What are the limitations of derivative-based models for optimization in machine learning?

Faris Chaudhry

September 15, 2021

Abstract

Most machine learning problems can be transposed into optimization problems where the goal is finding the global minima or maxima to minimize loss or maximize potential. Each of the main learning methodologies (namely supervised, unsupervised and reinforcement) along with the models to represent and solve optimization problems have limitations - computationally and theoretically - that have to be identified and mitigated against to create an effective model. The focus here is continuously differentiable objective functions and thus derivative-based solutions are used.

Contents

Chapter 1

Introduction to Machine Learning and Optimization

1.1 What is Machine Learning?

Machine learning (ML) is a subfield of artificial intelligence (AI) and, broadly speaking, is the use of computational methods and models to improve performance and predictions through experience [3, p. 1]. Unlike humans, this learning is based entirely on data and statistics with experience being gained through interaction with a data set or an environment of some kind.

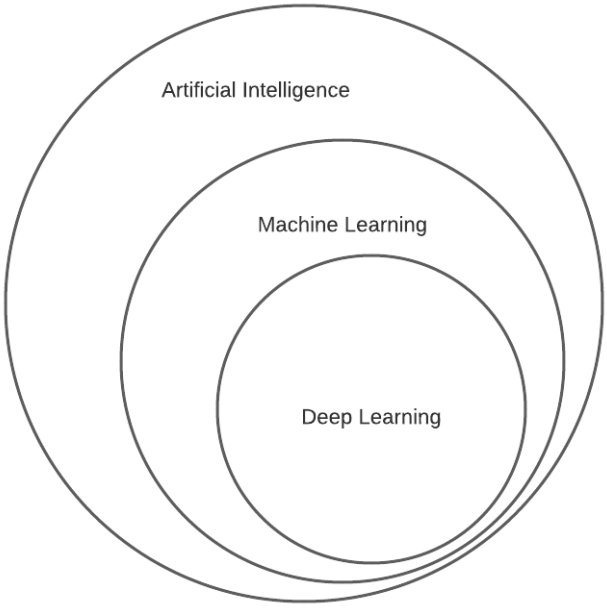

Figure 1.1: subfields of AI

There are 3 primary categories of learning philosophies (supervised, unsupervised and reinforcement) for ML models, with other hybrid methods being combinations of these. Each type of learning lends itself to certain types of problems. Supervised learning is used for classifying images and extrapolating data. Unsupervised learning takes raw data and finds patterns such as the overall distribution or groups with similar attributes. Reinforcement is used in complex systems with many changing variables that would be computationally difficult to solve otherwise, like chess.

1.2 Prerequisite Conditions for Derivative-Based Optimization

Optimization revolves around minimizing the loss or maximizing the value of a function. In the context of ML, optimization is vital to ensure modelling produces the greatest accuracy. The goal is to optimize an objective function, which is the representation of the variables being simulated. The solution to the objective function will be either a minimum (minima) or maximum (maxima) point (collectively called the set of extrema) as this is when the value of a function is highest or lowest.

The derivative is the linear approximation (tangent) to a function at a point. Suppose there was a function $f(x)$ then, intuitively, the derivative with respect to x would be how much the value of $f(x)$ changed with a small nudge in the x-direction. It is important to note that, by Fermat's theorem on stationary points [13], all critical points (extrema and saddle points) have first derivative equal to 0. Visually, this is because the tangent to any turning point will have a gradient of 0. See the function $y = x^2$ (fig. 1.2) which has a minima at $(0, 0)$.

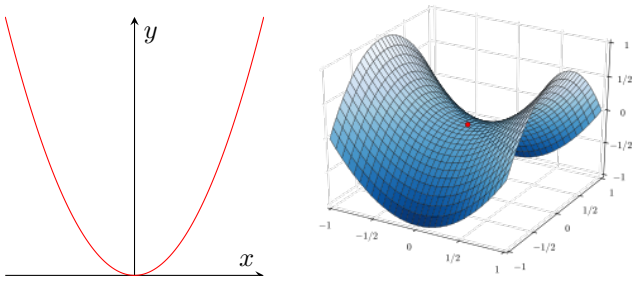

Figure 1.2: graph of x^2 (left) and an example of a saddle point (right)

So the objective becomes to find the location and nature - whether it's a minima, maxima or saddle point - of each critical point. A saddle point, such as the one pictured above, is defined as being a minimum point in one direction while being a maximum in other. This gives it a derivative of 0 but it isn't a solution to the optimization problem.

