PROJECT PROPOSAL

Stochastic Gradient Descent

Raman Bansal and Ruchir Garg

Dr. Kapil Ahuja

Introduction

There have been substantial developments in the field of neural networks. Some of them have been driven by external factors such as the increase of available data and computing power. The Internet made public massive amounts of labeled and unlabeled data. The ever-increasing raw mass of user-generated and sensed data is made easily accessible by databases and Web crawlers. Nowadays, anyone having an Internet connection can parse the 4,000,000+ articles available on Wikipedia and construct a dataset out of them. Anyone can capture a Web TV stream and obtain days of video content to test their learning algorithm. Another development is the amount of available computing power that has continued to rise at steady rate owing to progress in hardware design and engineering. While the number of cycles per second of processors has thresholded due to physics limitations, the slow down has been offset by the emergence of processing parallelism, best exemplified by the massively parallel graphics processing units (GPU). These developments have raised the following question: Can we make use of this large computing power to make sense of these increasingly complex datasets? Neural networks are a promising approach, as they have the intrinsic modeling capacity and flexibility to represent the solution. Their intrinsically distributed nature allows one to leverage the massively parallel computing resources. During the last two decades, the focus of neural network research and the practice of training neural networks underwent important changes. Learning in deep (or “deep learning”) has to a certain degree displaced the once more prevalent regularization issues, or more precisely, changed the practice of regularizing neural networks. This new paradigm has started to spread over a large number of applications such as image recognition, speech recognition, natural language processing, complex systems, neuroscience, and computational physics. We show that a stochastic gradient descent (learning one example at a time) is suited for training most neural networks.

Learning and Generalization

There are several approaches to automatic machine learning, but much of the successful approaches can be categorized as gradient-based learning methods. The learning machine, as represented in Figure 1.1, computes a function *M(Zp, W)* where *Zp* is the p-th input pattern, and *W* represents the collection of adjustable parameters in the system. A cost function *Ep = C(Dp, M(Zp, W)),* measures the discrepancy between Dp, the “correct” or desired output for pattern *Zp*, and the output produced by the system. The average cost function *Etrain(W)* is the average of the errors Ep over a set of input/output pairs called the training set *{(Z1, D1), ...(ZP , DP)}*. In the simplest setting, the learning problem consists in finding the value of W that minimizes *Etrain(W)*. In practice, the performance of the system on a training set is of little interest. The more relevant measure is the error rate of the system in the field, where it would be used in practice. This performance is estimated by measuring the accuracy on a set of samples disjoint from the training set, called the test set. The most commonly used cost function is the Mean Squared Error:

$$E^p = \frac{1}{2}(D^p - M(Z^p, W))^2, \qquad E_{train} = \frac{1}{P}\sum_{p=1} E^p$$

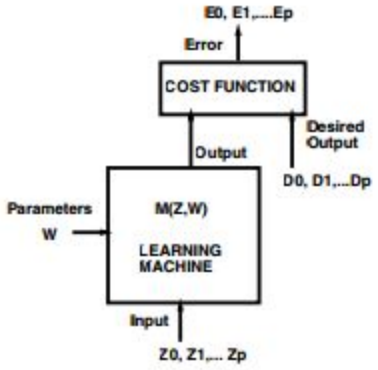

Fig. 1.1. Gradient-based learning machine

Problem Formulation

Let us first consider a simple supervised learning setup. Each example *z* is a pair *(x, y)* composed of an arbitrary input *x* and a scalar output *y*. We consider a loss function *l(yˆ, y)* that measures the cost of predicting *yˆ* when the actual answer is *y*, and we choose a family *F* of functions *fw(x)* parametrized by a weight vector *w*. We seek the function *f* *F* that minimizes the loss *Q(z,w) = (fw(x), y)* averaged on the examples. Although we would like to average over the unknown distribution *dP(z)* that embodies the Laws of Nature, we must often settle for computing the average on a sample *z1 ...zn.*

$$E(f) = \int \ell(f(x), y)\, dP(z) \qquad E_n(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i)$$

The *empirical risk En(f)* measures the training set performance. The expected risk *E(f)* measures the generalization performance, that is, the expected performance on future examples. The statistical learning theory justifies minimizing the empirical risk instead of the expected risk when the chosen family *F* is sufficiently restrictive.

Gradient Descent

It has often been proposed to minimize the empirical risk *En(fw)* using gradient descent (GD). Each iteration updates the weights w on the basis of the gradient of *En(fw)*,

$$w_{t+1} = w_t - \gamma \frac{1}{n} \sum_{i=1}^{n} \nabla_w Q(z_i, w_t) ,$$

where γ is an adequately chosen learning rate. Under sufficient regularity assumptions, when the initial estimate Wo is close enough to the optimum, and when the learning rate γ is sufficiently small, this algorithm achieves linear convergence. Much better optimization algorithms can be designed by replacing the scalar learning rate γ by a positive definite matrix Γ t that approaches the inverse of the Hessian of the cost at the optimum :

$$w_{t+1} = w_t - \Gamma_t \frac{1}{n} \sum_{i=1}^{n} \nabla_w Q(z_i, w_t) .$$

This second order gradient descent (2GD) is a variant of the well known Newton algorithm. Under sufficiently optimistic regularity assumptions, and provided that Wo is sufficiently close to the optimum, second order gradient descent achieves quadratic convergence.

Traditional numerical optimization methods, such as gradient descent, and Newton's method can be used to minimize the objective function.
However, when the sample size is very large, we need to evaluate the gradients of all the summand functions, which can be very time-consuming. Given the limitation of the traditional methods,stochastic gradient descent is proposed to speed up the computation by approximating the true gradient by the sum of a randomly selected subset of the summand functions.

Stochastic Gradient Descent

The stochastic gradient descent (SGD) algorithm is a drastic simplification. Instead of computing the gradient of *En(fw)* exactly, each iteration estimates this gradient on the basis of a single randomly picked example

$$w_{t+1} = w_t - \gamma_t \nabla_w Q(z_t, w_t) .$$

The stochastic process *{wt, t= 1,... }* depends on the examples randomly picked at each iteration. It is hoped that behaves like its expectation despite the noise introduced by this simplified procedure. Since the stochastic algorithm does not need to remember which examples were visited during the previous iterations, it can process examples on the fly in a deployed system. In such a situation, the stochastic gradient descent directly optimizes the expected risk, since the examples are randomly drawn from the ground truth distribution.

Application

The stochastic gradient descent algorithm can be applied to a number of classic machine learning schemes. We plan to apply it to linear regression, logistic regression.

References

Bottou, L. (2012) Stochastic gradient descent tricks.. *In Neural Networks: Tricks of the Trade,* 421–436. Springer.