On Neyman-Pearson optimality of binary neural net classifiers

Raymond Veldhuis [1] and Dan Zeng [2]

[1]University of Twente
[2]Affiliation not available

June 03, 2021

Abstract

In classical binary statistical pattern recognition optimality in Neyman-Pearson sense, achieved by a (log) likelihood ratio based classifier, is often desirable. A drawback of a Neyman-Pearson optimal classifier is that it requires full knowledge of the (quotient of the) class-conditional probability densities of the input data, which is often unrealistic. The design of neural net classifiers is data driven, meaning that no explicit use is made of the class-conditional probability densities of the input data. In this paper a proof is presented that a neural net can also be trained to approximate a log-likelihood ratio and be used as a Neyman-Pearson optimal, prior-independent classifier. Properties of the approximation of the log-likelihood ratio are discussed. Examples of neural nets trained on synthetic data with known log-likelihood ratios as ground truth illustrate the results.

On Neyman-Pearson optimality of binary neural net classifiers

Raymond Veldhuis, *Senior Member, IEEE,* Dan Zeng, *Member, IEEE*

Abstract—In classical binary statistical pattern recognition optimality in Neyman-Pearson sense, achieved by a (log) likelihood ratio based classifier, is often desirable. A drawback of a Neyman-Pearson optimal classifier is that it requires full knowledge of the (quotient of the) class-conditional probability densities of the input data, which is often unrealistic. The design of neural net classifiers is data driven, meaning that no explicit use is made of the class-conditional probability densities of the input data. In this paper a proof is presented that a neural net can also be trained to approximate a log-likelihood ratio and be used as a Neyman-Pearson optimal, prior-independent classifier. Properties of the approximation of the log-likelihood ratio are discussed. Examples of neural nets trained on synthetic data with known log-likelihood ratios as ground truth illustrate the results.

Index Terms—binary classification, Neyman Pearson, log-likelihood ratio, neural networks.

1 Introduction

There is a difference between how classical statistical pattern classifiers and classifiers based on (deep) neural nets are designed and optimised. In classical binary statistical pattern recognition optimality in Neyman-Pearson sense is often desirable [1]. A binary classifier is optimal in Neyman-Pearson sense if the probability of a false-negative decision is minimised given a maximum probability of a false-positive decision, or vice versa. The optimality can be local, at only one specific given probability of a false-positive decision, or global, for all false-positive probabilities. Unless stated otherwise, here we consider global Neyman-Pearson optimality. Neyman-Pearson optimality does not rely on the knowledge of prior probabilities of the classes – priors in short – and can be applied in cases where these are unknown or where it is important to guarantee a minimum probability of a false-positive decision.

Global Neyman-Pearson optimal classifiers output a statistic called the likelihood ratio, or a monotonic function thereof, which is compared to a threshold [1]. Only if the likelihood ratio exceeds the threshold, a positive decision is taken. The threshold controls the trade-off between the probabilities of false-positive and false-negative decisions. The last two decades have shown an increased interest in the likelihood ratio in the forensic sciences, where it is used to measure the strength of evidence in an objective way [2].

It can be seen as an advantage that a Neyman-Pearson optimal classifier is *prior-independent*, which renders it suitable for various applications that involve changing priors or various cost models. For instance, minimising Bayesian costs can be achieved by adapting the decision threshold, and thus the *point of operation* to the actual priors and the cost model.

A drawback of a Neyman-Pearson optimal classifier is that its design requires full knowledge of the (quotient of the) class-conditional probability densities of the input data. Usually this information is not available and the designer has to rely on assumptions about or estimates of these probabilities from training data. For high-dimensional input data, e.g. large images, this may not be a realistic option. As a compromise, often a sub-optimal, yet prior-independent, classifier is designed that, instead of a likelihood ratio, outputs a score that is compared with a threshold.

The design, and in particular the training, of neural net classifiers is data driven, meaning that no explicit use is made of the class-conditional probability densities of the input data. Neural net classifiers are generally not prior-independent, but are optimised for the frequencies of occurrence of the classes, or empirical priors, as they are represented in the training data. Commonly the output of a binary neural net classifier is a prediction of the input sample's posterior class probability, which depends on these priors. The training of neural net classifiers aims at minimising a loss function that reflects the aggregated class prediction errors, e.g. the binary cross-entropy loss [3]. Because the empirical priors determine the weight of class prediction errors in the loss function, they affect the result of the training.

The fact that well-trained neural nets can approximate posterior class probabilities raises the question *whether neural nets can also be trained to approximate a likelihood ratio* and be used as a Neyman-Pearson optimal, prior-independent classifier. This would make them more generally applicable outside the context of the (priors of the) training data, while the optimality would guarantee best possible classification performance at all points of operation.

The main contribution of this paper is a proof that sufficiently rich, i.e. with a sufficient number of neurons, binary neural net classifier can indeed be trained to be globally optimal in Neyman-Pearson sense, producing a log-

- *R. Veldhuis is with the University of Twente, Enschede, The Netherlands. E-mail: r.n.j.veldhuis@utwente.nl*
- *D. Zeng was with the University of Twente, Enschede, The Netherlands, and is currently with Southern University of Science and Technology, Shenzhen, China.*
- *Both authors contributed equally to this paper.*

Manuscript submitted June 1, 2021.

likelihood ratio as classification score. A second contribution is an analysis of the approximation of the log-likelihood ratio.

These results need to be put somewhat into perspective. In many applications, including biometric face recognition and medical imaging, the use of (deep) neural nets has dramatically improved the classification performance. In many of these cases the network is used as a feature (or representation) extractor and extracted features are then compared with a simple non-optimised classifier such as a Euclidean distance or a cosine-similarity classifier. Feature extractors can, for instance, be obtained by first training a multi-class deep neural net classifier and then removing the final fully connected layers, by training an autoencoder for feature extraction [4], or by applying a generically pre-trained network such as a version of ResNet [5]. The simple classifiers that compare extracted features have not been optimised for the problem at hand, yet the overall classification performance is often spectacular. We believe that this is because in the given application domains these networks are capable of extracting very discriminative features that are already well separable in feature space and hence do not benefit from an optimised classifier. Indeed, our analysis of the approximation behaviour of Neyman-Pearson optimal neural net classifiers shows that the objective function for training the network becomes flat if the probability densities of the features of the two classes have little or no overlap. This means that in those cases there is a large set of classifiers that all have near-optimal classification performance, illustrating that well-separable features do not benefit from classifier optimisation. Therefore, in our opinion *the importance of this paper is more for the less spectacular but nevertheless relevant classification problems characterised by features that are not well separable, and for which choosing an optimal classifier can make a difference, than for classification problems where the distribution of classes allows for almost perfect recognition.*

The remainder of this paper is organised as follows. Section 2 discusses the literature that provides the relevant background for this paper and discusses related work. Section 3 reviews an elementary result from Bayesian statistics that relates the log-likelihood ratio, posterior probability classifiers and the sigmoid activation function. Section 4 formalises the conditions that a neural network must satisfy in order to realise a log-likelihood ratio as a classification score. Section 5 then provides a proof that a properly trained neural network can actually realise a log-likelihood ratio. How the approximation of the log-likelihood ratio depends on its actual value, and how it is affected by unbalanced priors during training and by the discriminative properties of the data, is discussed in Section 6. In Section 7 we illustrate the theoretical results with a number of examples on synthetic data in which we compare approximated log-likelihood ratios with ground truth. Finally, Section 8 presents conclusions.

2 Related work

In machine learning, classifiers are often categorised as generative or discriminative [6]. As is explained in [7], the classical likelihood-ratio based classifier that relies on (estimates of) class-conditional probability density functions is a typical generative classifier, whereas the binary neural net classifier, based on maximising the maximum posteriori class probability via learning, is a typical discriminative classifier. A common characteristic of generative and discriminative classifiers is that their decisions are based on a discriminant function [7], which can be thresholded or turned into a posteriori probability. The difference is found in the way the discriminant function is obtained. For generative classifiers it follows from specifying the class-conditional probability densities, whereas for discriminative classifiers a parametrised function (e.g. linear or quadratic) is assumed for the discriminant and the parameters are obtained by training the classifier directly, often using logistic regression [7]. Training a binary neural net classifier terminated with a sigmoid activation function, as we describe in our main result in Section 5, can be seen as a generalised form of logistic regression, since the sigmoid is actually the logistic function. In this view the network realises the discriminant function of which the parameters are trained.

There are relatively few publications that relate discriminative machine learning to Neyman-Pearson optimality. Representative examples are [8], [9], [10], [11]. A comprehensive overview is given in [11]. These papers mainly study local Neyman-Pearson optimality of a classifier and its convergence properties at a given false-positive rate rather than focussing on the approximation of the log-likelihood ratio by a neural net. In fact, in most of these publications the likelihood ratio is only briefly mentioned and not discussed. The concept of a plug-in Neyman-Pearson estimator as described in [11] comes closest to our result.

An important assumption in our proof is that a sufficiently rich, well-designed neural net can approximate the log-likelihood ratio as a function of the input data. The first important result that supports this assumption is the universal approximation theorem [12] that says that a neural network with one hidden layer containing a finite number of neurons can approximate multivariate, continuous function. Later results provide relations between the accuracy of the approximation, the smoothness of the function to be approximated, and the depth and width of the network. From [13] and [14], it can be concluded that a neural network is capable of approximating a log-likelihood ratio, if it is a sufficiently smooth function of the input data. The latter is a reasonable assumption, as we may assume that class-conditional probabilities are often smooth functions of the input. In [14] function smoothness is formalised, but here the intuitive understanding that a function does not oscillate or jump too much on a limited interval suffices. In [15], [16], [17], [18] the trade-off between depth and width of neural nets is treated.

In the example in Subsection 7.2 we simulate the application of a log-likelihood ratio classifier for biometric comparison. One of the earlier papers demonstrating the potential of the log-likelihood ratio for biometric comparison is [19]. Our current approach to biometric comparison is similar to that in [20], and [21], which are based on classical (non-neural net based) feature extraction. In [22] and [23] log-likelihood ratio based comparison is applied to biometric features extracted by a CNN. In all these references, the underlying assumption is that the biometric features have Gaussian probability density functions. One

reason for this assumption is that an explicit form can be derived for Gaussian log-likelihood ratios with parameters that can be learned from data. Here we will demonstrate that a neural net can be trained to perform the same task and be optimised for the actual underlying probability densities.

3 Log-likelihood ratio and posterior probability

In this section we review a well-known relation, e.g. see [24, Section 4.2], between the posterior probability of a binary classifier and the log-likelihood ratio that is also at the basis of logistic regression parameter estimation [7].