1.2.1 Continuity and Differentiability

The most essential requirement to using derivative-based methods will be that the objective function is continuous and twice-differentiable (the derivative of the function must also be differentiable) over the interval that contains the solution.

This is because to find the location and nature of critical points, the first and second derivative of a function are required [12].

For a function $f(x)$ to be continuous over the interval $I = [a, b]$

$$\forall k \in I, \lim_{x \to k} f(x) = f(k) \tag{1}$$

This means that, given any number in the interval, as x approaches that number it would be equal to putting the number into the function. This prevents any discontinuity since the limit wouldn't exist at discontinuous points. In fig. 1.3, $\lim_{x \to 0^+} = +\infty$ and $\lim_{x \to 0^-} = -\infty$. These values contradict meaning the limit isn't defined.

For a single-valued function, $f(x)$, the derivative, $f'(x)$, exists only when the following limits exists.

$$\frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x} \tag{2}$$

However, most objective functions will be multi-valued, to account for all the range of variables that affect the output, so this definition must be extended. This is the same principle but restricted to a specific direction. Suppose there is a function $f(x_1, \cdots, x_i)$ then the derivative with respect to a certain variable, x_n, will be.

$$\frac{\partial f}{\partial x_n} = \lim_{x \to 0} \frac{f(x_1, \cdots, x_n + \Delta x, \cdots, x_i) - f(x_1, \cdots, x_n, \cdots, x_i)}{\Delta x} \tag{3}$$

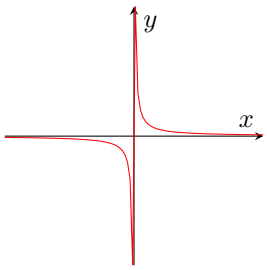

Figure 1.3: graph of $\frac{1}{x}$

In practice these rigorous definitions are not used, but the concept of continuity and differentiability is important.

- The first and second derivative must exist for an objective function to be solvable in this method, which is the major limiting factor. Although derivative-free methods do exist, they tend to be approximations of the exact values and heuristic in theory.
- Although many functions have discontinuities, like asymptotes or singularities, many times these are removable either by defining an interval without them or assigning an arbitrary value at a point for continuity.

1.2.2 Concavity and Convexity

When a function has only 1 minima or maxima over an interval it becomes much easier to find the global minimum or maximum since there is no need to check which point is a local extremum and which is the global extremum. Functions like these are called convex and concave where convex functions have a minimum point and concave functions have a maximum point. A convex function [8] can visually be described as having all its points below a line segment drawn between any 2 points (fig. 1.4) while a concave function has all points above.

It is important to note that concavity and convexity are not opposites. A function can be concave, non-concave, convex or non-convex. In addition, reflecting a function in the x-axis will reverse its concavity or convexity. Suppose $f(x)$ is convex then $-f(x)$ is concave.

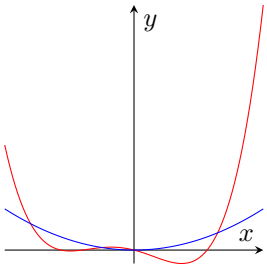

Figure 1.4: convex (blue) and non-convex (red)

If the objective function is a non-convex or non-concave function this doesn't prevent the use of derivative-based optimization. However it does restrict the range of methods that can be used to find the global solution. For example, iterative methods to find extrema might converge to a local extrema while neglecting other possible values. Moreover, it will increase the complexity of the problem computationally since there will be a range of possible global extrema that have to be checked - which can be particularly difficult when certain derivative tests are inconclusive.

Chapter 2

Supervised Learning

2.1 Application of Supervised Learning

The philosophy of supervised learning is to use labelled training data to map between an input vector and a target vector. In this case, the model is given data with input variables and the correct associated target values corresponding with them [1, p. 105]. Effectively, the model is creating a pattern out of which inputs cause certain outputs so that, given new inputs, the correct outputs can be predicted.

