Modern Machine Learning Algorithms: Applications in Nuclear Physics

by

Robert Solli

Thesis

for the degree of

Master of Science

Faculty of Mathematics and Natural Sciences
University of Oslo

4th March 2019

Abstract

In this thesis a novel filtering thechinque of AT-TPC noise events is presented using clustering techniques on the latent space produced by a Variational Autoencoder(VAE)

Chapter 1

Theory

1.1 Neural networks

While the basis for the modern neural network was laid more than a hundred years ago in the late 1800's but what we think of in modern terms was proposed by McCulloch und Pitts (1943). They described a computational structure analogous to a human neuron. Dubbed an Artificial Neural Network (ANN) it takes input from multiple sources, weights that input and produces an output if the signal from the weighted input is strong enough. A proper derivation will follow but for the moment we explore this simple intuition. These artificial neurons are ordered in layers, each successively passing information forward to a final output. The output can be categorical or real-valued in nature. A simple illustration of two neurons in one layer is provided in figure 1.1

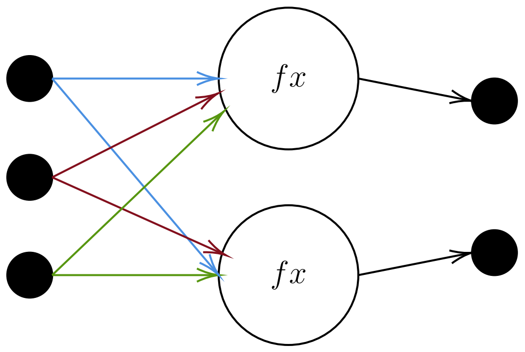

Figure 1.1: An illustration of the graph constructed by two artificial neurons with three input nodes. Colored lines illustrate that each of the input nodes are connected to each of the neurons in a manner we denote as fully-connected.

The ANN produces an output by a "forward pass". If we let the input to an ANN be $x \in \mathrm{R}^N$, and letting the matrix $W \in \mathrm{R}^{N \times D}$ be a representation of the weight matrix forming the connections between the input and the artificial neurons. Lastly we define the activation function $a(x)$ as a monotonic, once differentiable, function on R^1. The function $a(x)$ determines the complexity of the neural network together with the number of neurons per layer and number of layers. For any complex task the activation takes a non-linear form which allows for the representation of more complex problems. A layer in a network implements what we will call a forward pass as defined in function 1.1.

$$\hat{y} = a(\langle x|W\rangle)_D \tag{1.1}$$

In equation 1.1 the subscript denotes that the function is applied element-wise and we denote the matrix inner product in bra-ket notation with $\langle\cdot|\cdot\rangle$. Each node is additionally associated with a bias node ensuring that even zeroed-neurons can encode information. Let the bias for the layer be given as $b \in \mathrm{R}^D$ in keeping with the notation above. Equation 1.1 then becomes:

$$\hat{y} = a(\langle x|W\rangle)_D + b \tag{1.2}$$

As a tie to more traditional methods we note that if we only have one layer and a linear activation $a(x) = x$ the ANN becomes the formulation for a linear regression model. In our model the variables that need to be fit are the elements of W that we denote W_{ij}. While one ordinarily solves optimization problem for the linear regression model by matrix inversion, we re-frame the problem in more general terms here to prime the discussion of the optimization of multiple layers and a non linear activation function. The objective of the ANN is formulated in a "loss function", which encodes the difference between the intended and achieved output. The loss will be denoted as $\mathcal{L}(y, \hat{y}, W)$. Based on whether the output is described by real values, or a set of probabilities this function, $\mathcal{L}$, takes on the familiar form of the Mean Squared Error or in the event that we want to estimate the likelihood of the output under the data; the binary cross-entropy. We will also explore these functions in some detail later. The ansatz for our optimization procedure is given in the well known form of a gradient descent procedure in equation 1.3

$$W_{ij} \leftarrow -\eta \frac{\partial \mathcal{L}}{\partial W_{ij}} + W_{ij} \tag{1.3}$$

1.1.1 Backpropagation

In the vernacular of the machine learning literature the aim of the optimization procedure is to "train" the model to perform better on the regression, reconstruction or classification task at hand. Training the model requires the computation of the total derivative in equation 1.3. This is also where the biological metaphor breaks down, as the brain is almost certainly not employing an algorithm so crude as to be formulated by gradient descent. Backpropagation, or automatic differentiation, described most famously in Chapter 8 of **?** is a method of computing the partial derivatives required to go from the gradient of the loss w.r.t the output of the ANN to the gradient w.r.t the individual neuron weights in the layers of the ANN. The algorithm begins with computing the total loss, here exemplified with the squared error function, in equation 1.4

$$E = \mathcal{L}(y, \hat{y}, W) = \frac{1}{2}\sum_n \sum_j (y_{nj} - \hat{y}_{nj})^2 \tag{1.4}$$

The factor one half is included for practical reasons to cancel the exponent under differentiation. As the gradient is multiplied by an arbitrary learning rate η this is ineffectual on the training itself. The sums define an iteration over the number of samples, and number of output dimensions respectively. Taking the derivative of 1.4 w.r.t the output, $\hat{y}$, we get

$$\frac{\partial E}{\partial \hat{y}_j} = \hat{y}_j^M - y_j \tag{1.5}$$

Recall now that for an ANN with M layers the output fed to the activation function is

$$x_j^M = \langle a^{M-1} | W^M \rangle + b_j \tag{1.6}$$

Where the superscript in the inner product denote the output of the second-to-last layer and the weight matrix being the last in the layers. The vector x_j is then fed to the activation to compute the output

$$\hat{y}_j^M = a(x_j^M) \tag{1.7}$$

The activation function was classically the sigmoid (logistic) function but during the last decade the machine learning community has shifted to largely using the rectified linear unit (ReLU) as activation. Especially after the success of **?** with AlexNET in image classification. Depending on the output (be it regression or classification) it might be useful to apply the identity transform or a soft max function in the last layer. This does not change the derivation except to change the derivatives in the last layer. We here exemplify the back propagation with the ReLU, which has the form

$$\text{ReLU}(x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{1.8}$$

The ReLU is obviously monotonic and its derivative can be approximated with the Heaviside step-function.

