Training artificial neural network pdf

Artificial neural networks are relatively raw networks of neurons based on the neural structure of the brain. They process the records one by one and learn by comparing their classification with the item (which at the beginning is largely arbitrary) and known as the actual classification record. Errors resulting from the initial classification of the first item shall be redirected back to the network and used to modify the network algorithm for the second time and for many iterations. Roughly speaking, the neuron's artificial neural network has a set of input values (xi) and associated scales (wi)A function (g) that sums up the scales and maps results output (y). Neurons are organized layers. The input layer is not composed of full neurons, but rather consists simply of the values of the data record that make up the inputs of the next layer of neurons. The next layer is called a hidden layer; can be several hidden layers. The final layer is an output layer with one node for each class. A single sweep forward through the network results in a value being set to each output node, and the record is assigned to any class node with the highest value. The training phase of the artificial neural network is known to the correct class of each item (called guided training), and the output nodes can be assigned the correct values -- 1 for the node corresponding to the correct class and 0 for the others. (In practice, it has been found that it is better to use values of 0.9 and 0.1, respectively.) Therefore, it is possible to compare the calculated values of network output nodes with their correct values and to calculate the error word (Delta rule) for each node. These error terms are then used to adjust the weights of hidden layers, so hopefully next time the output values will be closer to the correct values. The main feature of neural networks Iterative learning process Is an iterative learning process where data cases (rows) are presented to the network individually and the weights associated with input values are adjusted each time. After all cases have been submitted, the process often starts again. During this phase of study, the network learns by adjusting the weight so that it is possible to predict the correct class mark of the input samples. The study of the neural network is also called connection learning, which is due to the connections between units. The advantages of neural networks include their high tolerance for noisy data, as well as their ability to classify patterns for which they have not been trained. The most popular neural network algorithm is the back-reproduction algorithm offered in the 1980s. If the network is designed for a specific application, it's ready to be training. To start this process, the initial scales (described in the following section) are selected randomly. Then the training or learning begins. The Commission has processes training data records one by one using the weight and functions of hidden layers, then compares the outputs obtained with the desired outputs. Then, the errors spread back through the system, causing the system customization of the weight of the application to process the next record. This process happens time and time again when the scales are constantly tweaked. During network training, the same data shall be processed several times when the community weight is continuously improved. Note that some networks never learn. This may be because input data does not contain specific information from which the desired output is derived. Networks are not concentrated even when there is not enough data to allow full learning. Ideally, there should be enough data to prevent some of the data as a validation set. Feedforward, Back-Propagation Next, back-propagating architecture was developed in the early 1970s from several independent sources (Werbor; Parker, don't you think Rumelhart, Hinton and Williams? This independent cooperation was due to the proliferation of articles and talks at various conferences that stimulated the whole industry. Currently, this synergistically developed back-propagation architecture is the most popular, efficient and easily learning model of complex, multilayered networks. Its greatest strength is in nonlinear solutions to poorly defined problems. A typical back-propagation network has an input layer, an output layer, and at least one hidden layer. There is no theoretical limit to the number of hidden layers, but there is usually only one or two. Some work has been done to show that a maximum of five layers (one input layer, three hidden layers, and an output layer) are required to solve problems. Each layer is completely connected to the next layer. As noted above, the training process typically uses some variants of the delta rule, which begins with a calculated difference between actual outputs and desired outputs. This error increases the connection scales in proportion to the global accuracy scaling factor with the error times. This means that inputs, output, and desired output must be in the same processing element. The complex part of this learning mechanism is that the system determines which input contributed most to the wrong output and how the error in that element is corrected. A passive node would not help with the error and would not need to change your weight. To resolve this issue, training inputs are applied to the network input layer, and the desired outputs are compared at the output layer. During the learning process, further network control is performed and the output of each element is calculated by layer. The difference between the output of the final layer and the desired output layer(s) normally altered by the derivative of the transmission function, and the connection scales are usually adjusted by a delta rule. This process will continue for the previous layer(s) until the input layer has reached the input layer. Network structuring Number of layers and number of processing elements per layer are important decisions. These parameters of feedforward, back-reproduction topology are also the most et promotion - they are an art network designer. There is no measurable, best response layout of the network to a specific application. There are only general rules picked up over time and followed by most scientists and engineers applying