Supervised learning problems are split into 2 categories: classification problems and regression problems.

2.1.1 Classification

Classification problems are about predicting the class labels of an object. A common example of classification is assigning a digit label to a handwritten digit. However, these objects could be anything that can be labelled like sentences or sounds.

Let x_n be a feature/parameter of the object and l_n be a label.
Then a general classification function can be described as the mapping:

$$[x_1, x_2, \cdots] \mapsto [l_1, l_2, \cdots]$$

Given a particular vector of features the goal is to assign a set of class labels.

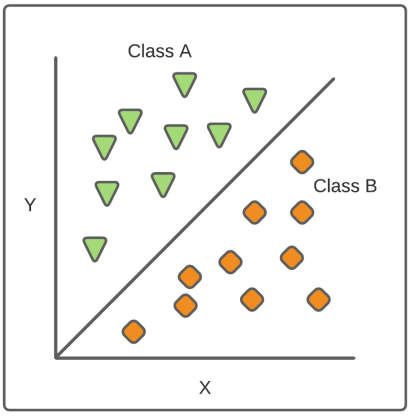

Figure 2.1: example of classification

2.1.2 Regression

Regression problems involve predicting a numerical value from the feature vector of an object. For example, given many variables about a stock (past history), predict the future value of the stock. Regression is about creating a mapping function from $\mathbb{R}^n \mapsto \mathbb{R}$.

Let x_n be a feature/parameter of the object and k be the numerical value associated with it.
Then a general regression function can be described as the mapping:

$$[x_1, x_2, \cdots] \mapsto k$$

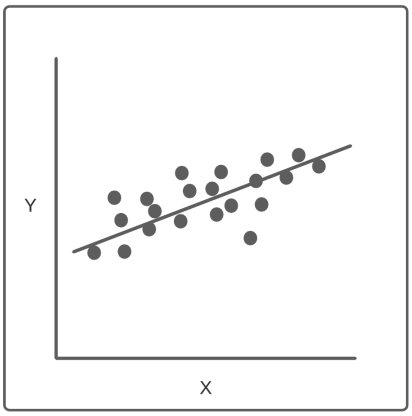

Figure 2.2: example of regression

2.2 General Optimization of Supervised Learning Problems

The optimization of supervised learning problems requires minimizing the average value of the loss function using the training samples. This produces the most accurate approximation to the underlying function to extrapolate values.

The general equation [7, p. 3] for this can be written as:

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L(y^i, f(x^i, \theta)) \tag{4}$$

where N is the number of training samples, θ is the parameter of the mapping function, x^i is a feature vector and y^i is the array of labels associated with that feature vector. L is a specific type of objective function, called a loss function, which is, by definition, meant to be minimized and f is the mapping function that takes the feature vector to the label vector.

The problem with using training samples is that the resulting function might be over-fitted to the given data. This would mean that, although the model is accurate for the training data it has been given, accuracy is reduced on new objects. A method to deal with this is through a regularization item, λ:

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L(y^i, f(x^i, \theta)) + \lambda \|\theta\|_2^2 \tag{5}$$

Regularization fundamentally discourages learning complex models [15, p. 4]. The output will be affected by multiple features and as the number of features increases, the model becomes more complicated. Regularization will selectively favour the most significant features while minimizing the importance of features which have little effect on the final output.

2.3 Limitations of Supervised Learning

Models don't generalize well from observed, training data to unseen data [15, p. 1-3]. The accuracy might be near perfect on training data while being poor on unseen data, somewhat mimicking the model memorizing the training data without grasping the mapping function. Many of the reasons for this can be attributed to the data set.

Firstly is the problem of choosing a suitably sized dataset. Learning using only a small number of samples limits the accuracy since there is not enough data to realize any meaningful connections. A large amount of training with labelled data, while increasing the training accuracy, will decrease the overall accuracy of the model when validating unseen data (validation accuracy). This is because of diminishing returns - resulting from the learning speed slowing down - causing memorization instead of learning. Consequently, one of the considerations is to have the right size data set to alleviate the risk of under-fitting or over-fitting. An easy way to do this is to train the model in batches and, after each epoch, check the validation accuracy until the error rate starts to increase.

