Article

Yet Another Discriminant Analysis (YADA): A Probabilistic Model for Machine Learning Applications

Richard V. Field, Jr. [1,*], Michael R. Smith [1], Ellery J. Wuest [2] and Joe B. Ingram [1]

1   Sandia National Laboratories, Albuquerque, NM 87185, USA; msmith4@sandia.gov (M.R.S.);
    jbingra@sandia.gov (J.B.I.)
2   Klipsch School of Electrical and Computer Engineering, New Mexico State University,
    Las Cruces, NM 88003, USA; ellerywu@nmsu.edu
*   Correspondence: rvfield@sandia.gov

Abstract: This paper presents a probabilistic model for various machine learning (ML) applications. While deep learning (DL) has produced state-of-the-art results in many domains, DL models are complex and over-parameterized, which leads to high uncertainty about what the model has learned, as well as its decision process. Further, DL models are not probabilistic, making reasoning about their output challenging. In contrast, the proposed model, referred to as Yet Another Discriminate Analysis(YADA), is less complex than other methods, is based on a mathematically rigorous foundation, and can be utilized for a wide variety of ML tasks including classification, explainability, and uncertainty quantification. YADA is thus competitive in most cases with many state-of-the-art DL models. Ideally, a probabilistic model would represent the full joint probability distribution of its features, but doing so is often computationally expensive and intractable. Hence, many probabilistic models assume that the features are either normally distributed, mutually independent, or both, which can severely limit their performance. YADA is an intermediate model that (1) captures the marginal distributions of each variable and the pairwise correlations between variables and (2) explicitly maps features to the space of multivariate Gaussian variables. Numerous mathematical properties of the YADA model can be derived, thereby improving the theoretic underpinnings of ML. Validation of the model can be statistically verified on new or held-out data using native properties of YADA. However, there are some engineering and practical challenges that we enumerate to make YADA more useful.

Keywords: machine learning; explainability; probabilistic model; synthetic data; uncertainty quantification

MSC: 68T01; 68T37; 62H05

1. Introduction

Deep learning (DL) has achieved state-of-the-art results in several domains [1–4]. However, DL techniques produce very complex models due to the large number of parameters and nonlinear relationships that are represented, making them difficult to interpret [5], susceptible to adversarial manipulation [6,7], and overconfident in their predictions [8,9]. As DL is being considered in more high-consequence domains, several research fields are emerging, including explainability [10,11], ethical AI [12], and uncertainty quantification [13,14], to quantify the risks associated with using DL. Herein, we examine the application of a probabilistic model that is straightforward, can be used for a wide variety of ML tasks, including classification, explainability, and uncertainty quantification, and is competitive in most cases with many state-of-the-art DL models. The proposed probabilistic model represents both the marginal distributions and pairwise correlations from the dataset, and is an extension to similar models in engineering disciplines [15,16] and the Gaussian copula in finance [17–19], medicine [20,21], geomechanics [22], climate [22–24],

and astronomy [25]. The use of copulae in machine learning has thus far been limited; for example, ML classification has been studied [26].

Often in ML applications, the input features are modeled (at least theoretically) as random variables. However, DL does not produce probabilistic models, making reasoning about their outputs and quantifying their uncertainty challenging [27]. Bayesian neural networks seek to build probabilistic models, but they are expensive to train and increase the overall complexity of the model [28]. In problems of practical interest, the features/variables are interdependent, often times in very complex ways, and the full joint distribution function is required to correctly represent them. However, calculating and implementing a model for the full joint distribution from data are not feasible, even with a relatively small number of features. One solution that is often pursued is to assume that the features are mutually independent, so that the joint distribution is equal to the product of the marginal distributions, which can easily be estimated from data. This assumption is very limiting and can lead to poor results.