A binary classifier aims to classify input samples $\boldsymbol{z} \in \mathcal{Z} \subset \mathbb{R}^n$, with $\mathcal{Z}$ the sample space, into the mutually exclusive classes C and $\overline{C}$. The classifier produces a discriminant, or score, $s(\boldsymbol{z})$. The sample $\boldsymbol{z}$ is classified into C if $s(\boldsymbol{z}) > T$, with T a predefined threshold controlling the trade-off between the probabilities of false-negative and false-positive decisions. The optimal choice for $s(\boldsymbol{z})$ in Neyman-Pearson sense – giving the best trade-off between the probabilities of false-negative and false-positive decisions – is the likelihood ratio, defined by

$$\mathrm{lr}(\boldsymbol{z}) \stackrel{\mathrm{def}}{=} \frac{p(\boldsymbol{z}|C)}{p(\boldsymbol{z}|\overline{C})}, \tag{1}$$

with $p(\boldsymbol{z}|C)$ and $p(\boldsymbol{z}|\overline{C})$ the class-conditional probability density functions of the random variable $\underline{\boldsymbol{z}}$. The relation between the posterior probability $P\{C|\boldsymbol{z}\}$ and the log-likelihood ratio $\mathrm{llr}(\boldsymbol{z}) = \log(\mathrm{lr}(\boldsymbol{z}))$ follows from Bayes' rule and is given by

$$\begin{aligned} P\{C|\boldsymbol{z}\} &= \frac{\mathrm{e}^{\mathrm{llr}(\boldsymbol{z})+\rho}}{1+\mathrm{e}^{\mathrm{llr}(\boldsymbol{z})+\rho}} \\ &= \sigma(\mathrm{llr}(\boldsymbol{z})+\rho), \end{aligned} \tag{2}$$

with $\sigma(x) = \frac{\mathrm{e}^x}{1+\mathrm{e}^x}$ the sigmoid activation, or logistic, function and

$$\rho = \log \frac{P\{C\}}{1-P\{C\}} \tag{3}$$

the log prior odds. Equation (2) shows how a prior-independent log-likelihood ratio classifier can used to predict the posterior class probability by feeding the log-likelihood ratio into a sigmoid that, in neural network terms, has the log prior odds ρ as a bias term. This result is also presented in [7, Equation (2)]

4 Conditions for Neyman-Pearson optimality of neural nets

Our starting point is a binary classifier with input $\boldsymbol{z}$, implemented as a neural net with a predefined topology. The weights are ordered in a vector $\boldsymbol{w} \in \mathcal{W} \subset \mathbb{R}^m$, with $\mathcal{W}$ the set of allowable weight vectors. This classifier has a linear output layer with a single output neuron that computes a score $s(\boldsymbol{z}|\boldsymbol{w})$ that can be thresholded to predict whether or not $\boldsymbol{z}$ belongs to class C. It is important to note that because of the structure of neural nets, there may be multiple weight vectors $\boldsymbol{w}$ that realise the same score function $s(\boldsymbol{z}|\boldsymbol{w})$. For instance, many networks are invariant to certain permutations of weights and, if ReLU activation functions are used, scaling weights in one layer can be compensated for in the next.

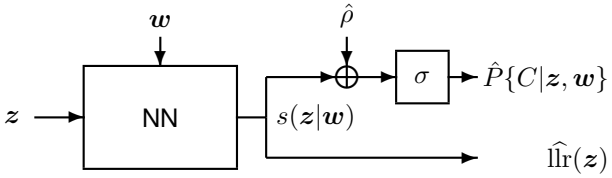

Fig. 1. Configuration to train the network NN as a likelihood ratio classifier. NN has a linear output layer with a single output neuron that computes a score $s(\boldsymbol{z}|\boldsymbol{w})$. The sigmoid activation function σ predicts the posterior class probability $\hat{P}\{C|\boldsymbol{z},\boldsymbol{w}\}$, which is used to train the weights $\boldsymbol{w}$ of the network NN, such that $s(\boldsymbol{z}|\boldsymbol{w})$ converges to a predictor $\widehat{\mathrm{llr}}(\boldsymbol{z})$ of the log-likelihood ratio. The parameter $\hat{\rho}$, defined in (6), is an estimator for the log-prior odds.

For training is available a training set $\mathcal{X}$ consisting of a subset $\mathcal{X}^+$ with N^+ positive input samples $\boldsymbol{z}$ belonging to class C and a subset $\mathcal{X}^-$ with N^- negative input samples $\boldsymbol{z}$ belonging to the alternative class $\overline{C}$. In total there are $N^+ + N^- = N = |\mathcal{X}|$ training samples. We define

$$\hat{p}^+ \stackrel{\mathrm{def}}{=} \frac{N^+}{N},\ \hat{p}^- \stackrel{\mathrm{def}}{=} \frac{N^-}{N}, \tag{4}$$

as the empirical priors of the classes C and $\overline{C}$, respectively.

Inspired by (2) we terminate the network by extending it with a sigmoid activation function that outputs a prediction

$$\hat{P}\{C|\boldsymbol{z},\boldsymbol{w}\} = \sigma(s(\boldsymbol{z}|\boldsymbol{w}) + \hat{\rho}), \tag{5}$$

of the posterior probability that $\boldsymbol{z}$ is in class C. Here

$$\hat{\rho} \stackrel{\mathrm{def}}{=} \log \frac{\hat{p}^+}{\hat{p}^-} \tag{6}$$

is the bias term that estimates the log-prior odds.

The idea is to train the extended network as a predictor $\hat{P}\{C|\boldsymbol{z},\boldsymbol{w}\}$ for the posterior probability with a suitable loss function such that the score function $s(\boldsymbol{z}|\boldsymbol{w})$ will converge to the likelihood ratio. After training, the network without the extension is then a prior-independent, Neyman-Pearson optimal binary classifier. This approach avoids the definition of a loss function that ensures Neyman-Pearson optimality while operating directly on $s(\boldsymbol{z}|\boldsymbol{w})$. The configuration for training a network NN as a likelihood ratio classifier is shown in Figure 1.

The training of the network requires a loss function $L(\boldsymbol{w},\mathcal{X})$ that quantifies the classification performance of the network extended with the sigmoid in (5) on the training set $\mathcal{X}$ as a function of the weights $\boldsymbol{w}$ and an optimisation procedure that can derive the optimal weights as

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} L(\boldsymbol{w},\mathcal{X}). \tag{7}$$

The question at hand is:

Can we train the weights $\boldsymbol{w}$ of the network such that $s(\boldsymbol{z}|\boldsymbol{w}^*)$ approximates the log-likelihood ratio $\mathrm{llr}(\boldsymbol{z})$?

For a positive answer to this question, the following two conditions must be satisfied:

Condition 1 [Richness]: The topology and richness of the network must be such that with proper weights it can realise the log-likelihood ratio. This means that

$$\exists_{\tilde{\boldsymbol{w}}\in\mathcal{W}}\forall_{\boldsymbol{z}\in\mathcal{Z}} : s(\boldsymbol{z}|\tilde{\boldsymbol{w}}) = \mathrm{llr}(\boldsymbol{z}). \tag{8}$$

Here we assume that this condition is satisfied. In practice, this condition may be too strong and it would hopefully be sufficient to have $s(\boldsymbol{z}|\tilde{\boldsymbol{w}}) \simeq \mathrm{llr}(\boldsymbol{z})$, which is reasonable to assume since neural nets with ReLU activation functions are known to be good function approximators [13], [15], [16], [17], [18]. In Section 7 we will show by means of examples to what extent the approximation is successful.

Condition 2 [Loss function]: The loss function $L(\boldsymbol{w}, \mathcal{X})$ must be such that the minimisation in (7) indeed results in weights that realise the log-likelihood ratio as a score function. This means that

$$\forall_{\boldsymbol{w}^*=\arg\min_{\boldsymbol{w}} L(\boldsymbol{w},\mathcal{X})} : s(\boldsymbol{z}|\boldsymbol{w}^*) = \mathrm{llr}(\boldsymbol{z}). \tag{9}$$

Note that the mere existence of a weight vector $\tilde{\boldsymbol{w}} \in \mathcal{W}$ as in (8) does not guarantee convergence of training to this weight vector and that a condition $\boldsymbol{w}^* = \tilde{\boldsymbol{w}}$ would be too restrictive since multiple weight vectors may realise the same score function. Condition 2 will be further discussed in the following sections.

5 Proof of Neyman-Pearson optimality

In this section we prove that with proper training a sufficiently rich binary neural net classifier can achieve Neyman-Pearson optimality. We assume that Condition 1 is satisfied and show that the binary cross-entropy (BCE) loss function [3] given by

$$L_{\mathrm{BCE}}(\boldsymbol{w}, \mathcal{X}) = \tag{10}$$
$$-\left(\hat{p}^+\frac{1}{N^+}\sum_{\boldsymbol{z}\in\mathcal{X}^+}\log(\sigma) + \hat{p}^-\frac{1}{N^-}\sum_{\boldsymbol{z}\in\mathcal{X}^-}\log(1-\sigma)\right),$$

with σ short for $\sigma(s(\boldsymbol{z}|\boldsymbol{w}) + \hat{\rho})$, satisfies Condition 2 above.

We assume that the training set $\mathcal{X}$ is a draw from statistically independent random samples $\underline{\boldsymbol{z}}$ with probability densities $p(\boldsymbol{z}|C)$ for samples from class C and $p(\boldsymbol{z}|\overline{C})$ for those from class $\overline{C}$; hence its notation as a random variable $\underline{\mathcal{X}}$. Because of the central limit theorem, for large N^+ and N^- the two averages in the right-hand side of (10) will converge to their expectations. Therefore, for large N^+ and N^-, we have

$$\begin{aligned} L_{\mathrm{BCE}}(\boldsymbol{w}, \underline{\mathcal{X}}) &= -\big(\hat{p}^+E_{\underline{\boldsymbol{z}}|C}\{\log(\sigma(s(\underline{\boldsymbol{z}}|\boldsymbol{w}) + \hat{\rho}))\} \\ &\quad + \hat{p}^-E_{\underline{\boldsymbol{z}}|\overline{C}}\{\log(1-\sigma(s(\underline{\boldsymbol{z}}|\boldsymbol{w}) + \hat{\rho}))\}\big) \\ &\quad + \underline{e}_N(\boldsymbol{w}) \\ &\stackrel{\mathrm{def}}{=} \overline{\mu}(\boldsymbol{w}) + \underline{e}_N(\boldsymbol{w}), \end{aligned} \tag{11}$$

with $\overline{\mu}(\boldsymbol{w})$ a convenient shorthand,

$$\underline{e}_N(\boldsymbol{w}) \sim N(0, \frac{1}{N}\overline{\mathrm{var}}(\boldsymbol{w})), \tag{12}$$

and

$$\begin{aligned} \overline{\mathrm{var}}(\boldsymbol{w}) &= \hat{p}^+\mathrm{var}_{\underline{\boldsymbol{z}}|C}\{\log(\sigma(s(\underline{\boldsymbol{z}}|\boldsymbol{w}) + \hat{\rho}))\} + \\ &\quad \hat{p}^-\mathrm{var}_{\underline{\boldsymbol{z}}|\overline{C}}\{\log(1-\sigma(s(\underline{\boldsymbol{z}}|\boldsymbol{w}) + \hat{\rho}))\} \end{aligned} \tag{13}$$

another convenient shorthand for averaged variances.