The quality of the data heavily influences how effective the transition between training and validation will be. For example, using skew data (in the sense that the training samples don't represent the whole domain of input variables) will lead to a model that is accurate given niche combinations of variables and has to extrapolate on others. There is also some unpredictable instability in the data called noise. Reducing noise is key in ensuring an accurate model and requires the data set to be pruned so that there remains mostly high-quality accurate data. The problem with this is that a noise-free training set will be more inaccurate than a data set with some noise since the finished model will be processing unseen data that will have inherent noise.

Finally, there has to be a realistic bound to the model's complexity. As more inputs are added, the model becomes more accurate on average but has lower consistency and will require more training data. The culmination of all these factors is that training data requires great resources to produce. Without considering where the unlabelled data is coming from, human intervention will be required to label the data which means that each label has to be calculated for potentially thousands or millions of training samples. Apart from the amount of time this takes, human error can be a great detriment to the quality of the data (measurement errors and mislabelling). Most of the time, producing a good model is limited by how much good data you can obtain.

Chapter 3

Unsupervised Learning

3.1 Application of Unsupervised Learning

Unsupervised learning is different to supervised learning in the way that it uses unlabelled data [1, p. 105]; instead of learning from a mapping of inputs to a known output, the model is given only the inputs to learn from and has to make sense of the data without guidance. As a result of this, unsupervised learning revolves around extracting relationships from the data without the inherent human biases caused by choosing the correct output beforehand.

Unsupervised learning problems strive to solve 2 problems: finding clusters of similar data and summarizing the underlying distribution of the data (density estimation).

3.1.1 Clustering

Unlike classification, where the classes are predefined, clustering requires the model to define its own cluster of data based on the similarities of the features.

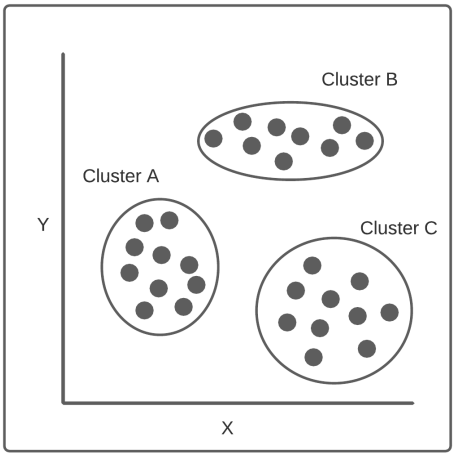

Figure 3.1: example of clustering

Optimization will involve making the variance of each cluster as small as possible (variance will be equivalent to the distance from the centre of the cluster). This can be done by iterating through each cluster and checking the distance from the centre of the cluster for each sample.

$$\min_{s} \sum_{k=1}^{K} \sum_{x \in S_k} \|x - \mu_k\|_2^2 \tag{6}$$

where s is the variance, K is the number of clusters, S_k is the set of samples for that cluster, μ_k is the centre of a cluster [7, p. 3-4].

3.1.2 Density Estimation

The assumption is that there exists some probability distribution to describe the relationship between the variables [5]. Density estimation is a useful asset in modelling to estimate the properties of a given data set (variance, skewness, type of distribution).

Suppose there exists a set of continuous random variables, $(x_1, \cdots, x_n)$, then there is a probability distribution that the set models, $P(x_1, \cdots, x_n)$. The goal is to find a probability density function (PDF) that can describe the mapping: $(x_1, \cdots, x_n) \rightarrow P(x_1, \cdots, x_n)$. The assumption is that this will PDF will be continuous since our objective function has the precondition of being continuous.