this architecture to their problems. Rule one: The complexity of the relationship between input data and the desired output should also increase the number of processing elements of the hidden layer. Rule two: If the process you are modeling is separated for several steps, additional hidden layers may be required. If the process is not separated by stages, additional layers may simply allow the training kit to be memorized, not a real general solution that is effective with other data. Rule number: The amount of training data available determines the limit of the number of processing elements of the hidden layer(s). To calculate this upper limit, use the cases in the training data set and share that number by the sum of the number of nodes in the network input and output layers. Then re-rate this result with a scaling factor between 5 and 10. The larger tag factors are used for relatively less noisy data. If you use too many artificial neurons training kit is remembered. If this happens, there will be no generalisation of the data, rendering the network useless in new data sets. Last updated August 6, 2019 or why stetohastic gradient descent is used to train neural networks. The installation of a neural network involves the use of a training data set to update the scales of the model to create good output input mapping. This training process is solved by using an optimization algorithm that searches through space for possible values in a neural network to model scales with a set of scales, resulting in a good performance training data set. In this post, you will discover the challenge of training a neural network framed optimization problem. After reading this post, you know: Training the neural network involves using an optimization algorithm to find a set of scales for the best map inputs for outputs. The problem is difficult, not only because the fault surface is non-kulev and contains local minins, flat spots and is very multidimensional. The stohastic gradient descent algorithm is the best overall algorithm to solve this complex problem. Kick-start your project with my new book Better Deep Learning, including step-by-step tutorials and Python source code files for all examples. Let's get started. Why Training The neural network is HardPhoto by Loren Kerns, some rights reserved. Overview This tutorial is divided into four parts; They are: Learning optimization complex optimization features Error Surface impact training learning optimization Deep learning neural network models learn to map inputs to outputs given in training data set examples. The training process involves finding a set of weights in a network that turns out to be good, or good enough to solve a specific problem. This training process is iterative, which means that it progresses step by step with small model updates for each iteration and, in turn, with a change in the performance of each iteration of the model. The iterative training process for neural networks solves an optimisation problem that finds a minimal error or loss for parameters (model scales) that result in minimal error or loss when evaluating examples of training dataset. Optimization is aimed at the search procedure and optimization problem that we want to solve when training the neural network model is very complex. This raises the question, what exactly is the problem of optimising this process? Take my free 7-day email crash course now (with sample code). Click to sign up and also get a free PDF ebook version of the course. Download your FREE Mini-Course challenges optimization Training for deep learning neural networks is very challenging. The best general algorithm known to solve this problem is a stofihastic gradient descent, where model scales are updated for each iteration using a reproduction error algorithm. Optimization in general is a very difficult task. [...] When training neural networks, we have to face a general non-grid case. — Page 282, Deep Learning, 2016. The optimization process can be conceptually understood as searching through the landscape to find a solution that is satisfactory enough. A dot on the landscape is a specific set of scales for the model and the height of this point is the assessment of a set of scales where the valleys represent good models with low loss values. It is common to understand optimization problems and the landscape cult is called fault space. In general, E(w) [error function scales] is a multidimensional function and impossible to visualize. If it could be drawn as a function w [scales], however, E [fault function] could seem like a landscape of mountains and valleys... — Page 113, Neural Smithing: Monitoring The Learning of Feedforward Artificial Neural Networks, 1999. The optimization algorithm takes over this landscape by updating scales and searching for good or low-altitude areas. With simple optimization problems, the shape of the landscape is a large bowl and finding the bottom is simple, so simple that highly efficient algorithms can be designed to find the best solution. These types of optimization problems are called mathematically convex. curved error Surface Error surface, which we want to navigate when optimizing the scales of the neural network is not in bowl shape. It's a landscape with many mountains and valleys. These types of optimization problems are called mathematically non-convex. An example of a non-convex error on the Surface Actually does not have an algorithm to solve the problem of finding the optimal set of weight neural network polynomials. Mathematically, the problem of optimisation solved by neural network training is called NP-complete (e.g. they are very difficult to solve). We prove this problem NP-complete and thus show that learning in neural networks is not an effective overall solution. — Design of the neural network and the complexity of learning, 1988. The main features of the Error Surface there are many types of non-convex optimization problems, but the specific type of problem we solve when training the neural network is particularly complex. We can characterize the difficulties in relation to the features of the landscape or the fault surface that the optimization algorithm may encounter and must navigate to be able to provide a good solution. There are many aspects of optimizing neural network scales that make the problem complex, but the three often mentioned features of the fault landscape are the presence of local minis, flat areas and a large-scale search space. Reproduction can be very slow, especially in multi-layered networks, where the cost area is usually non-rectangular, non-kulev and large-scale with many local minimums and/or flat areas. — Page 13, Neural Networks: Trade Tricks, 2012. 