Because of (11), minimising the loss function on a large training set is effectively equivalent to minimising $\overline{\mu}(\boldsymbol{w})$. We prove that for a network that is sufficiently rich (Condition 1), $\overline{\mu}(\boldsymbol{w})$ is minimized by a set of weights $\tilde{\boldsymbol{w}}$, such that $s(\boldsymbol{z}|\tilde{\boldsymbol{w}}) = \mathrm{llr}(\boldsymbol{z})$. For that purpose, we define the functional

$$\begin{aligned} Q(s) &\stackrel{\mathrm{def}}{=} -(\hat{p}^+E_{\underline{\boldsymbol{z}}|C}\{\log(\sigma(s(\underline{\boldsymbol{z}}) + \hat{\rho}))\} \\ &\quad +\hat{p}^-E_{\underline{\boldsymbol{z}}|\overline{C}}\{\log(1-\sigma(s(\underline{\boldsymbol{z}}) + \hat{\rho}))\}) \end{aligned} \tag{14}$$

as a modification of $\overline{\mu}(\boldsymbol{w})$ in that $s(\boldsymbol{z}|\boldsymbol{w})$ has been replaced by a score function $s(\boldsymbol{z})$ that is unconstrained by the network topology. We show that $Q(s)$ attains a global minimum at $s(\boldsymbol{z}) = \mathrm{llr}(\boldsymbol{z})$. The prove uses the calculus of variations. It assumes that $s^*(\boldsymbol{z})$ minimises $Q(s)$ and then analyses the effect of a small perturbation $s(\boldsymbol{z}) = s^*(\boldsymbol{z}) + \gamma\epsilon(\boldsymbol{z})$, with γ a small number and $\epsilon(\boldsymbol{z})$ an arbitrary function. Hence, the final objective function is

$$\tilde{Q}(\gamma) \stackrel{\mathrm{def}}{=} Q(s^* + \gamma\epsilon), \tag{15}$$

which is minimised by solving $s^*(\underline{\boldsymbol{z}})$ from setting $\tilde{Q}'(0) = 0$, with $\tilde{Q}'(\gamma)$ the first derivative of $\tilde{Q}(\gamma)$ w.r.t. γ given by

$$\begin{aligned} \tilde{Q}'(\gamma) &= -\hat{p}^+E_{\underline{\boldsymbol{z}}|C}\{(1-\sigma(s^*(\underline{\boldsymbol{z}}) + \gamma\epsilon(\underline{\boldsymbol{z}}) + \hat{\rho}))\epsilon(\underline{\boldsymbol{z}})\} \\ &\quad +\hat{p}^-E_{\underline{\boldsymbol{z}}|\overline{C}}\{\sigma(s^*(\underline{\boldsymbol{z}}) + \gamma\epsilon(\underline{\boldsymbol{z}}) + \hat{\rho}))\epsilon(\underline{\boldsymbol{z}})\}. \end{aligned} \tag{16}$$

Because of the construction of $\tilde{Q}(\gamma)$ (15) we have that $\tilde{Q}'(0) = 0$ at the minimum, or

$$\begin{aligned} -\hat{p}^+E_{\underline{\boldsymbol{z}}|C}\{(1-\sigma(s^*(\underline{\boldsymbol{z}}) + \hat{\rho}))\epsilon(\underline{\boldsymbol{z}})\} \\ +\hat{p}^-E_{\underline{\boldsymbol{z}}|\overline{C}}\{\sigma(s^*(\underline{\boldsymbol{z}}) + \hat{\rho})\epsilon(\underline{\boldsymbol{z}})\} &= 0. \end{aligned} \tag{17}$$

This can be expanded into

$$\begin{aligned} \int_{\boldsymbol{z}} \epsilon(\boldsymbol{z})(-\hat{p}^+\frac{1}{1+\mathrm{e}^{s^*(\boldsymbol{z})+\hat{\rho}}}p(\boldsymbol{z}|C) \\ +\hat{p}^-\frac{\mathrm{e}^{s^*(\boldsymbol{z})+\hat{\rho}}}{1+\mathrm{e}^{s^*(\boldsymbol{z})+\hat{\rho}}}p(\boldsymbol{z}|\overline{C}))\mathrm{d}\boldsymbol{z} &= 0, \end{aligned} \tag{18}$$

which must hold for arbitrary $\epsilon(\boldsymbol{z})$. Therefore, it must be that

$$\begin{aligned} -\hat{p}^+\frac{1}{1+\mathrm{e}^{s^*(\boldsymbol{z})+\hat{\rho}}}p(\boldsymbol{z}|C) \\ +\hat{p}^-\frac{\mathrm{e}^{s^*(\boldsymbol{z})+\hat{\rho}}}{1+\mathrm{e}^{s^*(\boldsymbol{z})+\hat{\rho}}}p(\boldsymbol{z}|\overline{C}) &= 0, \end{aligned} \tag{19}$$

from which, by using using (6), we derive

$$s^*(\boldsymbol{z}) = \log\frac{p(\boldsymbol{z}|C)}{p(\boldsymbol{z}|\overline{C})} = \mathrm{llr}(\boldsymbol{z}), \tag{20}$$

which proves the result.

In order to present an expression for the second derivative of $\tilde{Q}(\gamma)$ w.r.t. γ, note that

$$p(\boldsymbol{z}) = \hat{p}^+p(\boldsymbol{z}|C) + \hat{p}^-p(\boldsymbol{z}|\overline{C}), \tag{21}$$

is the probability density function of $\boldsymbol{z}$ as represented in the training data. We define

$$\begin{aligned} \psi(\gamma, \boldsymbol{z}) &\stackrel{\mathrm{def}}{=} \\ &\sigma(\mathrm{llr}(\boldsymbol{z}) + \gamma\epsilon(\boldsymbol{z}) + \hat{\rho})(1-\sigma(\mathrm{llr}(\boldsymbol{z}) + \gamma\epsilon(\boldsymbol{z}) + \hat{\rho})). \end{aligned} \tag{22}$$

The second derivative of $\tilde{Q}(\gamma)$ w.r.t. γ is then given by

$$\tilde{Q}''(\gamma) = \int_{\boldsymbol{z}} \epsilon^2(\boldsymbol{z})\psi(\gamma, \boldsymbol{z})p(\boldsymbol{z})\mathrm{d}\boldsymbol{z}, \tag{23}$$

which is positive for all γ. Therefore, $Q(s)$ attains a global minimum at $s(\boldsymbol{z}) = \mathrm{llr}(\boldsymbol{z})$.

Because of Condition 1, a $\tilde{\boldsymbol{w}}$ exists, such that $s(\boldsymbol{z}|\tilde{\boldsymbol{w}}) = \mathrm{llr}(\boldsymbol{z})$. This $\tilde{\boldsymbol{w}}$ globally minimises the first term of right-hand side of (11). For $N \to \infty$, $\tilde{\boldsymbol{w}}$ then also minimises $L_{\mathrm{BCE}}(\boldsymbol{w}, \underline{\mathcal{X}})$. This, however, does not guarantee that training will converge to a specific $\tilde{\boldsymbol{w}}$, since multiple weight vectors $\boldsymbol{w}$ may realise the same score function. However, $Q(s)$ attains a global minimum at $s(\boldsymbol{z}) = \mathrm{llr}(\boldsymbol{z})$. Therefore, the set of weights $\boldsymbol{w}^*$ that result after convergence will realise the log-likelihood ratio, or

$$\begin{aligned} \boldsymbol{w}^* &= \arg\min_{\boldsymbol{w}} \lim_{N\to\infty} L_{\mathrm{BCE}}(\boldsymbol{w}, \mathcal{X}) \\ &\Rightarrow \lim_{N\to\infty} s(\boldsymbol{z}|\boldsymbol{w}^*) = \mathrm{llr}(\boldsymbol{z}). \end{aligned} \tag{24}$$

This proves that a sufficiently rich neural net trained with a binary cross-entropy loss function can compute a log-likelihood ratio and hence realise a Neyman-Person optimal classifier.

In practice, neural nets are iteratively trained by stochastic gradient descent methods based on batches, where each batch is drawn from a much larger training set. In each iteration $\boldsymbol{w}$ is updated. Then, the loss function that is minimised is based on batches instead of on the entire training set and $\mathcal{X}$, $\underline{\mathcal{X}}$, and N in (10) and (11) must be taken as batches and their size, respectively.

6 Properties of llr approximation

After the training has been terminated because the loss function has converged to approximate its minimum, the classification performance of the network extended with a sigmoid function, predicting $P\{C|\boldsymbol{z}\}$, will be (close to) optimal depending on the networks capabilities. In this section we study the sensitivity of the loss function to variations of

$$\widehat{\mathrm{llr}}(\boldsymbol{z}) \stackrel{\mathrm{def}}{=} s(\boldsymbol{z}|\boldsymbol{w}^*) \tag{25}$$

around the optimum $\mathrm{llr}(\boldsymbol{z})$ for large N in order to analyse possible deviations of $\widehat{\mathrm{llr}}(\boldsymbol{z})$ from $\mathrm{llr}(\boldsymbol{z})$. The deviation is modelled as $\widehat{\mathrm{llr}}(\boldsymbol{z}) = \mathrm{llr}(\boldsymbol{z}) + \gamma\epsilon(\boldsymbol{z})$, with γ a small number expressing the strength of the deviation and $\epsilon(\boldsymbol{z})$, $\int_{\boldsymbol{z}} \epsilon^2(\boldsymbol{z})\mathrm{d}\boldsymbol{z} = 1$, an arbitrary function expressing its shape. For N large enough, $|\gamma|$ sufficiently small, and using that $\tilde{Q}'(0) = 0$, we approximate the increase of the loss function in the direction of the deviation $\epsilon(\boldsymbol{z})$ by its Taylor expansion at the minimum by

$$L_{\mathrm{BCE}}(\boldsymbol{w}, \underline{\mathcal{X}}) \simeq Q_{\min} + \frac{1}{2}\tilde{Q}''(0)\gamma^2 + \underline{e}_N(\boldsymbol{w}), \tag{26}$$

with $Q_{\min}$ the minimum of $Q(s)$, attained at $s(\boldsymbol{z}|\boldsymbol{w}^*) = \mathrm{llr}(\boldsymbol{z})$. From (23) it follows that

$$\tilde{Q}''(0) = \int_{\boldsymbol{z}} \epsilon^2(\boldsymbol{z})\psi(0, \boldsymbol{z})p(\boldsymbol{z})\mathrm{d}\boldsymbol{z}, \tag{27}$$

with $\psi(0, \boldsymbol{z}) = \sigma(\mathrm{llr}(\boldsymbol{z}) + \hat{\rho})(1 - \sigma(\mathrm{llr}(\boldsymbol{z}) + \hat{\rho}))$, (22), which is a convenient expression to analyse the convergence of the binary cross-entropy loss close to the optimum. In general, if $\tilde{Q}''(0)$ becomes small for a certain $\epsilon(\boldsymbol{z})$, a deviation of $\mathrm{llr}(\boldsymbol{z})$ in the direction of such $\epsilon(\boldsymbol{z})$ will have very little impact on the minimisation of $Q(s)$ and may prevent further convergence towards $\mathrm{llr}(\boldsymbol{z})$, due to secondary effects caused by the term $\underline{e}_N(\boldsymbol{w})$.

