INNS COM00029H Open Assessment

Y3843100

March 23, 2020

1 [20 marks] Discussion of architectures.

This section should:

- describe (briefly) the data you have, and how much there is of it.
- identify the type of problem
- identify which classes of architectures would be suitable
- give a brief discussion of the technical features of the architectures, and the advantages and disadvantages of each
- state which class of architecture you are going to use and justify your choice, relating the characteristics of the problem to the advantages/disadvantages of the architecture.

To do this you might need to:

- do some preliminary experiments with simple versions of the architecture to get a feel for what will work
- do some exploratory data analysis to see what the characteristics of the data are
- consider the principles involved and relate them to the problem.

The dataset contains fetal cardiotocograms (CTGs) from 2126 patients[1] each of which has 21 different recorded features (input variables). The CTGs have been annotated by three expert obstetricians creating two categories of classes.[2] One is a 10 tuple with respect to the fetal heart rate FHR patterns and the other is a three tuple regarding fetal state. This gives us two supervised classification problems with respectively 10 and 3 distinct classes. Neural Network architectures that can handle classification problems need to have appropriate activation functions and the ability to specify the targets (outputs) as a finite set of discrete classes. We will discuss **three** different conceptual models with respect to their ability to be configured for classifying our high dimensional CTG dataset.

[1]Zahra Hoodbhoy et al. "Use of machine learning algorithms for prediction of fetal risk using cardiotocographic data". In: *International Journal of Applied and Basic Medical Research* 9.4 (2019), p. 226.

[2]Ayres de-Campos Bernardes Garrido Marques-de Sa Pereira-Leite. *UCI Machine Learning Repository*. 2000. URL: `https://archive.ics.uci.edu/ml/datasets/cardiotocography`.

1.0.1 Perceptron

The *perceptron* is a basic network structure in which our output class y is determined by a weighted sum of our inputs X that is evaluated against some hard limit (threshold or activation function) $y = H(\sum_i^X x_i w_i)$. Both the advantages and disadvantages of the perceptron are in its simplicity. On one hand we have intuitive behaviour in the fact that the perceptron finds a line that bipartitions our data space, but on the other we are limited only to linearly separable classes. Furthermore a perceptron can perform only binary classification or at best *one-vs-many*.

1.0.2 Multi-Layer Perceptron MLP networks

The shortcomings of the single perceptron are addressed by its orchestrated counterpart, the multi-layer perceptron *MLP*. The three main differences as highlighted by Haykin[3] are:

- neuron activation functions are differentiable (sigmoid functions are often chosen), unlike the hard limits we had before
- between our input and output layers we construct one or more *hidden layers* containing one or more neurons
- the input, output and hidden neurons are highly connected

By composing neurons together we can learn more complex patterns at the expense of more complicated learning rules. The benefit of MLPs is that that they can approximate virtually any function provided there is enough data. The disadvantages of using MLPs come from the fact that they are prone to overfitting on high dimensional data and they do not necessarily have a simple intuitive meaning as the perceptron classifier. We can theoretically use MLPs for our two classification problems as we can specify the number of output neurons to be three and ten respectively. An issue arrises with what network structure would be viable to capture the properties of our high dimensional data.

1.0.3 Radial Basis Function RBF networks

RBF networks are a single hidden layer MLP where Euclidean distance between the inputs and some point in space associated with the neuron's centre is computed instead of linear activation function. More specifically, the hidden layers calculate a radially-symmetric function (usually a Gaussian) from the inputs $f_i(x) = \exp\left(-\frac{|x-c_i|^2}{2\sigma^2}\right)$ where c_i is the centre for neuron i. The outputs are the weighted sums of the different basis functions in the hidden layer. RBF networks classify new data points by associating them to the closest c_i. The benefit of this architecture is that it is often faster to train compared to MLPs, but a substantial drawback is that they struggle with generalising outside of the margins of the training data. But an argument can be made that for our specific task, if we get new patient data that has low response for all of our current classes, that is a potential indicator of an anomaly that would require further investigation.

In the choice of an architecture it is important to look at the data itself. Our data does not appear to be linearly separable and our task is to discriminate between all the distinct classes, therefore we rule out the perceptron as a viable architecture. MLP and RBF are theoretically viable for both of our classification problems. They both allow for supervised multiclass classification problems and they can solve non-linearly separable problems. Because of our argument that low response from the RBF network is a useful indicator for an anomaly, we

[3]Simon Haykin. *Neural Networks: A Comprehensive Foundation*. 2nd. USA: Prentice Hall PTR, 1998. ISBN: 0132733501.

will still consider RBFs in this experiment. The severity of the disadvantages for MLP models can be lessened by paying close attention to the meaning of the data and incrementally explore different network structures, applying Occam's razor as a prerequisite. Because of this reasoning, we will empirically compare different MLP and RBF networks for our two classification problems.

2 [40 marks] Creation and application of neural networks.

This section should

- Describe the chosen inputs to (and outputs from) the networks.
- Describe how the data you started with have been preprocessed.
- Give sufficient detail for someone else to process a new batch of data for use with the final trained network.
- State which training algorithm you selected, and explain how you selected that training algorithm. For this training algorithm, give sufficient detail to enable someone to use the same training algorithm in exactly the same way. This does NOT mean (for example) describing gradient descent in great detail. It DOES mean giving any parameters, initialisation, etc, even if they are the toolbox defaults.
- Explain the process you went through in making the selection of the final architecture, for example, the number of neurons or the number of layers to use.

