x^D = 0$ – this does not matter. But: we can add a small perturbation in these final coordinates which achieve a large change in

$$h(w,\tilde{x}) \approx h(w,x) + \nabla_x h(w,x) \cdot (\tilde{x} - x).$$

Thus, by moving x away from the set of sensible data even a little to $\tilde{x}$, we can cause the model to behave quite differently. The main ingredient here is the high dimension of the ambient space.

Of course, there are limitations: The setting above is heavily idealized, and there are models which would zero out the final variables in the gradient. Additionally, to cause the model to fail, we need to change x to a point $\tilde{x}$ such that the dot product between the gradient $\nabla_x h(w,x)$ and $\tilde{x}-x$ is large, which means that we need to have an idea about the inner workings of the model. If we were to choose $\tilde{x} = x + \varepsilon N$ randomly with, say, a standard Gaussian variable N, then the probability of selecting such a good direction is vanishingly small if D is large.

Still, when the 'surface' on which real data lives becomes more complicated (for instance, a manifold rather than a linear space) and a bit of stochastic noise is involved, algorithms may not pick up directly on the low-dimensional intrinsic structure. Whenever this implicit dimension reduction fails, we are vulnerable to adversarial attacks. And if our model is not interpretable, if we have no good understanding on which basis it assigns labels, we may not realize when something is off.

4.7. What does the ML community say?

While the machine learning community is well aware of its societal impact and its challenges, its response has been far from uniform. Where prominent researchers like Ali Rahimi and Geoffrey Hinton (one of the 'godfathers of AI', who left Google due to concerns about AI) have urged caution, others like Yann LeCun (another 'godfather' and the Chief AI Scientist at Meta) have been more optimistic.

Community organizers currently appear to veer towards the latter camp. For the first time, NeurIPS 2024 included a 'high school track' where students could submit essays that 'highlight either demonstrated positive social impact or the potential for positive social impact using machine learning.' Critical contributions were not solicited.

In view of the many ethical complications, the third and final 'godfather' Yoshua Bengio has called for ethics training for computer scientists and government regulation, akin to the controls put in place for other sensitive industries such as planes and pharmaceuticals.

4.8. The role of mathematics

Many older models in machine learning became popular after or in tandem with their theoretical analysis. By contrast, deep learning has primarily been driven by its practical successes rather than grown out of a theoretical understanding. This has made it hard to chart its limitations and guarantee its performance, leading to 'alchemical' practices which are only more recently giving way to rigorous ideas and scaling laws [KMH+20]. It has been described as an art rather than a science. Naturally, in art, there are no fundamental truths, and in their absence, opinions and debates flourish.

This is the role of the applied mathematician: To find fundamental guarantees and fundamental barriers that cannot be overcome, to establish the power and limitation of models. A proof cannot be argued with.

That is not to say that mathematical theory does not contribute to better models: Two recent classes of generative AI models, Wasserstein-GANs and more recently diffusion models (the technology underlying for instance the image generators DALL-E and Stable Diffusion), build on rather complex mathematical foundations. Undeniably though, there has been a gap between theory and practice, and, in the words of Yann LeCun: 'When empirical evidence clashes with a theory, it is the theory that is wrong (or misinterpreted), not the universe.'

4.9. Go forth and learn

It may sound like I believe that the negatives of machine learning outweigh its benefits. That is not the case. I desperately want a self-driving car, and despite the challenges, I can see the enormous promise of machine learning. In fact, I highly encourage you to study machine learning: On the one hand, it is as fascinating and vibrant a field of research as you can find today. More importantly, though, I believe literacy in statistics and data-driven sciences is becoming crucial to the functioning of our society, and the more we all understand about the mechanisms of machine learning, the better.

Artificial Intelligence is a wonderfully powerful tool, and it would be foolish to forsake it entirely. But we best remember that the intelligence we create may well be less akin to the intelligence of the human mind and more to the inscrutable mind of Lovecraft's eldritch god Cthulhu.

5. How do I get started?

Take a class! Many departments, including the Department of Mathematics at Pitt where I work, offer classes regularly or on a special topics basis.

My own point of entry into machine learning was the overview article [HH19], which gives a nice basic introduction to neural networks specifically for mathematicians. There is the excellent textbook [SSBD14], available for free as a pdf from Shai Shalev-Schwartz's website. It predates the deep learning boom however, and neural networks are not covered in great detail. Other theoretical textbooks such as [PZ24] are appearing (preprint available on arXiv) and books with a practical focus such as [LBH15, BB23] are excellent sources for practical intuition and a broader overview.

As I tried to demonstrate, there are no 'wrong' classes to take: Even classes which at first glance have very little connection to machine learning may prove useful in the end. However, some classes are very directly useful and should not be missed: The foundations of statistics, probability theory, linear algebra, optimization and numerical mathematics are ubiquitous in machine learning, and they may be the most directly useful. Similarly, without coding skills, machine learning is much like swimming on dry land. The predominant languages in the ML community are Python and, to a much lesser extent, R and Matlab. If you want to develop efficient libraries for deep learning, a language with finer control such as C or C++ might be the way to go. The large deep learning libraries PyTorch and TensorFlow are implemented at this 'lower level', but with a Python interface for easier access.

For a deeper understanding, there is very little that compares to a course in real analysis, but this comes with an asterisk: Real Analysis is a class which teaches the foundations of a way of thinking by examining the simplest case of a function in one variable (at least for the first semester). Naturally, functions in one variable are often not the most relevant case for applications. Like when learning a new language, this foundation is not useful unless you build on it later. The earlier you take real analysis, the more benefit you can get from the way of thinking it introduces you to.