For the further analysis of the approximation of the llr approximation here and in Section 7, it is convenient to model the deviations $\epsilon(\boldsymbol{z})$ as a sum

$$\epsilon(\boldsymbol{z}) = \epsilon_0(\mathrm{llr}(\boldsymbol{z})) + \underline{\eta}, \tag{28}$$

with $\epsilon_0(\mathrm{llr}(\boldsymbol{z}))$ only dependent on the llr and $\underline{\eta}$ a zero-mean random component with an llr-conditional probability density function $q(\eta|\mathrm{llr})$. The term $\epsilon_0(\mathrm{llr})$ is a bias in the predicted llr; the term $\underline{\eta}$ is a noise component, dependent on the llr. The simulation results in Figures 5 and 10 in Section 7 illustrate that this is a plausible model. Because $\psi(0, \boldsymbol{z})$ only depends on llr, we can change variables in the integral in (27) for $\tilde{Q}''(0)$ and rewrite this as

$$\begin{aligned} \tilde{Q}''(0) \;=\; & \int_{llr} \epsilon_0^2(llr)\psi(0, llr)p(llr)\mathrm{d}llr + \\ & \int_{llr} \sigma^2_{\eta|llr}\psi(0, llr)p(llr)\mathrm{d}llr, \end{aligned} \tag{29}$$

with $\sigma^2_{\eta|\mathrm{llr}}$ the conditional variance of the noise and $p(\mathrm{llr})$ the probability density function reflecting the frequency of occurrence of the ground truth llr in the training data. We see that in (29) $\psi(0, \mathrm{llr})p(\mathrm{llr})$ defines an llr-dependent penalty function

$$\mathrm{pen}(\mathrm{llr}) \stackrel{\mathrm{def}}{=} \psi(0, \mathrm{llr})p(\mathrm{llr}) \tag{30}$$

on both the bias $\epsilon_0(\mathrm{llr})$ and on the noise variance $\sigma^2_{\eta|\mathrm{llr}}$. Wherever this penalty function decreases, the bias and the noise can cause $\widehat{\mathrm{llr}}(\boldsymbol{z})$ to deviate form $\mathrm{llr}(\boldsymbol{z})$. Given the binary cross-entropy loss function (26), this can only be mitigated by increasing N, which will reduce the impact of $\underline{e}_N(\boldsymbol{w})$ on the minimisation. The penalty function (30) can be made more insightful as follows. Let $p(\mathrm{llr}|C)$ and $p(\mathrm{llr}|\overline{C})$ denote the class-conditional probability density functions reflecting the frequency of occurrence of the ground truth llr in the training data, then

$$p(\mathrm{llr}) = \hat{p}^+ p(\mathrm{llr}|C) + \hat{p}^- p(\mathrm{llr}|\overline{C}), \tag{31}$$

and we can derive that

$$\mathrm{pen}(\mathrm{llr}) = \frac{\hat{p}^+\hat{p}^- p(\mathrm{llr}|C)p(\mathrm{llr}|\overline{C})}{\hat{p}^+ p(\mathrm{llr}|C) + \hat{p}^- p(\mathrm{llr}|\overline{C})}. \tag{32}$$

Below we use (32) to present three properties of llr approximation.

Property 1 – Approximation at high and low log-likelihood ratios: If $\mathrm{llr} \lll 0$ then $p(\mathrm{llr}|C) \lll p(\mathrm{llr}|\overline{C})$ and the penalty function $\mathrm{pen}(\mathrm{llr}) \simeq 0$. Then bias and noise are not penalised and may cause the predicted log-likelihood ratios to deviate from true ones. Since $p(\mathrm{llr}|C)$ and $p(\mathrm{llr}|\overline{C})$ are unimodal, $\mathrm{pen}(\mathrm{llr})$ attains its maximum somewhere in between their maxima, hence at moderate log-likelihood ratio values. The poorer approximation at lower and higher values of the log-likelihood ratio may not be a serious problem from a classification point of view, since there the classifier will almost always make correct decisions. In conclusion: *Approximation of the* llr *is best for moderate*

values of the log-likelihood ratio and will degrade if its magnitudes increases.

The approximation at high and low log-likelihood ratios is illustrated in Figure 2 by the blue line for a 1-dimensional toy example. In this example, $z|C \sim N(2,1)$ and $z|\overline{C} \sim N(-2,1)$, and $p^+ = p^- = 0.5$. Here, $\mathrm{llr}(z) = 4z$. The blue line is the penalty function $\mathrm{pen}(\mathrm{llr})$, (30). Here we see that the approximation is best for $\mathrm{llr} = 0$ and becomes rapidly worse if the llr moves away from 0.

Property 2 – Approximation with unbalanced priors: If $\hat{p}^+ < \hat{p}^-$, the penalty function $\mathrm{pen}(\mathrm{llr})$ will shift towards $p(\mathrm{llr}|C)$, and If $\hat{p}^+ > \hat{p}^-$, the penalty function $\mathrm{pen}(\mathrm{llr})$ will shift towards $p(\mathrm{llr}|\overline{C})$. That means that the prediction of the log-likelihood ratio may become less accurate at $\mathrm{llr} = 0$. If, even stronger, $\hat{p}^+ \lll \hat{p}^-$, then the penalty function $\mathrm{pen}(\mathrm{llr}) \simeq 0$. That means that accuracy of the approximation will globally decrease. Hence, during operation the network can be adapted to unbalanced priors by varying ρ, but during training it is recommended to keep the priors more or less balanced. In conclusion: *Unbalanced priors will shift the point of best approximation towards the class with the lowest prior and overall approximation will degrade.*

Approximation with unbalanced priors is illustrated in Figure 2 by the red line for a similar 1-dimensional toy example as used to illustrate the previous approximation property. Here, $z|C \sim N(2,1)$ and $z|\overline{C} \sim N(-2,1)$, and $p^+ = 0.1$, $p^- = 0.9$. The red line is the penalty function $\mathrm{pen}(\mathrm{llr})$, which has shifted and decreased in magnitude in comparison to the blue line and location of best approximation has shifted from $\mathrm{llr} = 0$ to $\mathrm{llr} \simeq 2$.

Property 3 – Approximation with well-separated classes: If the class-conditional probability density functions $p(\mathrm{llr}|C)$ and $p(\mathrm{llr}|\overline{C})$ are well-separated, then the penalty function $\mathrm{pen}(\mathrm{llr}) \simeq 0$ in the range of llr where $p(\mathrm{llr}|C) \simeq p(\mathrm{llr}|\overline{C})$. That means that accuracy of the approximation will globally decrease. In this case, this may also not be problematic from a classification point of view, since it is easily understood intuitively that for well-separated classes the choice of the score function is not so critical. In conclusion: *Well-separated classes exhibit poorer approximation of the* llr *than classes with some overlap.*

Approximation with well-separated classes is illustrated in Figure 2 by the green line for a similar 1-dimensional toy example as used before, but with class-conditional expectations that are further separated. Here, $z|C \sim N(3,1)$ and $z|\overline{C} \sim N(-3,1)$, and $p^+ = p^- = 0.5$. Here $\mathrm{llr}(z) = 6z$. The green line is the penalty function $\mathrm{pen}(\mathrm{llr})$. We see that the approximation is best for $\mathrm{llr} = 0$ but overall it seems 2–3 orders of magnitude worse than for the previous examples.

Figure 2 shows how deviations on the likelihood ratio are penalised during training. It is remarkable that the highest magnitude of a penalty function not more than about 10^{-2}. This suggests a slow and an overall poor convergence of the approximation. In Section 7 we will train neural nets with synthetic data with known log-likelihood ratios and compare those to the ones predicted by the network, in order to get a clearer picture of the quality of the approximation.

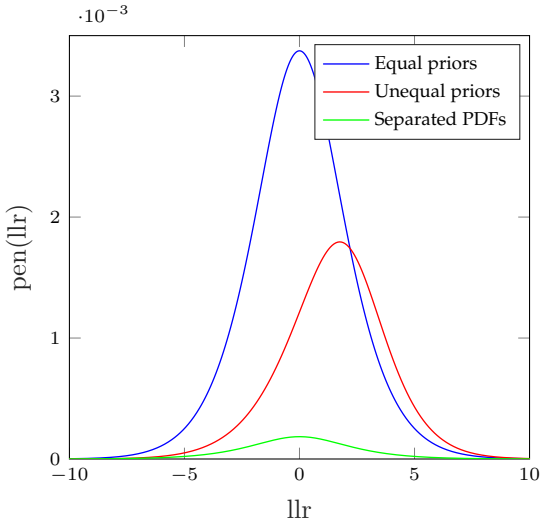

Fig. 2. Comparison of llr approximation for three toy examples. The penalty function $\mathrm{pen}(\mathrm{llr})$ is plotted as a function of the log-likelihood ratio llr for three cases: (1) Equal priors: $z|C \sim N(2,1)$ and $z|\overline{C} \sim N(-2,1)$, and $p^+ = p^- = 0.5$ (blue). The approximation is best at moderate log-likelihood ratios. (2) Unequal priors: $z|C \sim N(2,1)$ and $z|\overline{C} \sim N(-2,1)$, and $p^+ = 0.1$, $p^- = 0.9$. (red). The point of best approximation shifts towards the underrepresented class and overall approximation decreases. (3) Well-separated class-conditional probability density functions (PDFs): $z|C \sim N(3,1)$ and $z|\overline{C} \sim N(-3,1)$, and $p^+ = p^- = 0.5$ (green). The overal approximation has decreased by 2–3 orders of magnitude.