We present a model that captures the marginal distributions and pairwise correlations from the dataset, referred to as Yet Another Discriminant Analysis or YADA, which can be viewed as an intermediary step between modeling just the marginal distributions and modeling the full joint distribution. A list of benefits of YADA includes the following:

- YADA is less complex than other methods and yet is based on a mathematically rigorous foundation. It is straightforward to derive, for example, joint distributions, joint likelihood functions, joint entropy, and K-L divergence, among others.
- YADA can represent continuous or discrete-valued features with arbitrary marginal distributions, i.e., the normality assumption is not required.
- Pairwise correlations between features are well represented, i.e., an assumption of mutual independence is not required.
- Given an unlabeled test point, YADA can be used to assess the joint likelihood that the point came from any of the relevant classes. If the likelihood of the point is sufficiently small for all classes, this is an indication that this point is out of distribution and any predicted label should not be trusted. This can be used as a measure for confidence in the prediction that is lacking in standard ML.
- YADA has the ability to utilize the empirical distribution combined with an extrapolation model for the left and right tails for more refined feature modeling.
- The YADA model provides a mapping between the feature space and the space of multivariate normal (MVN) random variables, which can be very useful because many calculations are straightforward in the MVN space.
- YADA provides ML explanations based on the marginal likelihood for each feature, and can be used to create realistic synthetic data to address, for example, class and/or feature imbalance.

YADA is an extension to the Gaussian copula due to its use for explainability, out-of-distribution detection, prediction confidence, refined feature modeling, and synthetic data generation. While there have been some limited studies of Gaussian copulae for ML applications, we believe this paper to be the first survey in the broad use of the approach and inspire future research in this area.

Despite these benefits, YADA has several limitations. First, it can be less accurate for classification than large, complex DL models. Second, YADA captures only marginal distributions and pairwise correlations among features. Higher-order information, such as the full joint distribution, may not be captured accurately. Further, YADA operates only on the known feature vector (as opposed to DL models, which can learn their own set of features through the layers of the network). We view YADA as a first step to providing a theoretic model that would be suitable for high-consequence applications. Future work would address these short-comings of YADA to improve its performance goals.

The outline of our paper is as follows. The YADA model is described in detail in Section 2, with subsections on definitions and model properties. YADA for machine learning applications is presented in Section 3, with sections on training, synthetic data gener-

ation, classification, and uncertainty estimation; some of these topics require additional details, which are presented in the attached Appendices. Some concluding remarks and thoughts about future work are provided in Section 4. Examples with commonly available datasets are provided throughout the discussion to further illustrate the approach.

2. The YADA Model

Let $\mathbf{x} = (x_1, \ldots, x_d)^T$ denote a vector of $d \geq 1$ features with the corresponding class label $y \in \{0, \ldots, \kappa - 1\}$, where κ denotes the number of classes. Often, in machine learning applications, the features are modeled as random variables; we use the notation $\mathbf{X} = (X_1, \ldots, X_d)^T$ to represent a vector of random variables used to model the feature vector $\mathbf{x}$. In this section, we present a probabilistic model for $\mathbf{X}$, which we call Yet Another Discriminant Analysis (YADA), that can be used to represent correlated features with arbitrary marginal distribution.

The main premise behind the YADA model is to represent each feature as a function h of a standard normal or Gaussian random variable. This allows for the marginal distribution of each feature to be controlled by the functional form of h, while the correlation of the underlying d Gaussian random variables controls the correlation between features. Both continuous and discrete-valued features can be modeled in this way.

Hence, the YADA model captures the pairwise correlations and marginal distributions from the dataset of interest. Many probabilistic models assume that the features are either normally distributed, mutually independent, or both, which can limit their performance (see Figure 1). While YADA does not have these limitations, it is important to acknowledge that there is no guarantee that higher-order information, such as joint distributions of any two or more features, will be represented accurately.

Figure 1. How YADA compares with other discriminant methods.

The definition of the YADA model is presented in Section 2.1, followed by a derivation of some properties of the model in Section 2.2. Examples using standard machine learning datasets are included throughout for demonstration.

2.1. Definition

We define the YADA model in this section. We start with a single continuous feature to simplify the discussion, then generalize the model to the case of two or more features that can be continuous or discrete.

