Short Introduction to Deep Learning and its Math – Part II

Branislav Kisačanin
Nvidia Corporation and AwesomeMath Academy
b.kisacanin@ieee.org

Abstract

Since the "Deep Learning Big Bang" started in 2012, we are witnessing an unprecedented penetration of artificial intelligence into most, if not all, spheres of life, including science, engineering, medicine, business, politics, law, and entertainment. It is widely recognized that the main ingredients fueling this explosion of deep learning applications have been: advances in algorithms, availability of data, and developments in GPU computing. In this paper we briefly discuss these factors and slice through the algorithms to see the math at their foundations. We show that the key mathematical disciplines that empower deep learning are linear algebra, calculus, optimization, and probability and aim to explain that gaining a more fundamental understanding of why deep neural networks are so good at learning, will open the doors to even greater discoveries but also greater responsibilities.

Keywords: Artificial intelligence, machine learning, deep learning, neural networks.

1 Introduction

In Part I of this paper we discussed the constituent parts of the "Deep Learning Big Bang" and took a look at the math behind basic regression and classification learning approaches. They are a great place to learn the basic blocks of much more complex deep neural networks.

We begin Part II with a discussion of issues encountered during training. They happen even with simple regression and classifier models. Of particular interest is the problem of overfitting. Accordingly, we cover regularization of the cost function, which helps us alleviate overfitting. After that we talk about deep neural networks and the current state of affairs in this discipline. Finally we provide pointers for further reading and learning.

2 Learning is not easy

Even with simple regressions and classifiers, we observe a number of challenges during the training process. We will now consider some of them because we encounter them with DLNs too. To better understand these issues, let us briefly describe the training process beyond the iterative optimization we discussed earlier.

Typically, labeled datasets we use for training are divided into three subsets:

- *Training data.* Batches or the entirety of data from this subset are used to compute gradients that steer the learning process. As iterations progress, we evaluate the accuracy the model achieves on training data and report that as *training accuracy.*

- *Validation data.* This subset is not used in gradient computations, i.e., it does not affect learning directly, rather it is used to evaluate the accuracy of the model as it continues to learn to allow us to adjust the *learning hyper-parameters* such as the number of learning iterations or the model size. The accuracy the model achieves on validation data is the *validation accuracy.*

- *Test data.* This subset is not used in training at all, neither for the model parameters nor for learning hyper-parameters. It is kept away from the training process for objective evaluation of generalization properties of the trained model, i.e., how well the model works on unseen data. The accuracy the model achieves on test data is the *test accuracy.*

Here are several typical issues encountered during training. We also discuss established ways to address them.

- *Slow rate or lack of convergence of training accuracy.* If during training we observe that the training accuracy is slow to change or that it behaves erratically, the first thing to try is to change the learning step size. It it is too small, each learning iteration will move too slowly towards the optimal solution. On the other hand, if it is too large, the iterations may be jumping over the optimal solution, resulting in erratic behavior of the training accuracy. A good question here would be: How do we know whether convergence is slow? We don't right away, but if we try to increase the learning step size and it speeds things up, we know it was slow.

- *Unsatisfactory training accuracy.* First of all, how do we know if the training accuracy is satisfactory or not? In general, we always treat it as unsatisfactory (unless it is 100%) and want to improve it. We can try different model architectures and different model sizes. Using a model with more parameters typically increases the learning capacity of the model. In all this we have to be careful because using models with unnecessarily large number of parameters will likely result in overfitting (see the next item in this list) and will require more computational resources than necessary. This is a concern even when deploying your network on big servers and especially in embedded applications, such as autonomous driving. It all costs money in the end.

- *Overfitting.* If during training we observe a large difference between training and validation accuracy, this indicates that the model is not generalizing well what it learned from the training subset of our data. This phenomenon is called *overfitting* and indicates that the model has become specialized for the training data by relying on features that are not really representative of actual object classes and as a consequence has poor generalization properties. In general, we can reduce overfitting by

  - enlarging or augment the training set
  - using regularization

  Adding more data to a dataset is certainly desirable, but when it is not feasible, there are inexpensive ways to do a lot in that direction. For image data, the existing images in the dataset can be modified in different ways while preserving their labels. For example, an image of a dog remains a picture of a dog if we crop it, flip it around a vertical axis, change its brightness and contrast. This is call dataset augmentation. Regularization is another important technique to improve generalization properties of our trained model and we dedicate the next section to it.

3 L^2 regularization helps with generalization

In this section we take a closer look at L^2 regularization, an important method for improving generalization properties of models. We provide three different interpretations: intuitive, how it modifies iterations, and probabilistic.