7 Illustrative Examples

In this section we present two examples to illustrate the convergence of the approximation of the true log-likelihood ratio as a function of the amount of training data and the dimensionality of the input vector $\boldsymbol{z}$. This is done with synthetically generated data, drawn from known probability densities, because only then we can have ground truth log-likelihood ratios. The first example, in Subsection 7.1, uses non-Gaussian 2-dimensional data and is more challenging than the one-dimensional toy examples in Section 6 since it requires a non-linear classifier to separate the classes. The second example, in Subsection 7.2, simulates the biometric comparison problem where it has to be decided whether two biometric samples originate from the same individual.

We will present the results in four types of graphs. The first type plots the binary-cross-entropy loss defined in (10) and the root mean-square approximation error (RMSE) E_{T}, both obtained on validation data as function of the number of training epochs. The root mean-square approximation error E_{T} is defined as

$$E_{\mathrm{T}} \stackrel{\mathrm{def}}{=} \sqrt{\sum_{\boldsymbol{z} \in \mathcal{Z}} |\widehat{\mathrm{llr}}(\boldsymbol{z}) - \mathrm{llr}(\boldsymbol{z})|^2}, \tag{33}$$

with $\mathcal{Z}$ the validation or the test set. In practical applications with real data, the true log-likelihood ratio is not available and the root mean-square approximation error cannot be used to monitor and terminate the training process. Here we include root mean-square approximation error plots to illustrate the convergence behaviour of the training. In addition, we will also compute the root mean-square approximation

error E_0, which expresses the quality of the approximation around zero by

$$E_0 \stackrel{\text{def}}{=} \sqrt{\sum_{\boldsymbol{z}\in\mathcal{Z}_0(M)} |\widehat{\text{llr}}(\boldsymbol{z}) - \text{llr}(\boldsymbol{z})|^2}, \tag{34}$$

with $\mathcal{Z}_0(M) \subset \mathcal{Z}$ the M samples in the test set for which the median llr is closest to 0. In our examples we will take $M = 100$. The second type of graph that we will present plots the predicted log-likelihood ratio $\widehat{\text{llr}}$ as a function of the true log-likelihood ratio llr. The third type of graph plots the penalty function $\text{pen}(\text{llr})$, as shown in Figure 2, which is used to explain the convergence of llr approximation as shown in the first two type of graphs. The penalty function of the examples presented in this section is harder, though not impossible, to express analytically. Therefore, we derive it empirically from the ground truth llr that is available with the training data. This is done as follows. The penalty function (30) is rewritten as

$$\text{pen}(\text{llr}) = \frac{\frac{p^+}{p^-}\text{e}^{\text{llr}}}{(\frac{p^+}{p^-}\text{e}^{\text{llr}}+1)^2} p(\text{llr}). \tag{35}$$

As said, llr is available in the training data and can be used to estimate $p(\text{llr})$ via a histogram. Finally, the fourth type of graph shows the the trade-off between the false-positive rate (FPR) and the false-negative rate (FNR) as a function of varying thresholds for $\widehat{\text{llr}}$ in a Decision Error Trade-off (DET) plot. This graph will also show the DET plot for the true log-likelihood ratio llr.

The following settings are valid for both examples. Training and test samples will be drawn from known class-conditional probability density functions. In every example we will use balanced training and test sets with equal numbers of samples drawn from both classes. The numbers of training samples will be chosen as $K \in \{0.1\text{M}, 0.4\text{M}, 1.6\text{M}, 6.4\text{M}\}$, with $K^+ = K^- = K/2$ the numbers of samples per class. The validation sets and test will contain 10000 samples, 5000 from each class. Prior to every training epoch a random selection of the training data will extracted as a validation set and the remaining training samples will be randomly ordered into balanced batches with a batch size $N = 1000$.

With every training set a fully connected neural net classifier NN will be trained, with a number of input neurons corresponding to the input samples, one hidden layer with a varying number of hidden neurons depending on the example and with a ReLU activation function, and a linear output layer with one output neuron to predict the log-likelihood ratio $\widehat{\text{llr}}$. In the biometric experiments (Example 2), batch normalisation and dropout ($\text{rate} = 0.3$) will be applied after the hidden layer. The configurations of the networks in both examples are summarised in Table 1. During training the output will be extended with a sigmoid function as in shown in Figure 1 with $\hat{\rho} = 0$ in order to predict the posterior probabilities needed for the binary cross-entropy loss defined in (10).

The binary cross-entropy loss will be minimised using the Adam optimiser [25] with an initial learning rate of 1.0×10^{-3}. In the biometric experiments (Example 2), the learning rate will be reduced to 0.5×10^{-3} during training. After each epoch the binary-cross-entropy loss defined in (10) and the root mean-square approximation error E_{T} defined in (33) will be computed on a validation set. After training the llr prediction and the the recognition performance of the network will tested on an independent test set drawn from the same distributions.

TABLE 1
Configurations of the neural nets NN in Example 1 (Figure 1) and Example 2 (Figure 8).

Layers	Example 1	Example 2
input	2	200
hidden	64 ReLU – –	2048 ReLU batch normalisation dropout, $\text{rate} = 0.3$
output	1 linear	1 linear

7.1 Example 1: Non-Gaussian class-conditional probability density functions.

In this 2-dimenssional example, we represent the class-conditional probability densities as functions of $\boldsymbol{z}$ represented in polar coordinates (r, θ):

$$p(r, \theta|C) = \beta(r, a_1, b_1) r \frac{1}{2\pi}, \tag{36}$$

$$p(r, \theta|\overline{C}) = \beta(r, a_0, b_0) r \frac{1}{2\pi}, \tag{37}$$

with $\beta(r, a, b)$ the beta-probability density with parameters a and b [24]. In this example $(a_1, b_1) = (5, 15)$, and $(a_0, b_0) = (10, 5)$. Scatter plots of samples drawn from these class-conditional probability density functions are shown in Figure 3. The samples in C and $\overline{C}$ are blue and red, respectively. An expression for the ground truth log-likelihood ratio is

$$\begin{aligned} \text{llr}(\boldsymbol{z}) &= \log \frac{B(a_0, b_0)}{B(a_1, b_1)} \\ &\quad + (a_1 - a_0)\log(|\boldsymbol{z}|) + (b_1 - b_0)\log(1 - |\boldsymbol{z}|), \end{aligned} \tag{38}$$

with $|\boldsymbol{z}| \leq 1$ the L2 norm of $\boldsymbol{z}$, and $B(a, b)$ the beta function [26].

The training and test sets consist of xy-coordinate pairs that are drawn from $p(r, \theta|C)$ and $p(r, \theta|\overline{C})$ and converted to Cartesian coordinates. Four training sets with sizes $K = 0.1\text{M}, 0.4\text{M}, 1.6\text{M}, 6.4\text{M}$ are used to train the network in the configuration shown in Figure 1. The configuration of the neural net is summarised in Table 1. The training is performed over 70 epochs. The reason for a fixed number of epochs, rather than using a stop criterion such as a non-decreasing validation loss is that the binary cross-entropy loss function evaluates classification performance rather than llr approximation performance. The latter is only indirectly reflected in the loss function since the true llr is generally not available for training.

Figure 4 shows how the validation binary cross-entropy loss defined in (10) and the root mean-square approximation error (33) converge for the four different sizes K of training sets. Note that, in terms of epochs, the training converges faster if the training set is larger. This is because with a fixed

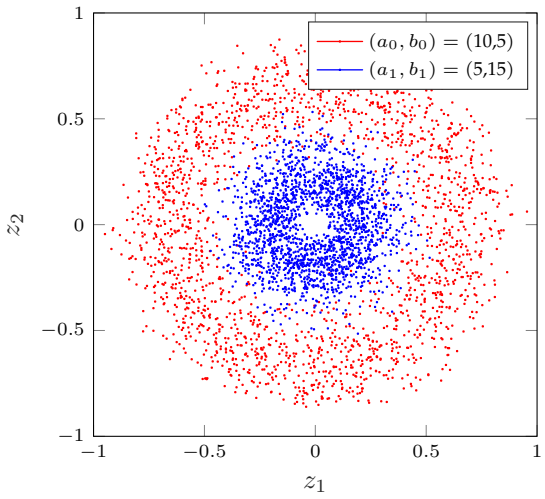

Fig. 3. Scatter plots of the samples used to train the classifier in Example 1: Non-Gaussian class-conditional probability density functions, for different numbers of training samples. Samples of class C (blue) and $\overline{C}$ (red) have 2D probability density functions (in polar coordinates (r, θ)) $\beta(a_1, b_1, r) r \frac{1}{2\pi}$ and $\beta(a_0, b_0, r) r \frac{1}{2\pi}$, respectively, with the parameters specified in the legend.

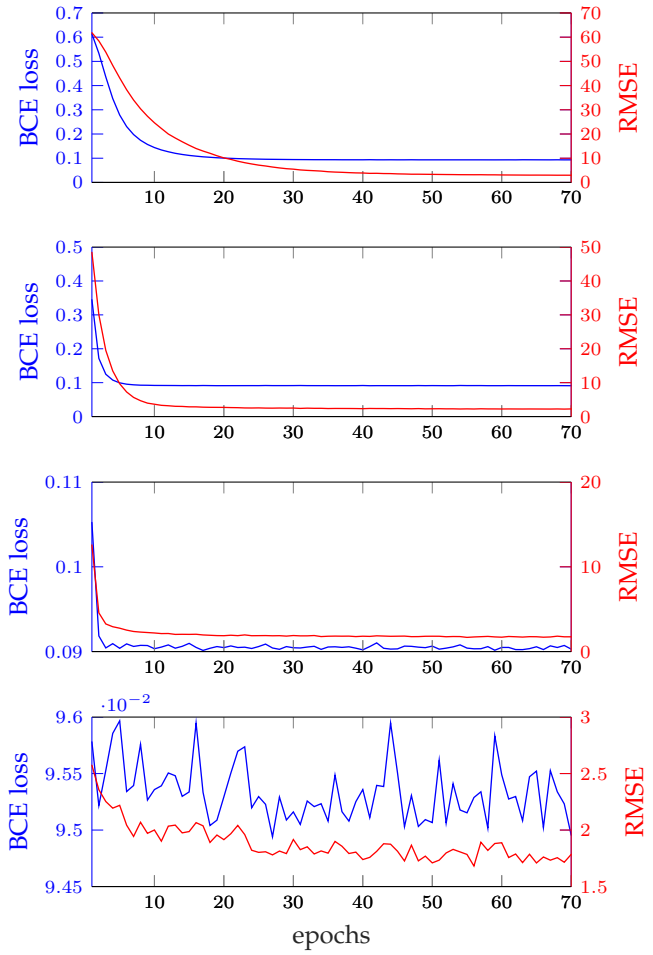

Fig. 4. Convergence of validation binary cross-entropy loss (10) on left y-axis and root mean-square approximation error (33) on right y-axis during training for Example 1: Non-Gaussian class-conditional probability density functions, for different numbers of training samples K. Panels from top to bottom: $K = 0.1\text{M}, 0.4\text{M}, 1.6\text{M}, 6.4M$.

batch size, the larger training sets will have more iterations per epoch.