2.1.1. Single Feature

Consider first the case of a single feature $\mathbf{x} = x$, and let X be a continuous random variable that represents a model for x. Further, let $G \sim N(0, 1)$ denote a standard normal or Gaussian random variable with zero mean, unit variance, and probability density function (pdf) $\phi(u) = (1/\sqrt{2\pi})e^{-u^2/2}$, $-\infty < u < \infty$. The proposed YADA model for feature x is given by

$$X = h(G) = F^{-1}(\Phi(G)), \tag{1}$$

where F is an arbitrary cumulative distribution function (cdf) and $\Phi(z) = \int_{-\infty}^{z} \phi(u)\,du$ denotes the cdf of G. Because F and Φ are both cdfs (monotonic functions), $h = F^{-1} \circ \Phi$ is invertible, that is,

$$G = h^{-1}(X) = \Phi^{-1}(F(X)). \tag{2}$$

We can therefore interpret G defined by Equation (2) as the Gaussian image of X.

In general, the functional form of h is nonlinear so that X defined by Equation (1) is a non-Gaussian random variable, and it is simple to show that F is actually the cdf of X. This follows because

$$\Pr(X \leq a) = \Pr(h(G) \leq a) = \Pr(G \leq \Phi^{-1}(F(a))) = \Phi(\Phi^{-1}(F(a))) = F(a).$$

For example, let X be an exponential random variable defined by cdf $F(x) = 1 - e^{-\lambda x}$, $x \geq 0$, where $\lambda > 0$ is a parameter. By Equation (1), X can be expressed as $h(G) = -(1/\lambda)\ln(1 - \Phi(G))$ because $F^{-1}(y) = -(1/\lambda)\ln(1 - y)$ is the inverse cdf of X. Similarly, $X = h(G) = a + (b - a)\Phi(G)$, with $a < b$, is a uniform random variable on the interval $[a, b]$. These examples are illustrated in Figure 2. On the left is a histogram of 10,000 samples of a standard Gaussian random variable G. These samples can be mapped to samples of an exponential or uniform random variable using the mapping function, h, described above. The model defined by Equation (1) is referred to as the Nataf transformation [29] or translation random variable [15].
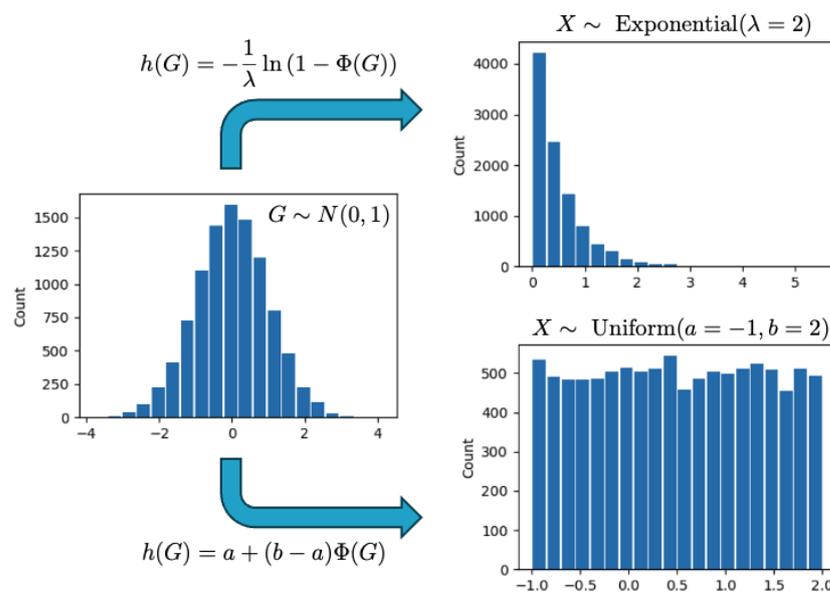
Figure 2. Histograms of random samples of exponential and uniform random variables from samples of a standard Gaussian variable using Equation (1).

2.1.2. Two or More Features

Next, consider the more general case of $d > 1$ continuous features. The YADA model for the ith feature is given by

$$X_i = h_i(G_i) = F_i^{-1}(\Phi(G_i)), \; i = 1, \ldots, d, \tag{3}$$

where each $G_i \sim N(0, 1)$ is a normal or Gaussian random variable with zero mean and unit variance, and each h_i is an invertible function. The Gaussian variables $\{G_1, \ldots, G_d\}$ are correlated and have joint pdf:

$$\phi_d(u_1, \ldots, u_d; \mathbf{c}) = \phi_d(\mathbf{u}; \mathbf{c}) = (2\pi)^{-d/2} \det(\mathbf{c})^{-1/2} e^{-1/2(\mathbf{u}^T \mathbf{c}^{-1} \mathbf{u})} \tag{4}$$

where $\mathbf{c} = \{\mathrm{E}[G_i\,G_j]\}$ is a $d \times d$ correlation matrix with determinant $\det(\mathbf{c})$; Equation (4) is also referred to as the multivariate normal pdf. Note that $\mathbf{c}$ is symmetric and positive definite with ones on the diagonal. The model described by Equation (3) is referred to as a translation random vector [16] and Gaussian copula [17].