To do this you might need to:

- Test of one or more networks to demonstrate the effect of different preprocessing choices on the performance of the network.
- Try different training algorithms on one or more networks to compare performance.
- Evaluate a number of networks, and record details of their structures and how they performed. You may summarise repeated tests of the same structure.

A crucial observation for our classification problems is to note that classifying biological anomalies is naturally going to mean our classes are unbalanced **[[insert histogram of classes]]**. This is the case for both the FHR patterns and the fetal state problems. To mitigate this, we use a simple oversampling method that replicates data from our underrepresented cases until they match the most common class. This method does not give us any new knowledge about the problems, but by scaling the data we take a step towards unbiased estimations. We also one-hot encode our classes for use with MATLAB's `patternnet`.

Noise reduction by high frequency noise and FHR spike removal has been done by the original researchers, but plotting the correlation and distributions of the features reveals further preprocessing is required **[[insert graph of plotmatrix before preproc]]**. Unsurprisingly, we observe the mean, mode and median are highly correlated. We keep only the median as a feature importance analysis (minimum redundancy maximum relevance **[[cn]]**) reveals that it has the highest predictive capabilities. Subsequently we deal with the skewed distributions and non regularised parameters by looking into normalising the data. An empirical test reviews that $(\mu = 0, \sigma^2 = 1)$ regularisation surpasses $[-1; 1]$ normalisation performance.

what you need to run this NN We disregard training algorithms that compute the Jacobian as they require the network to use a mean-squared error *MSE* loss function, which is not appropriate for our tasks (elaboration on choice of loss functions can be found in section 3). An empirical comparison of different training algorithms for a network with a single hidden layer with $N = \{10, 15, 20, 25, 30, 35, 40, 45, 50\}$ neurons can be seen in figures **insert comparison CE** and **insert comparison time**. It appears that the family of conjugate gradient backpropagation *CGB* (traincgb, traincgp and traincgf) achieve the best trade-off between performance and training time. At a glance, the results of CGB with Powell-Beale restarts (traincgb) appears to be the most consistent (least noisy) and therefore we will chose it as a training algorithm. The weights and biases for traincgb are selected via a Nguyen-Widrow[4] algorithm, which evenly distributes the weights and biases for the active regions across neurons at initialisation. CGB also does not use a fixed learning rate, but calculates the learning step at each iteration. All the initialisation parameters can be seen in figure **insert init param figure**.

how did we end up with this architecture Our output layers have 10 and 3 neurons respectively as we have one-hot encoded the class representation. Empirical comparisons were done by starting off with a single hidden layer network and incrementally add neurons to it. This was done independently for both classification problems.

3 [20 marks] Results and evaluation

This section should

- Explain the metric or metrics you have used for comparison between networks.
- Give a synopsis of the results obtained from the final selected network.
- Evaluate the results, in relation to the problem posed in the scenario.

To do this you might need to:

- Consider different metrics for performance, appropriate to the problem. Remember that a Mean Squared Error (MSE) on its own is not always helpful in judging how well something works.
- Identify anything of interest in the results, such as areas of particularly good or poor performance, or variation between different training runs.
- Reflect on the conclusions that you may draw from the results, and whether they are showing that the neural network is useful in this case.

All models have been trained with a cross-validation with a ratio **[[train,test,validation]]** respectively for the train, test and validation datasets. All metrics listed here are against the test set as it is the only truly unbiased estimator as the validation set is used to tune the hyperparameters of our network **[[cn]]**.

In regards to loss functions, as the decision space for a classification differs from that of a regression problem, we need to be wary of the usefulness of the MSE. **elaborate on why MSE is bad** .Due to that fact we have chosen $crossentropy = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C}(T_{i,j}log(X_{i,j}) + (1-T_{i,j})log(1-X_{i,j}))$ as our metric as it penalises neuron output dependent on how incorrect the prediction is. CE is shown to lead to better multinomial classification models compared to MSE.[5] A prerequisite to use it is the output neurons to have a softmax/sigmoidal activation function, which we conveniently have.

[4]MathWorks. *initnw (Nguyen-Widrow layer initialization function)*. Online; Retrieved March 23, 2020. 2020. URL: `https://uk.mathworks.com/help/deeplearning/ref/initnw.html`.

[5]Pavel Golik, Patrick Doetsch, and Hermann Ney. "Cross-entropy vs. squared error training: a theoretical and experimental comparison". In: *INTERSPEECH*. 2013.

4 [20 marks] Further application

In the previous sections you used a neural network to convert cardiotocogram features into a diagnosis. Another tool for detection and diagnosis of fetal abnormalities is the ultrasound scan, that produces an image of the a section through the fetus. Interpreting fetal scans is a highly complex task which require years of training. Assume the availability of a large number of fetal scan images, both normal and with some abnormality, and labelled to indicate different types of abnormalities. The task is for a neural network to process new ultrasound images, and to indicate which images needed further investigation. This section should discuss the issues you would need to consider in relation to:

- selection of an architecture
- construction of the network
- use of data for training
- evaluation of the network