In Figure 5 scatter plots showing the relation between the true log-likelihood ratio llr (38) and the predicted $\widehat{\mathrm{llr}}$ are shown for the four different sizes K of training sets. In the legend the root mean square approximation errors E_0 (34) of the $M = 100$ samples about $\mathrm{llr} = 0$ are also presented. The penalty function as a function of the ground truth llr, is shown in Figure 6 (blue graph). Figure 7 plots the trade-off between the false-positive rate (FPR) and the false-negative rate (FNR) as a function of varying thresholds for $\widehat{\mathrm{llr}}$ in a Decision Error Trade-off (DET) plot obtained from the training sets for different numbers of training samples K, specified in the legend. The black line in Figure 7 is the DET plot obtained when the true log-likelihood ratios of the test samples are used for classification.

As a first observation we note that the binary cross-entropy loss converges faster than the root mean-square approximation error (Figure 4). We assume that this is because the binary cross-entropy loss close to the minimum is a flat function and characterised by a low penalty function $\mathrm{pen}(\mathrm{llr})$, (30), that allows for substantial deviations of $\widehat{\mathrm{llr}}$, which will make convergence of llr approximation slow. The penalty function for this example is the blue line in Figure 6.

As a second observation we note that $\widehat{\mathrm{llr}}$ approximates llr for $-10 < \mathrm{llr} < 5$ for $K \geq 0.4\text{M}$, but that the relation between llr and $\widehat{\mathrm{llr}}$ is noisy in the sense that the predictions of a log-likelihood ratio seem to exhibit a certain randomness. We assume that this is because in a d-dimensional feature space the true log-likelihood ratios are constant on $(d-1)$-dimensional manifolds. In this simple example the manifold is a circle. The approximations, however, are not necessarily constant on the same manifolds, which leads to multiple predictions for the same llr. We observe that for increasing K, the predictions become less noisy. The penalty function in Figure 6 illustrates that away from $\mathrm{llr} = 0$ there is indeed a quickly decreasing penalty on deviations of $\widehat{\mathrm{llr}}$ from llr. This means that a good classification performance, characterised by the binary cross-entropy loss can be reached long before the llr approximation has converged. The scatterplots in Figure 5 also confirm that the approximation away from $\mathrm{llr} = 0$ degrades. In fact, the root mean-square approximation error E_0 about $\mathrm{llr} = 0$ is almost an order of magnitude below the global root mean-square approximation error E_T reached at 40 epochs, due to the deviations away from $\mathrm{llr} = 0$.

As a third observation we note that the approximations improve with increasing K and that, fully in line with the conclusion presented under *Property 1* in Section 6, the approximation is best at moderate values of llr around 0 and decreases with increasing $|\mathrm{llr}|$. We see that E_0 decreases with increasing K, but that differences between E_0 values become small for larger K while for larger $|\mathrm{llr}|$ the differences in approximation errors are still significant.

As a fourth observation we note that for the relatively simple problem of the example at hand, quite a large training set ($K > 1.6\text{M}$) is needed to achieve a good approximation over a longer range of llr values. This may

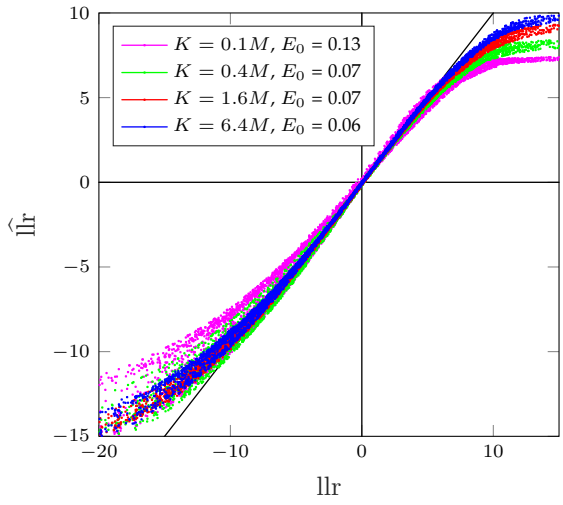

Fig. 5. Scatter plots of the predicted log-likelihood ratio $\widehat{\text{llr}}$ as function of the true log-likelihood ratio llr for Example 1: Non-Gaussian class-conditional probability density functions, for different numbers of training samples K, specified in the legend. The dimensionality of the biometric data is $d = 100$. The legend also provides the E_0 (34) approximation errors with $M = 100$.

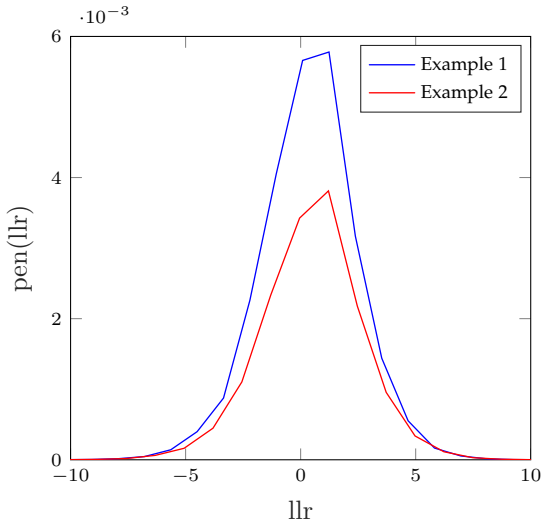

Fig. 6. Estimated penalty functions (30) as a function of ground truth llr, for Example 1 (blue graph) and Example 2 (red graph). The estimates are based on ground truth llrs as present in the training set and histogram approximations of $p(\text{llr})$.

be due to the type of network, but it is interesting to investigate to what extend the predicted $\widehat{\text{llr}}$ can be used as a classification score, even when llr is not yet predicted well. We see that the DET plots of the predicted log-likelihood ratios are practically on top of the DET plot for the true llr. This means that the neural net has achieved (near) optimal recognition performance, even though for $K \leq 0.4\text{M}$ the approximation of the log-likelihood ratio is still noisy and inaccurate near the upper and lower ends. This suggests *that from a classification point of view it still makes sense to apply the proposed training procedure to train a neural net as a (near) Neyman-Pearson optimal classifier, even when the score function $s(\boldsymbol{z}|\boldsymbol{w}^*)$ does not approximate log-likelihood ratio.*

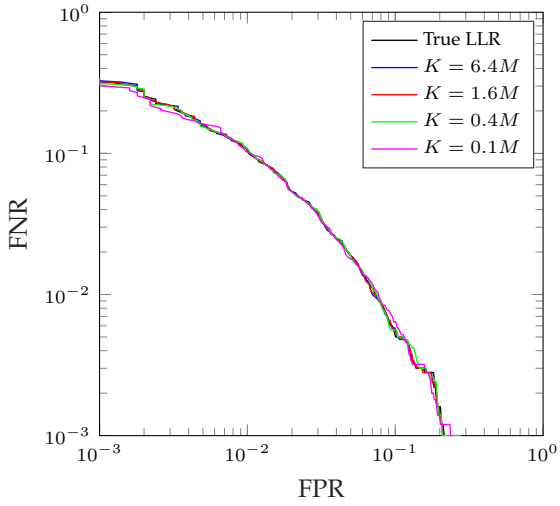

Fig. 7. DET plots of the predicted log-likelihood ratio $\widehat{\text{llr}}$ and of the true log-likelihood ratio llr for Example 1, non Gaussian class-conditional probability density functions, for different numbers of training samples K, specified in the legend.

7.2 Example 2: Simulated biometric comparison.

The goal of biometric comparison is to compare two biometric samples, usually features extracted from biometric data, in order to determine whether or not they originate from the same individual. Sample pairs originating from the same individual are called *mated*. Sample pairs from different individuals are *nonmated*. The common procedure for many types of biometric data is as follows. Let the samples be denoted by vectors $\boldsymbol{x} \in \mathbb{R}^D$ and $\boldsymbol{y} \in \mathbb{R}^D$. Let C denote the class of mated sample pairs $(\boldsymbol{x}, \boldsymbol{y})$, replacing $\boldsymbol{z}$, and $\overline{C}$ the class of nonmated sample pairs $(\boldsymbol{x}, \boldsymbol{y})$. This defines biometric comparison as a binary classification problem. Commonly a similarity score $s(\boldsymbol{x}, \boldsymbol{y})$ is computed that is thresholded in order to decide whether or not the samples are mated. The choice of threshold determines a trade-off between the false-positive rate and the false-negative rate, in standardised biometric terminology [27] known as the false-match rate (FMR) and the false-nonmatch rate (FNMR), respectively. If the biometric comparator is a log-likelihood ratio classifier, then it computes as a similarity score

$$\text{llr}(\boldsymbol{x}, \boldsymbol{y}) = \log \frac{p(\boldsymbol{x}, \boldsymbol{y}|C)}{p(\boldsymbol{x}, \boldsymbol{y}|\overline{C})}. \tag{39}$$

In order to be able to compute ground truth log-likelihood ratios for the data that we generate, we adopt Gaussian class-conditional probability density functions. They follow from an underlying probabilistic model, based on Gaussian between-subject and within-subject variations. Let $\underline{\boldsymbol{b}} \in \mathbb{R}^d$, $\underline{\boldsymbol{b}} \sim N(\boldsymbol{0}, \boldsymbol{\Lambda})$, with $\boldsymbol{\Lambda} \in \mathbb{R}^{d \times d}$ a positive diagonal matrix and $d \leq D$, denote a random vector defined by the identity of the individual providing the biometric sample. Then, varying $\underline{\boldsymbol{b}}$ is varying between identities, and $\boldsymbol{b}$ can be seen as a latent identity variable. That is why $\boldsymbol{\Lambda}$ is called the diagonalised between-subject covariance matrix. Let $\underline{\boldsymbol{\nu}} \in \mathbb{R}^d$, $\underline{\boldsymbol{\nu}} \sim N(\boldsymbol{0}, \boldsymbol{I} - \boldsymbol{\Lambda})$, with $\boldsymbol{I}$ the identity matrix, denote a random vector that models the variations that occur within an identity. Hence, $\boldsymbol{I} - \boldsymbol{\Lambda}$ is called the diagonalised within-