The inverse of Equation (3) is given by Equation (2) with the addition of subscript i and provides the marginal distribution for G_i, the Gaussian image of X_i. However, when applying this inverse mapping, there is no guarantee that the collection $\{G_1, \ldots, G_d\}$ will be jointly Gaussian.

The connection between the feature space and the space of multivariate Gaussian random variables is an important property of the YADA model, and one that is exploited in the following sections for classification, synthetic data, and other applications. To further illustrate this connection, consider Figure 3, which illustrates two features X_1 and X_2 mapped to two correlated Gaussian variables G_1 and G_2; the marginal histograms are also shown. The data can be mapped back and forth from the two spaces using Equation (3) and its inverse. These data came from the phoneme dataset [30].
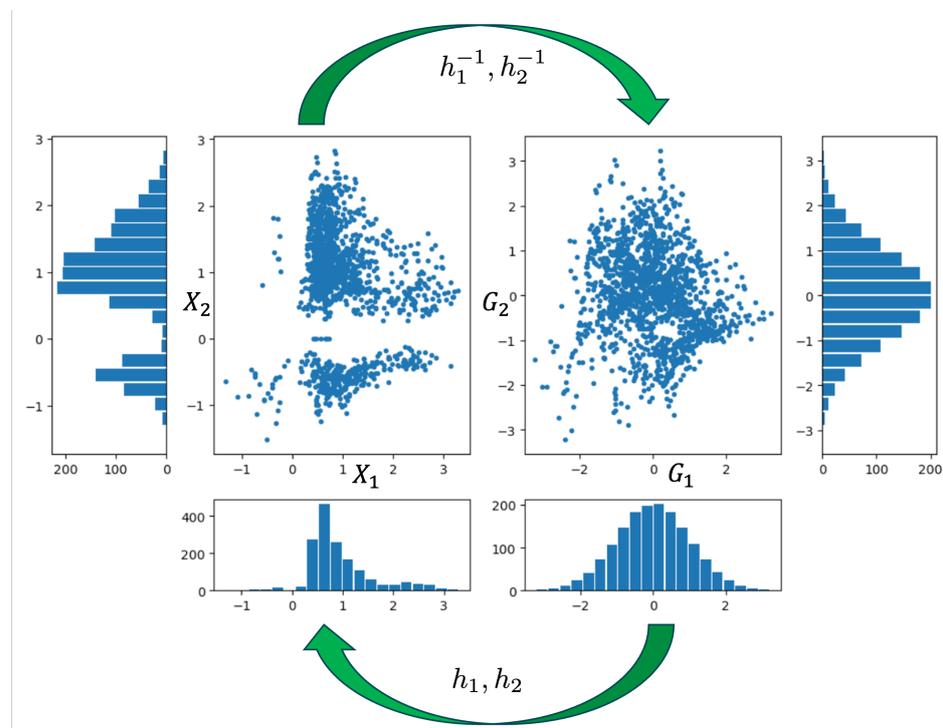
Figure 3. The connection between the feature space (left) and the space of multivariate Gaussian random variables (right) for the phoneme dataset [30], where X_i and G_i, $i = 1, 2$, are defined by Equation (3).

Throughout this section, we have assumed continuous-valued features/random variables. Discrete-valued features can be modeled as discrete random variables and, with some modifications, can be handled in the same manner; see Appendix A for additional details.

2.2. Properties

We next derive some properties of the YADA model, such as the correlation matrix and joint distribution functions of $(X_1, \ldots, X_d)$. As mentioned in the previous section, there is a direct connection between the features $\{X_i\}$ and their Gaussian images $\{G_i\}$; see Equation (3). It follows that many of the properties of $\{X_i\}$ can be expressed in terms of the well-known properties of $\{G_i\}$.