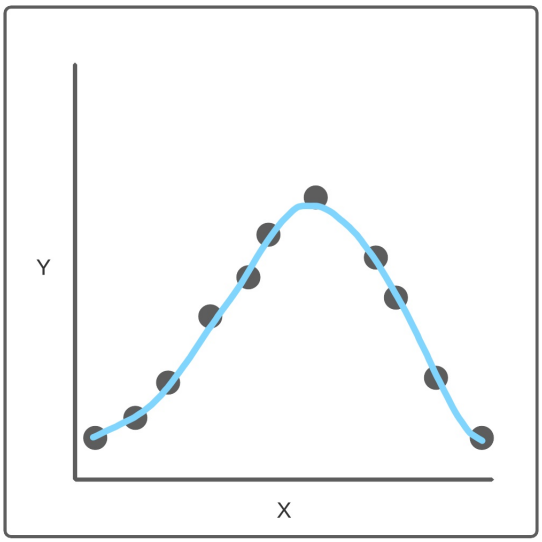

Figure 3.2: example of density estimation

Getting the exact function for the data is not only unlikely but also suboptimal. As more data is used, the approximation will get more complex and usually closer to the actual value however too much data increases the risk of over-fitting to the data, especially if there is abnormal data. Instead, the focus is to maximize the likelihood that a predicted PDF is correct.

In statistics, the likelihood function measure the goodness of fit between the data and the proposed PDF, so maximizing it finds the PDF which has the highest probability of fitting the data. A likelihood function is written as $p(x^i, \theta)$ where x^i is a data point and θ is a PDF. However, many data sets have large numeric ranges that can limit the effectiveness of this function so it is easier to work with the logarithmic likelihood function, $\ln p(x^i, \theta)$ or $\ell(x^i, \theta)$.

It is important to note that

$$\max p(x^i, \theta) = \max \ell(x^i, \theta) = \widehat{\ell}(x^i, \widehat{\theta}) \tag{7}$$

Since ln is strictly increasing, maximizing the logarithmic likelihood function also maximizes the likelihood function. Thus general density estimation [7, p. 4] is:

$$\max \frac{1}{N} \sum_{i=1}^{N} \ell(x^i, \theta) \tag{8}$$

where N is the number of training samples and x^i is a particular feature vector of a sample.

3.2 Limitations of Unsupervised Learning

The quality of data is, just like in supervised learning, pivotal to the ability for the model to function. Although data doesn't need to be pre-processed and labelled in the same manner as supervised learning, unsupervised learning still has the shortfall of requiring data that is representative of the whole data set to prevent skew results. Furthermore, the data needs to have a suitable resolution for clustering. If the data has a small resolution then segments that represent separate clusters could be grouped. Consider data which represents clusters '$2-3$', '$4-5$' and '$95-100$' units. There is a possibility that the first 2 clusters are grouped as 1 cluster. Problems like these could be resolved by using a recursive method to analyse the clusters of each cluster but in the end the process will always be time-consuming.

The nature of unsupervised learning - being the use of raw data - will lead to apprehension as to the usefulness and accuracy of the results. Without human intervention the practicality of the outputs will be greatly dependant on the model being used. Accuracy is generally lower than other learning options seeing that the model has to establish its own connections based on features in similarity with no notion as to whether they are correct. It is due to this fact that the relationships extracted have with no application.

Finally, unsupervised tasks will require a far greater amount of data than their supervised counterparts. Extracting worthwhile relationships with no way to predefine the output of the model will require much more data to not only lessen the effect that outliers and abnormalities would have on the prediction but also to compensate for the complexity, which scales exponentially with each input variable added.

It is as a consequence of these limitations that often unsupervised learning is less preferable to some hybrid models like semi-supervised learning which mixes labelled and unlabelled data to increase the chance of producing practical and relevant results.

Chapter 4

Reinforcement Learning

Reinforcement learning entails an agent interacting with an environment rather than using a traditional dataset [1, p. 105]. The learning process is done through trial and error to develop a feedback loop between the environment and the agent's experience. Each state of the environment should be mapped to the action that maximizes the reward.

This game theory approach assigns a payoff yield to each action that encourages or discourages certain actions. A by-product of this is to consider how the length of time can affect the payoff of an action. For example, maybe an action has a high reward when considering only the next state but when looking at the next 10 states its value is lower. This can be seen in chess when, as the depth of the AI get higher, certain moves become worse as they compromise future positions.

