**Why Training a Neural Network Is Hard**

<https://machinelearningmastery.com/why-training-a-neural-network-is-hard/>

Fitting a neural network involves using a training dataset to update the model weights to create a good mapping of inputs to outputs.

This training process is solved using an optimization algorithm that searches through a space of possible values for the neural network model weights for a set of weights that results in good performance on the training dataset.

## Learning as Optimization

Deep learning neural network models learn to map inputs to outputs given a training dataset of examples.

The training process involves finding a set of weights in the network that proves to be good, or good enough, at solving the specific problem.

This training process is iterative, meaning that it progresses step by step with small updates to the model weights each iteration and, in turn, a change in the performance of the model each iteration.

The iterative training process of neural networks solves an optimization problem that finds for parameters (model weights) that result in a minimum error or loss when evaluating the examples in the training dataset.

## Challenging Optimization

Training deep learning neural networks is very challenging.

The best general algorithm known for solving this problem is stochastic gradient descent, where model weights are updated each iteration using the [backpropagation of error algorithm](https://machinelearningmastery.com/implement-backpropagation-algorithm-scratch-python/).

*Optimization in general is an extremely difficult task. […] When training neural networks, we must confront the general non-convex case.*

— Page 282, [Deep Learning](https://amzn.to/2rjgvLI), 2016.

An optimization process can be understood conceptually as a search through a landscape for a candidate solution that is sufficiently satisfactory.

A point on the landscape is a specific set of weights for the model, and the elevation of that point is an evaluation of the set of weights, where valleys represent good models with small values of loss.

This is a common conceptualization of optimization problems and the landscape is referred to as an “*error surface*.”

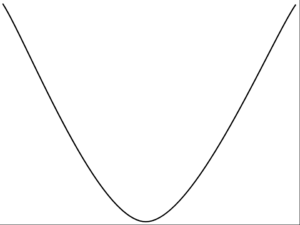
*In general, E(w) [the error function of the weights] is a multidimensional function and impossible to visualize. If it could be plotted as a function of w [the weights], however, E [the error function] might look like a landscape with hills and valleys …*

— Page 113, [Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks](https://amzn.to/2PgixWj), 1999.

The optimization algorithm iteratively steps across this landscape, updating the weights and seeking out good or low elevation areas.

For simple optimization problems, the shape of the landscape is a big bowl and finding the bottom is easy, so easy that very efficient algorithms can be designed to find the best solution.

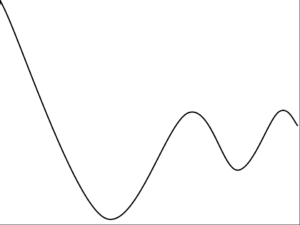
These types of optimization problems are referred to mathematically as convex.



Slika : convex error surface

The error surface we wish to navigate when optimizing the weights of a neural network is not a bowl shape. It is a landscape with many hills and valleys.

These type of optimization problems are referred to mathematically as non-convex.



Slika : non-convex error surface

In fact, there does not exist an algorithm to solve the problem of finding an optimal set of weights for a neural network in polynomial time.

Mathematically, the optimization problem solved by training a neural network is referred to as NP-complete (e.g. they are very hard to solve).