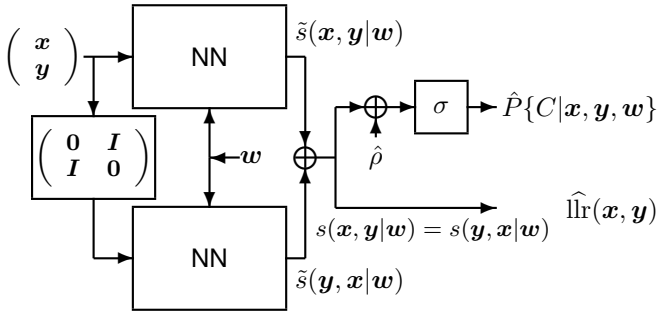

Fig. 8. The configuration of Figure 1 extended such that it is guaranteed to produce a symmetric prediction of the log-likelihood ratio with $\widehat{\mathrm{llr}}(\boldsymbol{x},\boldsymbol{y}) = \widehat{\mathrm{llr}}(\boldsymbol{y},\boldsymbol{x})$.

subject covariance matrix. We assume that all individuals share the same within-class covariance matrix. A random biometric sample $\underline{\boldsymbol{x}}$ is then obtained by first drawing an identity $\underline{\boldsymbol{b}}$ from $N(\boldsymbol{0},\boldsymbol{\Lambda})$ and then drawing a within-class variation $\underline{\boldsymbol{\nu}}$ from $N(\boldsymbol{0},\boldsymbol{I}-\boldsymbol{\Lambda})$ and computing

$$\underline{\boldsymbol{x}} = \boldsymbol{U}_2\boldsymbol{S}\boldsymbol{U}_1\left(\underline{\boldsymbol{b}}+\underline{\boldsymbol{\nu}}\right), \tag{40}$$

with $\boldsymbol{U}_1 \in \mathbb{R}^{d\times d}$ and $\boldsymbol{U}_2 \in \mathbb{R}^{D\times d}$ orthogonal projection matrices and $\boldsymbol{D} \in \mathbb{R}^{d\times d}$ a positive diagonal matrix. A mated pair $(\underline{\boldsymbol{x}},\underline{\boldsymbol{y}})$ is created by applying (40) to draw $\underline{\boldsymbol{x}}$ and $\underline{\boldsymbol{y}}$ with the same $\underline{\boldsymbol{b}} \sim N(\boldsymbol{0},\boldsymbol{\Lambda})$ and different draws for $\underline{\boldsymbol{\nu}}$. A nonmated pair $(\underline{\boldsymbol{x}},\underline{\boldsymbol{y}})$ is created by applying (40) to draw $\underline{\boldsymbol{x}}$ and $\underline{\boldsymbol{y}}$ with different draws for both $\underline{\boldsymbol{b}}$ and $\underline{\boldsymbol{\nu}}$. In this way the simulation mimics the Gaussian models that are assumed in, for instance, [19] and [20] in the context of face recognition. In traditional face recognition, the columns of $\boldsymbol{U}_2$ would be the Eigenfaces [28] and the columns of $\boldsymbol{U}_2\boldsymbol{S}\boldsymbol{U}_1$ the Fisherfaces [29].

In this model, the comparison performance solely depends on $\boldsymbol{\Lambda}$. The other parameters create dependencies between the observations and map the noisy identity variable $\boldsymbol{b}+\boldsymbol{\nu}$ to the outcome space, which makes the data more realistic and aggravates the task of the classifier. An expression for the ground truth log-likelihood ratio is

$$\begin{aligned} \mathrm{llr}(\boldsymbol{x},\boldsymbol{y}) \;=\; & -\frac{1}{2}\sum_{i=1}^{d}\log\left(1-\Lambda_{ii}^2\right) \\ & -\frac{1}{4}\sum_{i=1}^{d}\frac{\Lambda_{ii}}{1-\Lambda_{ii}}(u_i-v_i)^2 \\ & +\frac{1}{4}\sum_{i=1}^{d}\frac{\Lambda_{ii}}{1+\Lambda_{ii}}(u_i+v_i)^2, \end{aligned} \tag{41}$$

with u_i and v_i elements of $\boldsymbol{u} = \boldsymbol{U}_1^{\mathrm{T}}\boldsymbol{S}^{-1}\boldsymbol{U}_2^{\mathrm{T}}\boldsymbol{x}$, and $\boldsymbol{v} = \boldsymbol{U}_1^{\mathrm{T}}\boldsymbol{S}^{-1}\boldsymbol{U}_2^{\mathrm{T}}\boldsymbol{y}$, respectively.

In the experiment with synthetic biometric data, we choose dimensionalities $D = 100$ and $d = 50$. The diagonal elements of $\boldsymbol{\Lambda}$ are chosen such that the equal error rate (the FMR at the point of operation where FMR=FNMR) of the log-likelihood ratio classifier is approximately 3%. In particular, for every combination of a training and a test set, the diagonal elements of $\boldsymbol{\Lambda}$ are drawn uniformly randomly from the interval $[0.85, 0.95]$ and the diagonal elements of $\boldsymbol{S}$ are independently drawn from an exponential distribution with mean 1. Also for every combination of a training and a test set the matrices $\boldsymbol{U}_1$ and $\boldsymbol{U}_2$ are obtained by first generating two Gaussian random matrices $\boldsymbol{R}_i$, $i = 1, 2$, $\boldsymbol{R}_1 \in \mathbb{R}^{d\times d}$, $\boldsymbol{R}_2 \in \mathbb{R}^{D\times d}$,$(\boldsymbol{R}_i)_{jk} \sim N(0,1)$, and taking $\boldsymbol{U}_1$ and $\boldsymbol{U}_2$ from the singular value decompositions $\boldsymbol{R}_i = \boldsymbol{U}_i\boldsymbol{S}_i\boldsymbol{V}_i^{\mathrm{T}}$, $i = 1, 2$.

Two combinations of a training set and a test set are generated with $K = 1.6\mathrm{M}, 6.4\mathrm{M}$ training pairs $(\boldsymbol{x},\boldsymbol{y})$, respectively. With every training set a network configuration as in Figure 8 is trained, consisting of two identical neural nets with shared weights $\boldsymbol{w}$ that have pairs $(\boldsymbol{x},\boldsymbol{y})$ and $(\boldsymbol{y},\boldsymbol{x})$ as respective inputs of which the scores are added. This configuration guarantees that $s(\boldsymbol{x},\boldsymbol{y}|\boldsymbol{w}) = s(\boldsymbol{y},\boldsymbol{y}|\boldsymbol{w})$, which we find to result in less noisy approximations of the llr. The configuration of the neural net is summarised in Table 1. During training the output is extended with a sigmoid function as in (2) with $\hat{\rho} = 0$ in order to predict the posterior probabilities needed for the binary cross-entropy loss function defined in (10), as shown in Figure 8. The batch size is $N = 1000$, and the training is performed over 900 epochs for $K = 1.6\mathrm{M}$ and over 1300 epochs for $K = 6.4\mathrm{M}$. For $K = 1.6\mathrm{M}$ the learning rate is decreased from 1.0×10^{-3} to 0.5×10^{-3} after 600 epochs. For $K = 6.4\mathrm{M}$ this is at 1000 epochs.

Figure 9 shows how the validation binary cross-entropy loss defined in (10) and the root mean-square approximation error (33) converge for the two different sizes K of training sets. The jumps at 600 epochs for $K = 1.6\mathrm{M}$ and at 1000 epochs for $K = 6.4\mathrm{M}$ mark the decrease of the learning rate. Figure 10 shows scatter plots of the relation between the true log-likelihood ratio llr (41) and the predicted $\widehat{\mathrm{llr}}$ are shown for $K = 1.6\mathrm{M}$ (red dots) and $K = 6.4\mathrm{M}$ (blue dots). The penalty function as a function of the ground truth llr, is shown in Figure 6 (red graph). Figures 11 plots the trade-off between the false-match rate (FMR) and the false-nonmatch rate (FNMR) as a function of varying thresholds for $\widehat{\mathrm{llr}}$ in a Decision Error Trade-off (DET) plot obtained from the training sets for the 2 different sizes of training sets, with $K = 1.6\mathrm{M}$ and $K = 6.4M$ for left and right panel, respectively. The dashed lines are the DET plots obtained with ground truth llrs, the solid lines are de DET plots obtained with the predicted $\widehat{\mathrm{llr}}$.

Our observations are largely similar to those with Example 1. Figure 9 shows that the networks need considerably more epochs to converge and, as in Example 1, it seems that the root mean-square approximation error is still decreasing when the binary cross-entropy loss curve has flattened off. The approximations of the log-likelihood ratio, shown Figure 10 are noisy, and considerably more than in Example 1, but close for $-20 < \mathrm{llr} < 10$, which is a wider range than in Example 1. The E_0 (34) approximation errors shown in the legend of Figure 10 are also up to an order of magnitude larger. The approximations improve with increasing K, are best at moderate llr values around $\mathrm{llr} = 0$, and decreases with increasing $|\mathrm{llr}|$. This is in line with the penalty function shown in Figure 6 (red graph) that also predicts the best approximation around $\mathrm{llr} = 0$. The penalty functions in Figure 6 also indicate that the llr

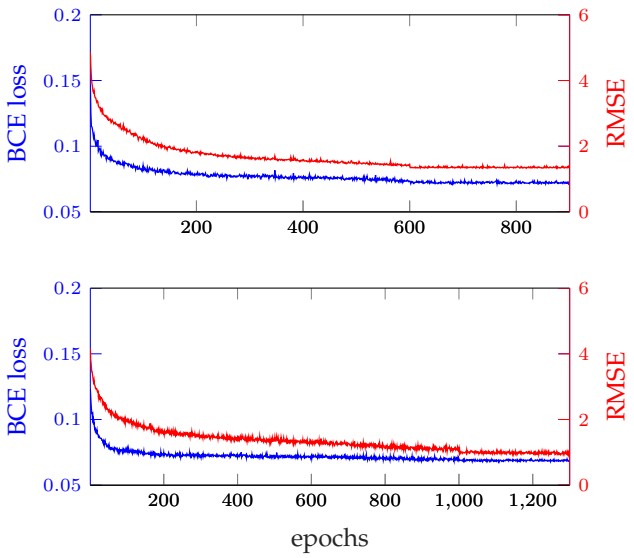

Fig. 9. Convergence of validation binary cross-entropy loss (10) on left y-axis and root mean-square approximation error (33) on right y-axis during training for Example 2: Simulated biometric class-conditional probability density functions, for different numbers of training samples K. Panels from top to bottom: $K = 1.6\mathrm{M}, 6.4\mathrm{M}$. The jumps at 600 epochs for $K = 1.6\mathrm{M}$ and at 1000 epochs for $K = 6.4\mathrm{M}$ mark the decrease of the learning rate.