The dynamic nature of reinforcement learning is useful in complex systems where computationally working out the whole game would be inefficient if not impossible. Consider chess [4] where the number of possible games is estimated as 10^{123} (Shannon's number) and is too big to compute effectively after even 5 moves (a half-move is counted after either white or black moves rather than a full move which requires both players to move).

Number of half-moves	Number of Possible Games
1	20
2	400
3	8,902
4	197,281
5	4,865,609

Figure 4.1: the exponential growth of possible chess games

4.1 Optimization in Reinforcement Learning

A policy function, $\pi(s)$, maps a state, s, to an action, a, to select the best course of action from the set of all actions, A, that the agent can take in each situation from the set of possible situations, S.

$$\pi(s) : s \to a \text{ where } s \in S, a \in A$$

By maximizing the expected value of this function [7, p. 4], the agent will select the best action to be performed in each state, maximizing the payoff of the actions.

$$\max_{\pi(s)} = \mathbf{E}\left[\sum_{k=1}^{T} \gamma^k r_{t+k} | S_t = s\right] \tag{9}$$

where T is the time horizon, γ is the discount factor, r is the reward function with respect to the turn and the time into the future being considered, S_t is a given state.

If the game was infinite - or has no predetermined stopping point - then a simple solution is to work out the $\lim_{T\to\infty}$ and truncate the series at some point to get a good approximation. Using this method of truncation would allow the depth to be changed (how far into the future turns are considered).

The discount factor, γ, is a value that prioritizes instant reward over a future reward [6]. If there is a constant risk which may cause failure to realize the reward then that future reward should have a decreased payoff to compensate for the risk. For example, suppose there is a 50% chance (implying $\gamma = 0.5$) that the game ends after every turn and you could choose either a payoff of 1 unit after 1 turn or a payoff of 10 units after 5 turns. Then the expected value of option 1 is $0.5^1 \cdot 1$ which is 0.5 and option 2 is $0.5^5 \cdot 10$ which is 0.3125 so option 1 is statistically better.

4.2 Limitations of Reinforcement Learning

A noisy environment will cause a sharp decrease in learning speed. If feedback is delayed (like the effect of an action doesn't happen until a few time intervals after) then differentiating cause and effect can be difficult. Unlike data sets, where the noise can be artificially removed, there is no possibility of removing this noise since then the environment would be different and the agent would be acting on an unrelated environment. It is because of this that the efficacy of the model will fluctuate as new game theory is being developed. Whenever new strategies are tried, the performance might decrease in the short term however the trial-and-error process will encourage a greater depth of strategy later.

Many of the parameters of an environment will have to be manually set. For example, the correct value for the discount factor is usually obvious - as close to 1 as possible for continuos environments and otherwise the risk every time step - but realistically there has to be some focus on short term rewards as an extremely long wait is not conducive to real life. This can be seen in chess where three-fold repetition and the 50 move rule (no captures or pawn moves in 50 turns) will result in a draw. In this case, there is a preset condition that prevents the best moves being played if the time scale is too large.

The subjective nature of payoffs is also pertinent to the development of a simulation. As systems get more complex it becomes more difficult to ascribe values to the actions that can be taken. Either these values will follow some subjective set of rules, in which case they are still influenced by some bias, or they must be chosen subjectively thus human bias creates preferred ways to operate leading to a sub-optimal set of actions.

One way to minimize human bias and thus reach the optimum solutions is to simplify the outcomes. For instance, in chess, the set of outcomes and payoffs can be reduced to a positive payoff for a win and a negative payoff for a loss without considering the possible decisions to get there. This methodology, when paired with deep learning, will lead to the computer playing many games with itself to work out the relative payoffs and tweak them over time as new strategies eclipse old ones. Over millions of games and multiple generations of neural networks, the result will be a model with the optimum policy function that has as little human bias as possible and therefore can express actions that would naturally be unorthodox or creative. The downside of this is the resources and time required to reach this transcendental state in a reasonable time frame.

Chapter 5

Mathematical Models for Optimization

5.1 Constraints on the Objective Function

Constraints are conditions that the solutions to the objective function must satisfy. In many cases, there are certain restrictions that should be added due to computational and resource limitations.