1. Local Minima Local minimum or local optima refer to the fact that the fault in the landscape includes several areas where the damage is relatively low. These are valleys where solutions in these valleys look good for the slopes and peaks around them. The problem is that, in a wider view, the valley has a relatively high height and may have better solutions. It's hard to know if the optimization algorithm is in the valley or not, so it's good practice to start the optimization process with a lot of noise that allows landscape samples to fall widely before choosing the valley to fall. By contrast, the lowest point in the landscape is called global mini-mini-. Neural networks can have one or more global mini-d.a. and the challenge is that the difference between local and global minimums may not make a difference. Its effect is that often finding a good enough set of weights is more applied and in turn more desirable than finding a global optimal or best set of weights. Nonlinear networks usually have several different depths to the local minimum. The aim of the training is to find one of these mini-dts. — Page 14, Neural Networks: Trade Tricks, 2012. A classic approach to solving the problem of local authorities is to restart the search process several times with another starting point (random initial scales) and allows the optimization algorithm to find different and hopefully better, local minis. This is called multiple restarts. Random restarts: One of the easiest ways to deal with local minimums is to train many different networks with different initial weights. — Page 121, Nerve Forging: Instructiond Learning in Artificial Neural Networks, 1999. 2. Flat areas (saddle points) Flat areas or saddle points are points in a landscape where the inclination is zero. These are flat areas between the peaks at the bottom of the valleys or regions. The problem is that zero gradient means that the optimization algorithm does not know which direction to move in order to improve the model. ... the presence of saddle points or regions where the fault function is very flat may result in some iterative algorithms stuck for a long time, thereby mimicking local mini-dbs. — Page 255, Neural Networks for the Recognition of Patterns, 1995. An example of a Saddlepoint error on the Surface, however, recent work may indicate that perhaps local minis and flat areas may be less challenging than previously believed. Do neural networks enter the local minimum and get out of them? Do they move at different speeds as they approach and then pass through different saddle points? [...] we present evidence with certainty, which suggests that the answer to all these questions is no. - Qualitatively characterised by problems with nervous network optimisation, 2015. High-Dimensional optimization problem solved when training neural network is large-scale. The weight of each network represents the second parameter or dimension of the fault surface. Deep neural networks often have millions of parameters, making the landscape navigate the algorithm on a very large scale, compared to more traditional machine learning algorithms. The problem with large-scale space navigation is that adding each new dimension greatly increases the distance between space points or hyperspace. This is often referred to as the curse dimensions. This phenomenon is called a curse. Of particular concern is that the number of possible different configurations in the set of variables increases exponentially as the number of variables increases. — Page 155, Deep Learning, 2016. The effects of training The complex nature of optimization problems to solve when using deep learning neural networks has an impact when training models in practice. In general, sttochastic gradient descent is the best algorithm available, and this algorithm does not provide guarantees. There is no formula to ensure that 1) the network approaches a good solution, 2) the approach is fast or (3) approach at all takes place. — Page 13, Neural Networks: Trade Tricks, 2012. We can summa rightly summa you as follows: Questionable solution quality. The optimization process may or may not find a good solution and solutions can only be compared relatively because of deceptive local minimums. Possibly a long training session. The optimization process can take a long time to find a satisfactory solution because of the iterative nature of the search. A possible failure. The optimization process may not progress (get stuck) or not find a viable solution because of the presence of flat areas. The task of effective training is to carefully configure, test, and set up the hyperparameters of the model and the learning process itself to best address this challenge. Fortunately, modern success can greatly simplify the search space and speed up the search process, often discovering models much larger, deeper and with better performance than previously thought possible. Further reading this section will provide more resources on the subject if you are looking to go deeper. Books Papers Articles Summary for this post, you discovered the challenge of training a neural network framed by an optimization problem. Specifically, you learned: Training the neural network involves using an optimization algorithm to find a set of scales for the best map inputs for outputs. The problem is difficult, not only because the fault surface is non-kulev and contains local minins, flat spots and is very multidimensional. The stohastic gradient descent algorithm is the best overall algorithm to solve this complex problem. Do you have any questions? Ask your questions in the comments below and I will do my best to answer. ... just a few lines of python code Discover how my new ebook: Better Deep learning It provides self-learning tutorials on topics such as: weight decay, batch normalization, dropout, model stacking and more ... Bring your projects a better deep learning! Skip the academics. Just the results. Look what's inside