Finally: Reach out about an undergraduate (or graduate) research project! Classes are great, but by necessity they tend to view a field in isolation. In practice, we rarely have the benefit of focusing on a single area of knowledge, but we need to combine our knowledge of statistics, our coding skills, our understanding of optimization and numerical analysis... And the best way to obtain hands-on experience is a research project or an internship. We need to be generalists, and we never know what we need to know until we are faced with a specific problem that defies easy classification.

References

[BB23] Christopher M Bishop and Hugh Bishop. *Deep learning: Foundations and concepts*. Springer Nature, 2023.

[CCTS20] Jeff Calder, Brendan Cook, Matthew Thorpe, and Dejan Slepčev. Poisson learning: Graph based semi-supervised learning at very low label rates. In *International Conference on Machine Learning*, pages 1306–1316. PMLR, 2020.

[CLW23] Hongrui Chen, Jihao Long, and Lei Wu. The L^∞-learnability of reproducing kernel Hilbert spaces. *CoRR*, 2023.

[DV24] Gerrit De Vynck. The AI hype bubble is deflating. now comes the hard part. *Washington Post*, 25 Apr 2024.

[EKW+24] Bora Erden, Malika Khurana, Ashley Wu, Kalina Borkiewicz, and Kellen Browning. Despite Trump's claims, footage shows large crowd at Harris's Detroit rally. *New York Times*, 12 Aug 2024.

[Erd23] Delger Erdenesanaa. A.I. could soon need as much electricity as an entire country. *New York Times*, 10 Oct 2023.

[Fri24] Danielle Friedman. A brief history of sexism in medicine. *New York Times*, 26 Feb 2024.

[Gel24] David Gelles. A.I.'s insatiable appetite for energy. *New York Times*, 11 July 2024.

[Gre22] Nicole Greenfield. Lithium mining is leaving Chile's indigenous communities high and dry (literally). *Natural Resources Defense Council*, 26 Apr 2022.

[HH19] Catherine F Higham and Desmond J Higham. Deep learning: An introduction for applied mathematicians. *Siam review*, 61(4):860–891, 2019.

[Hil20] Kashmir Hill. A face search engine anyone can use is alarmingly accurate. *New York Times*, 8 Jan 2020.

[Hil22] Kashmir Hill. The secretive company that might end privacy as we know it. *New York Times*, 26 May 2022.

[HK22] Kashmir Hill and Corey Kilgannon. Madison Square Garden uses facial recognition to ban its owner's enemies. *New York Times*, 22 Dec 2022.

[Kes15] David Kestenbaum. Keynes predicted we would be working 15-hour weeks. why was he so wrong? *NPR*, 13 Aug 2015.

[Key30] John Maynard Keynes. Economic possibilities for our grandchildren. In *Essays in persuasion*, pages 321–332. Springer, 1930.

[KMH+20] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[Kov19] Margaret Bull Kovera. Racial disparities in the criminal justice system: Prevalence, causes, and a search for solutions. *Journal of Social Issues*, 75(4):1139–1164, 2019.

[Kra24] Coralie Kraft. Trolls used her face to make fake porn. There was nothing she could do. *New York Times Magazine*, 31 July 2024.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[McQ23] Sarah McQuate. Q&A: UW researcher discusses just how much energy ChatGPT uses. *University of Washington News*, 27 July 2023.

[MKF+24] Cade Metz, Cecilia Kang, Sheera Frenkel, Stuart A. Thomson, and Nico Grant. How tech giants cut corners to harvest data for A.I. *New York Times*, 8 April 2024.

[New24] Newsroom. ChatGPT consumes 25 times more energy than Google. *The Brussels Times*, 12 May 2024.

[Pet85] Joan Petersilia. Racial disparities in the criminal justice system: A summary. *Crime & Delinquency*, 31(1):15–34, 1985.

[PZ24] Philipp Petersen and Jakob Zech. Mathematical theory of deep learning. *arXiv preprint arXiv:2407.18384*, 2024.

[Sac24] Goldman Sachs. AI is poised to drive 160% increase in data center power demand, 14 May 2024.

[SH23] Choe Sang-Hun. Deadly fire exposes harsh conditions migrant workers face in South Korea. *New York Times*, 25 June 2023.

[SMW+24] Andrew Ross Sorkin, Ravi Mattu, Bernhard Warner, Sarah Kessler, Michael J de la Merced, Lauren Hirsch, and Ephrat Livni. How bad is A.I. for the climate? *New York Times*, 6 May 2024.

[SSBD14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[STF21] Gleb Stolyarov and Gabrielle Tétrault-Farber. “face control": Russian police go digital against protesters. *Reuters*, 11 Feb 2021.

[SZS+13] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[Tan24] Freddy Tang. Discrete Laplacian on graphs, Dirichlet problem, and minimization of energy. *Pittsburgh Interdisciplinary Mathematics Review*, 1:47–65, Apr. 2024.

[Tho24] Stuart A Thompson. How 'deepfake Elon Musk' became the internet's biggest scammer. *New York Times*, 14 Aug 2024.

[Vig24] Neil Vigdor. Trump promotes A.I. images to falsely suggest Taylor Swift endorsed him. *New York Times*, 19 Aug 2024.

[Wik24] Wikipedia. Deep Fake Love, 2024.

Department of Mathematics, University of Pittsburgh, Pittsburgh, PA 15260
E-mail address: s.woj@pitt.edu