There are 3 types of constraints that must be considered:

- inequality constraints such as $x \geq k$.
- equality constraints such as $x = k$
- data type constraints such as x is an integer ($x \in \mathbb{Z}$)

Data type constraints are normally easy to deal with; changing with values the model checks or adding a conditional statement can be enough in many cases.

Equality constraints can be appended onto the objective function using the Lagrange multiplier [11]. Let $f(x)$ be an objective function and $c_n(x) = k_n$ be an equality constraint. Then the general Lagrange multiplier for n constraints would be:

$$\mathcal{L}(x) = f(x) - \lambda_1(c_1(x) - k_1) - \cdots - \lambda_n(c_n(x) - k_n) \tag{10}$$

The solution to the original objective function, $f(x)$ will be a saddle point on the Lagrange multiplier. One problem with this approach is that, firstly, for each constraint added there will be an extra partial derivative that needs to be computed and thus a more complex simultaneous equation. In addition, inequality constraints ($c(x) \geq k$) are not supported through this method. However, there is a generalization to the Lagrange multiplier called the Karush–Kuhn–Tucker (KKT) conditions.

5.2 Methods for Finding Extrema

5.2.1 Jacobian

Previously, it was ascertained that the locations of the critical points required the first derivative of the function to be 0 ($f' = 0$). For a single-valued function this is equivalent to solving

$$\frac{df}{dx} = 0$$

However for multi-valued functions a critical point will require all partial derivatives to be 0 at that point. The Jacobian [14] is a vector of the partial derivatives of the function that gives a vector of the

gradients at that point.

$$J_f = \nabla f = [\partial_{x_1} f, \cdots, \partial_{x_n} f] \tag{11}$$

This can be solved by either setting each partial derivative to 0 and solving it simultaneously or by finding the values when the value of the Jacobian is 0.

$$\partial_{x_1} f = \cdots = \partial_{x_n} f = 0 \text{ or } \|J\| = 0$$

Sometimes the resulting equations might be difficult to solve (polynomials of order greater than 4 for example) and brute force or numerical methods might need to be used.

5.2.2 Hessian

After finding the locations of the critical points, higher-order derivative tests must be used to determine the nature of the points. The Hessian [10] is like the 'Jacobian of the Jacobian' and is a representation of the change in the gradient.

$$H_f = \nabla(\nabla f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \tag{12}$$

Due to the commutative nature of second partial derivatives, computation time is significantly cut down as only the upper or lower triangular matrix has to be calculated and then mirrored.

$$\frac{\partial^2 f}{\partial y \partial x} = \frac{\partial^2 f}{\partial x \partial y} \tag{13}$$

5.2.3 Higher-Order Derivative Tests

Using the determinant and trace of the Hessian matrix where λ is an eigenvalue

$$trH = \sum_{i=1}^{n} H_{ii} = \sum_{\forall i} \lambda_i \text{ and } detH = \prod_{\forall i} \lambda_i \tag{14}$$

the second derivative test is as follows. Let $(x_1, x_2, \cdots, x_n)$ be a critical point substituted into the Hessian. Then [12]

- $detH > 0$ and $trH > 0 \implies$ local minimum.
- $detH > 0$ and $trH < 0 \implies$ local maximum.
- $detH < 0 \implies$ saddle point.
- $detH = 0 \implies$ inconclusive test.

Doing this for all critical points will determine which are extrema and comparing the values of the extrema will find the global minimum or maximum, which will be the solution to the optimization problem. In the case when the test is inconclusive, higher-order tests [9] can be used (although many derivatives would have to be calculated) or inspection.

For complex functions, visual inspection is usually impossible (since displaying something with greater than 3 dimensions is unintuitive) but numerical inspection can be useful, like adding dx_n for each variable and comparing the values. For example, if the neighbourhood of a point has values greater than the point it must be a minima.

The Hessian matrix is not without limitations; the computing time required to work out the Hessian scales greatly as each new variable is added and, for each critical point, the eigenvalues of the matrix will have to be worked out to find the trace and determinant. To mitigate this there are algorithms that can approximate the Hessian with the only condition that the matrix is invertible (the determinant is not 0).