$$H(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{1.9}$$

We again make explicit that knowing the form of equations 1.4, 1.8 and 1.9 is not necessary but provided for clarity. We then return to equation 1.5 and manipulate the expression via the chain rule

$$\frac{\partial E}{\partial x_j} = \frac{\partial E}{\partial \hat{y}_j}\frac{\partial \hat{y}_j}{\partial x_j} \tag{1.10}$$

The second derivative of the r.h.s we know from our choice of the activation to be equation 1.9, inserting to evaluate the expression we find

$$\frac{\partial E}{\partial x_j^M} = (\hat{y}_j - y_j)H(x_j^M) \tag{1.11}$$

To complete the derivation we further apply the chain rule to find the derivative in terms of the weight matrix elements.

$$\frac{\partial E}{\partial w_{ij}^M} = \frac{\partial E}{\partial x_j^M}\frac{\partial x_j^M}{\partial w_{ij}^M} \tag{1.12}$$

Recall the definition of x_j as the affine transformation defined in equation 1.1. The derivative of the inner product w.r.t the matrix elements is simply the previous layers output. Inserting this derivative of equation 1.6 we have the expression for our derivatives of interest.

$$\frac{\partial E}{\partial w_{ij}} = (\hat{y}_j - y_j)H(x_j)\mathrm{ReLU}(x_i^{M-1}) \tag{1.13}$$

Separately we compute the derivatives of 1.11 in terms of the bias nodes.

$$\frac{\partial E}{\partial b_j} = \frac{\partial E}{\partial x_j}\frac{\partial x_j}{\partial b_j} = (\hat{y}_j - y_j)H(x_j)\cdot 1 \tag{1.14}$$

This procedure is then repeated for the earlier layers computing the $\partial E/\partial w$ as we go. The backward propagation framework is highly generalizable to variations of activation functions and network architectures. The two major advancements in the theory of ANNs are both predicated on being fully trainable by the back-propagation of errors. Before we consider these improvements made by the introduction of Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN) we remark that not only are we free to chose the activation function remarkably freely the backpropagation algorithm also makes no assumptions on the transformation that constructs x_j. As long as it is once differentiable in terms of w_{ij} we are free to pick this transformation also.

1.2 Autoencoders

An Autoencoder is an attempt at learning a distribution over some data by reconstruction. The model consists of two discrete parts ; an encoder and a decoder. Where the encoder is in general a non linear map ψ

$$\psi : \mathcal{X} \rightarrow \mathcal{Z}$$

Where $\mathcal{X}$ and $\mathcal{Z}$ are arbitrary vector spaces with $\dim(\mathcal{X}) > \dim(\mathcal{Z})$. The second part of the model is the decoder that maps back to the original space.

$$\phi : \mathcal{Z} \rightarrow \mathcal{X}$$

The objective is then to find the configuration of the two maps ϕ and ψ that gives the best possible reconstruction, i.e the objective $\mathcal{O}$ is given as

$$\mathcal{O} = \arg\min_{\phi,\psi} ||X - \phi \circ \psi||^2 \tag{1.15}$$

Where the $\circ$ operator denotes function composition in the standard manner. As the name implies the encoder creates a lower-dimensional "encoded" representation of the input. This objective can be optimi This representation can be useful for identifying whatever information-carrying variations are present in the data. This can be thought of as an analogue to Principal Component Analysis (PCA) Marsland (2009). In practice the non-linear maps, ψ and ϕ, are most often parametrized and optimized as ANNs. ANNs are described in detail in section 1.1. More recently the Machine Learning community discovered that the decoder part of the network could be used for generating new samples form the sample distribution, dubbed "Variational Autoencoders" they are among the most useful generative algorithms in modern machine learning.

1.2.1 Variational Autoencoder

Originally presented by Kingma und Welling (2013) the Variational Autoencoder (VAE) is a twist upon the traditional autoencoder. Where the applications of an ordinary autoencoder largely extended to de-noising with some authors using it for dimensionality reduction before training an ANN on the output the VAE seeks to control the latent space. The goal is to be able to generate samples from the unknown distribution over the data. Imagine trying to draw a sample from the distribution of houses, we'd be hard pressed to produce anything remotely useful but this is the goal of the VAE.

Bibliography

[Kingma und Welling 2013] KINGMA, Diederik P. ; WELLING, Max: Auto-Encoding Variational Bayes. (2013), dec. – URL http://arxiv.org/abs/1312.6114

[Marsland 2009] MARSLAND, Stephen: *Machine Learning: An Algorithmic Perspective*. 2009

[McCulloch und Pitts 1943] MCCULLOCH, Warren S. ; PITTS, Walter: A logical calculus of the ideas immanent in nervous activity. In: *The Bulletin of Mathematical Biophysics* 5 (1943), dec, Nr. 4, S. 115–133. – URL http://link.springer.com/10.1007/BF02478259. – ISSN 0007-4985