2.2.1. Marginal Distributions

The marginal cdf and pdf of X_i, the model for the ith feature, define the probability distribution of the ith feature independent of the others. They can be expressed in terms of Φ and ϕ, the cdf and pdf of a standard univariate Gaussian random variable. This relationship is

$$F_i(x_i) = \Phi(w_i(x_i)) \tag{5}$$

and

$$f_i(x_i) = \frac{\mathrm{d}}{\mathrm{d}x_i} F_i(x_i) = \left| \frac{\mathrm{d}}{\mathrm{d}x} w_i(x_i) \right| \phi(w_i(x_i)), \tag{6}$$

where we have introduced function $w_i(x) = h_i^{-1}(x)$ for notational convenience. Functions F_i and f_i defined by Equations (5) and (6) are the marginal cdf and pdf, respectively, of non-Gaussian random variable X_i.

2.2.2. Joint Distribution

The marginal distributions are not sufficient to define a collection of two or more random variables; we need the joint distribution to perform this. Next, let

$$F(x_1, \ldots, x_d) = \mathrm{Pr}(X_1 \leq x_1, \ldots, X_d \leq x_d)$$

denote the joint cdf of all d features $X_1, \ldots, X_d$. It follows that the joint pdf of $X_1, \ldots, X_d$ can be expressed in terms of the d-variate normal pdf (Equation (4)), i.e.,

$$
\begin{aligned}
f(x_1, \ldots, x_d) &= \frac{\partial^d}{\partial x_1 \cdots \partial x_d} F(x_1, \ldots, x_d) \\
&= \left| \prod_{i=1}^d \frac{\mathrm{d}}{\mathrm{d}x_i} w_i(x_i) \right| \phi_d(w_1(x_1), \ldots, w_d(x_d); \mathbf{c}) \\
&= \prod_{i=1}^d \left(\frac{f_i(x_i)}{\phi(w_i(x_i))} \right) \phi_d(\mathbf{w}; \mathbf{c}),
\end{aligned}
\tag{7}
$$

where $\mathbf{w} = (w_1(x_1), \ldots, w_d(x_d))^T$ and the last line follows from the results in Appendix B. Hence, the joint pdf of $X_1, \ldots, X_d$ is the product of their marginal distributions and the joint pdf of $G_1, \ldots, G_d$.

As mentioned in the introduction, the YADA model works by capturing the pairwise correlations and marginal distributions of the dataset of interest. However, the joint pdf as imposed by Equation (7) may not be sufficient to represent the true joint distribution of the underlying data. This should be considered when building the YADA model and is a trade-off between more complex models.

To demonstrate the joint distribution of a YADA model, we consider a simple example. Let

$$
\begin{aligned}
X_1 &= h_1(G_1) = 2 + e^{G_1} \\
X_2 &= h_2(G_2) = 2 + e^{1/2 + G_2}
\end{aligned}
$$

be the YADA model for $d = 2$ features, where G_1 and G_2 are correlated Gaussian variables with $\mathrm{E}[G_1 G_2] = -0.2$. Features X_1 and X_2 are correlated and each has a log-normal marginal distribution. Contours of $f(x_1, x_2)$, the joint pdf of $(X_1, X_2)^T$, are illustrated by the right panel in Figure 4; the left panel illustrates contours of $\phi_2(u_1, u_2)$, the joint pdf of $(G_1, G_2)^T$, i.e., the Gaussian image of $(X_1, X_2)^T$. Again, the connection between these two functions is defined by Equation (7) with $d = 2$.
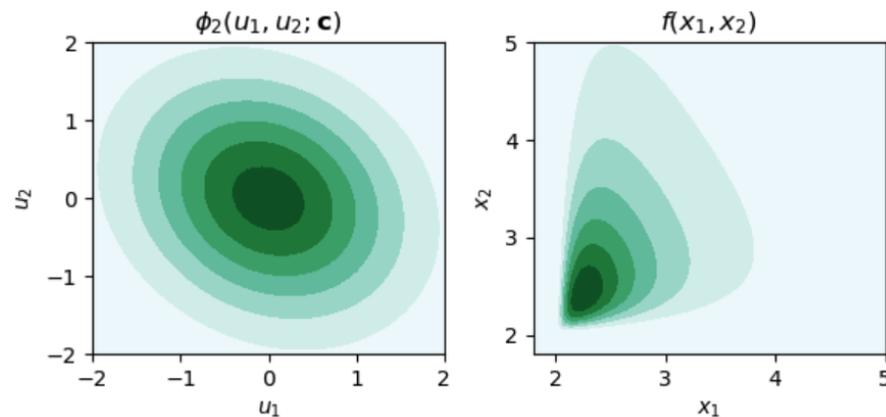
Figure 4. Contours of the joint pdf of (G_1, G_2) (**left**) and (X_1, X_2) (**right**).