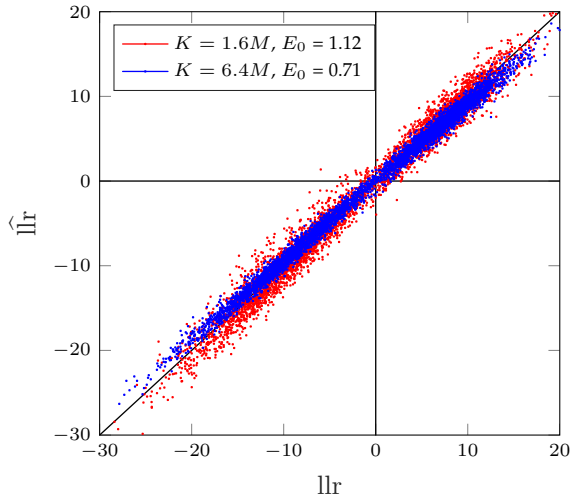

Fig. 10. Scatter plots of the predicted log-likelihood ratio $\widehat{\mathrm{llr}}$ as function of the true log-likelihood ratio llr for Example 2: Simulated biometric class-conditional probability density functions, for different numbers of training samples K, specified in the legend. The dimensionality of the biometric data is $d = 10$. The legend also provides the E_0 (34) approximation errors with $M = 100$.

of Example 2 can be worse than that of Example 1. The increased noisiness of the approximation can be explained by the increased dimensionality of the manifolds on which the llr is constant, from 1 to $d-1 = 49$, which leads to noisier approximation than in the case of Example 1. The poorer approximation for larger $|\mathrm{llr}|$ is due to the compressive behaviour of the sigmoid activation function that causes the penalty function (30) to vanish rapidly for larger $|\mathrm{llr}|$ values and makes the loss function insensitive to deviations

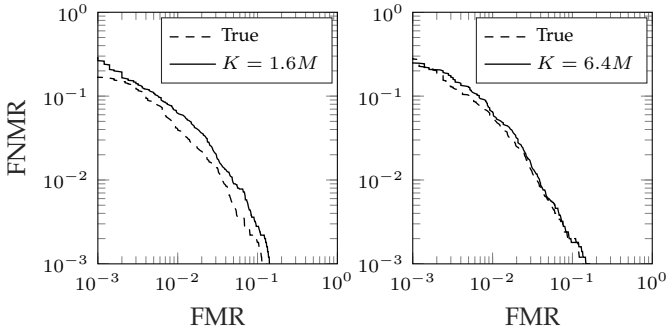

Fig. 11. DET plots of the predicted log-likelihood ratio $\widehat{\mathrm{llr}}$ and of the true log-likeliod ratio llr for Example 2: Simulated biometric class-conditional probability density functions, , for different numbers of training samples K. Panels from left to right: $K = 1.6\mathrm{M}, 6.4M$.

at larger $|\mathrm{llr}|$. The sigmoid activation function is crucial for the training procedure and the proof of Neyman-Pearson optimality, but its compressive behaviour limits good approximations at higher $|\mathrm{llr}|$. The DET plots in Figure 11 of the predicted log-likelihood ratios are approaching the DET plot for the true llr, in particular for $K = 6.4\mathrm{M}$. Although this approximation is not as close as in Example 1, we can say that the neural can achieved (near) optimal recognition performance in terms of DETs before the approximation of the llr has fully converged.

Finally, we note again that quite a large training set ($K > 1.6\mathrm{M}$) and many training epochs are needed to achieve a good approximation over a longer range of llr values. It is worth investigating other network topologies, and loss functions that are less compressive, but still warrant Neyman-Pearson optimality in order to improve this.

8 Conclusions

In this paper a proof is presented that binary neural network classifiers can output a score that converges to the log-likelihood ratio and hence, these networks can be optimal in Neyman-Pearson sense. For this, the network must be sufficiently rich and binary cross-entropy loss must be used for training.

A theoretical analysis shows that the approximation of the llr is best for moderate values of the log-likelihood ratio, that it requires balanced prior class probabilities in the training set, and that the approximation becomes poor is classes are well-separated in feature space.

Experiments with synthetic data, of which the ground truth log-likelihood ratios can be computed, confirm that the scores approximate the ground truth log-likelihood ratios over a range that is wide enough for practical applications and that the Decision Error Trade-off curve approximates that of the log-likelihood ratio classifier. However, the approximation is noisy and requires long training, 1000 epochs for the simulated biometric experiment, with extensive training sets of 1.6M–6.4M input samples. It is worth investigating whether other network topologies can reduce the training effort. The range of log-likelihood ratio values that can be approximated well is partly determined by the compressive behaviour of the sigmoid activation function

that is used during training in combination with the binary cross-entropy loss. It is, therefore, also worth investigating whether other combinations of activation function and loss function exist that can be used to approximate the log-likelihood ratio over a wider range.

ACKNOWLEDGMENT

The authors would like to thank the University of Twente spin-off company 20Face for supporting this research.

REFERENCES

[1] H. van Trees, *Detection, Estimation and Modulation Theory, Part I*. New York: John Wiley and Sons, 1968.
[2] D. Ramos-Castro, J. Gonzalez-Rodriguez, and J. Ortega-Garcia, "Likelihood ratio calibration in a transparent and testable forensic speaker recognition framework," in *2006 IEEE Odyssey - The Speaker and Language Recognition Workshop*, 2006, pp. 1–8.
[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.
[4] L. Zeune, "Automated ctc classification, enumeration and pheno typing: Where math meets biology," Ph.D. dissertation, University of Twente, Netherlands, February 2019.
[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385
[6] A. Y. Ng and M. I. Jordan, "On Discriminative vs. Generative Classifiers: A comparison of logistic regression and naive Bayes," in *Advances in Neural Information Processing Systems 14*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds. MIT Press, 2002, pp. 841–848.
[7] M. I. Jordan, "The Logistic Function? A tutorial discussion on probabilities and neural networks," Massachusetts Institute of Technology, Tech. Rep., 1995.
[8] A. Cannon, J. Howse, D. Hush, and C. Scovel, "Learning with the Neyman–Pearson and min–max criteria," Los Alamos National Laboratory, LANL Technical Report: LA-UR-02-2951, 2002.
[9] C. Scott and R. Nowak, "A Neyman–Pearson Approach to Statistical Learning," *IEEE Transactions on Information Theory*, vol. 51, no. 11, pp. 3806–3819, Nov. 2005. [Online]. Available: http://ieeexplore.ieee.org/document/1522642/
[10] P. Rigollet and X. Tong, "Neyman-Pearson Classification, Convexity and Stochastic Constraints," *Journal of Machine Learning Research*, vol. 12, no. 86, pp. 2381–2855, 2011.
[11] X. Tong, Y. Feng, and A. Zhao, "A survey on Neyman-Pearson classification and suggestions for future research: A **s** urvey on Neyman-Pearson classification," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 8, no. 2, pp. 64–81, Mar. 2016. [Online]. Available: http://doi.wiley.com/10.1002/wics.1376
[12] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, no. 4, pp. 303–314, Dec. 1989. [Online]. Available: https://doi.org/10.1007/BF02551274
[13] J. Schmidt-Hieber, "Nonparametric regression using deep neural networks with ReLU activation function," *arXiv:1708.06633 [cs, math, stat]*, Aug. 2017, arXiv: 1708.06633. [Online]. Available: http://arxiv.org/abs/1708.06633
[14] T. Suzuki, "Adaptivity of deep ReLU network for learning in Besov and mixed smooth Besov spaces: optimal rate and curse of dimensionality," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=H1ebTsActm
[15] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, Jan. 1989. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/0893608089900208
[16] H. Mhaskar and T. Poggio, "Deep vs. shallow networks: An approximation theory perspective," *arXiv:1608.03287 [cs, math]*, Aug. 2016, arXiv: 1608.03287. [Online]. Available: http://arxiv.org/abs/1608.03287
[17] B. Hanin, "Universal Function Approximation by Deep Neural Nets with Bounded Width and ReLU Activations," *arXiv:1708.02691 [cs, math, stat]*, Aug. 2017, arXiv: 1708.02691. [Online]. Available: http://arxiv.org/abs/1708.02691
[18] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang, "The Expressive Power of Neural Networks: A View from the Width," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 6231–6239.
[19] A. Bazen and R. Veldhuis, "Likelihood-ratio-based biometric verification," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 1, pp. 86–94, January 2004.
[20] D. Chen, X. Cao, L. Wang, F. Wen, and J. Sun, "Bayesian face revisited: A joint formulation," in *Proceedings of ECCV2012)*, 10 2012, pp. 566–579.
[21] Y. Peng, R. Veldhuis, and L. Spreeuwers, "Low-resolution face alignment and recognition using mixed-resolution classifiers," *IET Biometrics*, vol. 6, pp. 418–428(10), November 2017. [Online]. Available: http://digital-library.theiet.org/content/journals/10.1049/iet-bmt.2016.0026
[22] J. Chen, V. M. Patel, and R. Chellappa, "Unconstrained face verification using deep cnn features," in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, March 2016, pp. 1–9.
[23] D. Zeng, R. Veldhuis, L. Spreeuwers, and Q. Zhao, "Likelihood ratio based loss to finetune cnns for very low resolution face verification," in *2019 International Conference on Biometrics (ICB)*, June 2019, pp. 1–8.
[24] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.
[25] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980 [cs]*, Jan. 2017, arXiv: 1412.6980. [Online]. Available: http://arxiv.org/abs/1412.6980
[26] R. Askey and R. Roy, "Beta function," in *NIST Handbook of Mathematical Functions*, F. Olver, D. Lozier, R. Boisvert, and C. Clark, Eds. New York, NY: Cambridge University Press, 2010, pp. 135–148.
[27] "International ISO/IEC standard 2382-37, information technology — vocabulary — Part 37: Biometrics," 2012.
[28] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.
[29] P. N. Belhumeur, J. Hespanha, and D. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," *IEEE Transactions on Pattern Analysis and Machine Intelligence, Special Issue on Face Recognition*, vol. 17, no. 7, pp. 711–720, 1997.

Raymond Veldhuis graduated from the University of Twente, The Netherlands, in 1981. He received the Ph.D. degree from the Radboud University, Nijmegen, The Netherlands, in 1988. From 1982 to 1992, he was a Researcher with Philips Research Laboratories, Eindhoven, The Netherlands, in various areas of digital signal processing. From 1992 to 2001, he was involved in the field of speech processing at the Institute of Perception Research, Eindhoven. He is currently a Full Professor in Biometric Pattern Recognition with the University of Twente, where he is leading the Data Management and Biometrics group. His main research topic is machine learning for biometrics, including face recognition, fingerprint recognition, and vascular pattern recognition, and biometric template protection. His research is both applied and fundamental.

Dan Zeng received the B.E. and Ph.D. degrees in computer science and technology from Sichuan University in 2013 and 2018. From 2018 to 2020, she worked as a post-doc research fellow in the Data Management and Biometrics Group at the University of Twente, the Netherlands. She is currently a research assistant professor in the Department of Computer Science and Engineering at Southern University of Science and Technology. Her main research topics include 2D face recognition, 3D face modelling, and face super-resolution.