5.2.4 Iterative Methods

When exact solutions are too difficult to calculate either due to too many variables or the Hessian matrix being to large to reasonably store, iterative methods can be used as numerical approximations. Since the approximation will converge to the answer as the number of iterations approaches infinity, many iterative methods are far superior, in practice, because of the lower resource cost relative to the accuracy of the approximation.

Newton's method for finding extrema [2] is one of the most simple of the iterative methods and demonstrates many downfalls of early iterative formulae.

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)} \tag{15}$$

or in the case of multi-valued functions

$$x_{k+1} = x_k - [H_f(x_k)]^{-1} J_f(x_k) \tag{16}$$

where H^{-1} is the inverse Hessian and J is the Jacobian. Considering the preconditions, this technique is still quite limited. Although we don't have to solve simultaneous equations or compute a derivative test, the Hessian has to be calculated and inverted (which requires it to be non-singular).

More importantly, this sequence will only converge to one value even if the function has multiple critical points. The initial guess, x_0, has a huge impact on where the sequence converges assuming it converges. In this sense, Newton's method has the property of local convergence where the sequence will converge when x_0 is close to a critical value.

A few other things to consider are that the convergence point might be a saddle point rather than an extrema and sometimes the sequence might not converge but rather cycle between values without settling on the extrema. Most of the problems stem from choosing a bad starting point or discontinuous derivatives.

Many of these issues are addressed in more modern algorithms (each with their limitations). To reduce the computational resources required, quasi-newton methods use update formulas to approximate the Hessian at every step, reducing the time and space complexity of each iteration. Many different update formulas are used situationally. On the other hand, to reduce the effect of bad starting points, stochastic methods (like stochastic gradient descent) choose starting points randomly while reducing convergence time to compensate.

The reality is that there are many ideologies behind iterative optimization that all preserve the same concept of finding a series converging on a solution.

List of Figures

References

[1] Yoshua Bengio, Ian Goodfellow, and Aaron Courville. *Deep learning*. Vol. 1. MIT press Massachusetts, USA: 2017.

[2] Roger Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2013.

[3] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[4] Claude E Shannon. "XXII. Programming a computer for playing chess". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 41.314 (1950), pp. 256–275.

[5] Simon J Sheather. "Density estimation". In: *Statistical science* (2004), pp. 588–597.

[6] Peter D Sozou. "On hyperbolic discounting and uncertain hazard rates". In: *Proceedings of the Royal Society of London. Series B: Biological Sciences* 265.1409 (1998), pp. 2015–2020.

[7] Shiliang Sun et al. "A survey of optimization methods from a machine learning perspective". In: *IEEE transactions on cybernetics* 50.8 (2019), pp. 3668–3681.

[8] Lieven Vandenberghe and Stephen Boyd. *Convex optimization*. Vol. 1. Cambridge University Press Cambridge, 2004.

[9] Eric W. Weisstein. *Extremum Test. From MathWorld–A Wolfram Web Resource*. URL: `https://mathworld.wolfram.com/ExtremumTest.html`.

[10] Eric W. Weisstein. *Jacobian. From MathWorld–A Wolfram Web Resource*. URL: `https://mathworld.wolfram.com/Hessian.html`.

[11] Eric W. Weisstein. *Lagrange Multiplier. From MathWorld–A Wolfram Web Resource*. URL: `https://mathworld.wolfram.com/LagrangeMultiplier.html`.

[12] Eric W. Weisstein. *Second Derivative Test. From MathWorld–A Wolfram Web Resource*. URL: `https://mathworld.wolfram.com/SecondDerivativeTest.html`.

[13] Eric W. Weisstein. *Stationary Point. From MathWorld–A Wolfram Web Resource*. URL: `https://mathworld.wolfram.com/StationaryPoint.html`.

[14] Eric W. Weisstein. *Stationary Point. From MathWorld–A Wolfram Web Resource*. URL: `https://mathworld.wolfram.com/Jacobian.html`.

[15] Xue Ying. "An overview of overfitting and its solutions". In: *Journal of Physics: Conference Series*. Vol. 1168. 2. IOP Publishing. 2019, p. 022022.