3.1 Intuitive interpretation of regularization

If we have two trained models with equal training accuracy, but one model has much greater variability of parameters than the other, we expect the model with smaller parameter variability to generalize better, i.e., to have better accuracy on previously unseen data. Why? This is easy to understand on the example of image data: In the model with high variability of parameter values, the parameters with high absolute values in a way focus the decision of the model on a few image pixels, i.e., on a few localities. Consider an image of a dog that is classified well by a high parameter variability model. Shift the image slightly. It is still an image of a dog, but with everything shifted, it is not likely to be properly classified because the high variability model focuses on only a few pixels.

One way to keep the variability of the network parameters in check is to add a "regularization" term, penalizing high parameter values, to the cost function:

$$J_r(w) = J(w) + \alpha \|w\|^2.$$

Here w are the model parameters, h_w is the model, and X and y are the training inputs and the corresponding labels. α is another learning hyper-parameter that needs to be determined experimentally to balance the cost contributions of the two terms.

Historically, regularization was used in mathematics, for example to reduce Runge's phenomenon[1], observed in interpolation polynomials: Given n points in a plane, there is one and only one polynomial of order $n+1$ which fits those points perfectly. However, for $n > 4$, the values of the polynomial for points between the fitted points display a great variability that increases with n. For that reason a technique in which we use polynomials of order greater than $n+1$ is used along with a regularization term in the error function, which keeps the magnitudes of polynomial coefficients in check. With that, the variability of interpolation is reduced.

Similar to this, in DLN training we start with a model which can learn the training data really well. If we observe overfitting, we add the L^2 regularization term to the cost function to improve model's generalization properties.

3.2 Influence of L^2 regularization on iterations

If we consider the iterations for model parameters w when the cost function is augmented with the regularization term, then, with s denoting the learning step size,

$$w_{k+1} = w_k - s \operatorname{grad} J_r(w).$$

From $J_r(w) = J(w) + \alpha \|w\|^2$ we find

$$\operatorname{grad} J_r(w) = \operatorname{grad} J(w) + 2\alpha w,$$

which leads us to modified iterations

$$w_{k+1} = w_k(1 - 2s\alpha) - s \operatorname{grad} J(w).$$

[1]There is a related phenomenon in Fourier analysis called Gibbs' phenomenon.

Compared to the iterations without regularization, $w_{k+1} = w_k - s\,\text{grad}\,J(w)$, we see that the baseline for each iteration is reduced in magnitude: $\|w_k(1-2s\alpha)\| < \|w_k\|$.

This steers the parameters towards smaller values, especially in the direction of eigenvectors of the Hessian matrix corresponding to its smaller eigenvalues, i.e., in the directions in which the cost function changes slower. This becomes apparent when we look at the minimum of the regularized cost function w_r^*, where $\text{grad}\,J_r(w_r^*) = 0$. The Taylor series for $\text{grad}\,J(w)$ around its minimum w^* gives us

$$\text{grad}\,J(w_r^*) \approx \underbrace{\text{grad}\,J(w^*)}_{=0} + H(w_r^* - w^*),$$

so, at w_r^* we can write

$$\text{grad}\,J_r(w_r^*) = 0$$
$$2\alpha w_r^* + H(w_r^* - w^*) = 0$$
$$(H + 2\alpha I)w_r^* = Hw^*$$
$$w_r^* = (H + 2\alpha I)^{-1}Hw^*$$

Since Hessian is symmetric, its eigen-decomposition is equal to its singular value decomposition, therefore

$$H = Q\Lambda Q^T.$$

This allows us to continue

$$\begin{aligned} w_r^* &= (Q\Lambda Q^T + 2\alpha I)^{-1}Q\Lambda Q^T w^* \\ &= (Q(\Lambda + 2\alpha I)Q^T)^{-1}Q\Lambda Q^T w^* \\ &= Q(\Lambda + 2\alpha I)^{-1}\Lambda Q^T w^* \end{aligned}$$

The matrix $(\Lambda + 2\alpha I)^{-1}\Lambda$ is diagonal, with diagonal elements less than 1, especially in positions of smallest eigenvalues.

3.3 Probabilistic interpretation of L^2 regularization

If we formulate the task of finding the optimal values of model parameters in terms of a Bayesian maximum a posteriori probability (MAP) estimation, we find another way of convincing ourselves that L^2 regularization ensures we get parameters with moderate variability:

$$\begin{aligned} w_{MAP} &= \arg\max_w p(w|X) \\ &= \arg\max_w \; \log\, p(w|X) \\ &= \arg\max_w \; \log(p(X|w)p(w)) \\ &= \arg\max_w \; (\log\, p(X|w) + \log\, p(w)) \end{aligned}$$

One way to ensure our parameters have a small probability of having large magnitudes is to require $w \sim \mathcal{N}(0, I)$, when $\log\, p(w)$ yields the L^2 regularization term:

$$\log\, p(w) = w'w = \|w\|^2.$$

Interesting connection!

4 Deep Learning Networks: alchemy or science?

Having explained the process of training a classifier on simple, yet representative models, and having discussed various issues associated with learning and steps we can take to mitigate them, we are now ready to talk about the quantum leap in model complexity that happened in late 2000's and early 2010's. The prime example of this quantum leap was AlexNet, the DLN which won the 2012 ImageNet competition by a large margin. Since then practically all contestants, and certainly all winners of ImageNet competitions were DLNs of one sort or another.

It is instructive to look at the evolution of DLNs during the first years of the Deep Learning Bib Bang, say between 2012 and 2015. During that time, the numbers of layers, parameters, multiply-and-accumulate (MAC) operations simply exploded, leading to a steady reduction in error rates, and allowing the DLNs to achieve performance better than humans (estimated at about 5% error). In Table 1 we summarize these numbers:

Year	Network	Layers	Parameters	MACs	Top-5 error
2012	AlexNet	8	61M	724M	16.4%
2014	GoogLeNet	22	7M	1.4G	6.7%
2015	ResNet	152	60M	11.3G	3.6%

Note that due to an improved architecture, i.e., the way how the neurons are organized into layers and how the information flows through the network, GoogLeNet has a better performance, yet 9 times fewer parameters to train and 2.5 times lower error rate than AlexNet.

The trend of introducing new networks with novel ideas on how to improve their computational efficiency, representational ability, and memory footprint has continued to this day and deserves much more analysis than we can fit into a paper.

The developments in DLNs today are reminiscent of the brave age of discovery of chemical elements in the early 19th century. After the invention of the Voltaic pile by Alessandro Volta in 1799, chemists, most notably Humphry Davy, have applied electrical current to discover many new chemical elements using electrolysis. Looking at further analogies with chemistry, we expect the age of DLN discovery to continue, and to especially accelerate if we get the benefit of solid scientific classification of DLNs, such as was accomplished by Dmitri Mendeleev, with his periodic system of elements in 1869.

Hopefully, such a systematization will also help us understand how it is possible that neural networks can learn and generalize the real world knowledge.

5 Further Reading

To help the interested reader learn more, here we list several basic Machine Learning and DLN classes available on the web and freely available online books:

- A. Ng, *Machine Learning*, Stanford and Coursera (link)
- A. Karpathy et al, *CNNs for Visual Recognition*, Stanford CS 231n (link)
- I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press (2016) (link)
- C. M. Bishop, *Statistical Learning and Pattern Recognition*, Springer (2006) (link)

References

[1] J. M. Stokes et al., A deep learning approach to antibiotic discovery, *Cell.* Volume 180, Issue 4, pp. 688-702, 2020.

[2] M. Daily et al., Editors, Self-driving cars, Special issue of *Computer*, Volume 50, Issue 12, 2017.

[3] E. de Vries, et al., No Longer Lost in Translation: Evidence that Google Translate Works for Comparative Bag-of-Words Text Applications, *Political Analysis*, Volume 26, Issue 4, pp. 417-430, 2018.

[4] J. Sommerlad, Google Translate, *Independent*, 19 June 2018. (link)

[5] T. Park et al., Semantic image synthesis with spatially-adaptive normalization, *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* 2019. (link)

[6] M. Burns, Nvidia AI turns sketches into photo-realistic landscapes in seconds, *The Crunch*, 18 March 2019. (link)

[7] T. J. Sejnowski, The unreasonable effectiveness of deep learning in artificial intelligence, *Proc. of the National Academy of Sciences*, 2020.

[8] E. P. Wigner, The unreasonable effectiveness of mathematics in the natural sciences, *Comm. on Pure and Applied Mathematics*, Volume 13, pp. 1-14, 1960.

[9] W. S. McCulloch and W. Pitts, A logical calculus of the ideas immanent in nervous activity, *Bulletin of Mathematical Biophysics*, 1943.

[10] Y. Lecun et al., Gradient-based learning applied to document recognition, *Proceedings of the IEEE*, 1998.

[11] A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet classification with deep convolutional neural networks, *NIPS*, 2012.

[12] K. He et al., Deep Residual Learning for Image Recognition, *CVPR*, 2016.