2.2.3. Pairwise Correlations

The correlation between two features X_i and X_j can be expressed in terms of $c_{ij} = \mathrm{E}[G_i\, G_j]$, the correlation between their Gaussian images, i.e.,

$$\mathrm{E}[X_i\, X_j] = \mathrm{E}[h_i(G_i)\, h_j(G_j)] = \int_{\mathbb{R}^2} h_i(u)\, h_j(v)\, \phi_2\left(u, v; \begin{pmatrix} 1 & c_{ij} \\ c_{ij} & 1 \end{pmatrix}\right) \mathrm{d}u\, \mathrm{d}v, \qquad (8)$$

where $\mathrm{E}[A]$ denotes the expected value of random variable A, and ϕ_2 is the bi-variate normal pdf with zero mean defined by Equation (4) with $d = 2$. It is typically not possible to further simplify the relationship between $\mathrm{E}[X_i\, X_j]$ and $\mathrm{E}[G_i\, G_j]$.

We note that there is no guarantee that the resulting matrix $\{\mathrm{E}[X_i, X_j]\}$ constructed from these correlations will be positive semi-definite, as is required for it to be a valid correlation matrix. This issue has been studied extensively (see [16,31], Section 3.1) and some solutions have been suggested, e.g., [32]. However, this is a not an issue in our case because, as will be described in Section 3.1, we compute the correlation matrix of the Gaussian image of the data, which is guaranteed to be positive semi-definite [33] (Chapter 7).

2.2.4. Likelihood Functions

The likelihood function can be used to measure how well the YADA model explains observed data; we can define a marginal likelihood function for data on a single feature, as well as a joint likelihood for all features in the dataset. The marginal log-likelihood function is the log of the marginal pdf defined by Equation (6), i.e.,

$$\ln(f_i(x)) = \ln\left(|w_i'(x)|\right) - \frac{1}{2}\left(\ln(2\pi) + w_i(x)^2\right). \qquad (9)$$

The joint log-likelihood function follows from Equation (7) and can be expressed as

$$\begin{aligned} \ell(\mathbf{x}) = \ln(f(\mathbf{x})) &= \sum_{i=1}^{d} \ln\left(\frac{f_i(x_i)}{\phi(w_i(x_i))}\right) + \ln(\phi_d(\mathbf{w}; \mathbf{c})) \\ &= \sum_{i=1}^{d} \ln(f_i(x_i)) - \sum_{i=1}^{d} \ln(\phi(w_i(x_i))) \\ &\qquad - \frac{1}{2}\left(\ln(\det(\mathbf{c})) + \mathbf{w}^T \mathbf{c}^{-1} \mathbf{w} + d \ln(2\pi)\right). \end{aligned} \qquad (10)$$

Note that the first two terms in Equation (10) are the log of the marginal likelihood function for feature X_i and its Gaussian image, and the third term is the log-likelihood function of the d-variate normal distribution with zero mean.

2.2.5. Conditional Distributions

We can also derive the distribution of a conditional YADA model, which can be useful when the values of some of the features are known and fixed. Let the feature vector be partitioned as

$$(X_1, \ldots, X_d)^T = (Y_1, \ldots, Y_m, Z_{m+1}, \ldots, Z_d)^T = \begin{pmatrix} \mathbf{Y} \\ \mathbf{Z} \end{pmatrix}$$

where the features have been re-ordered such that any with known fixed values are collected into vector $\mathbf{Z}$, and the remaining $m \leq d$ features are random variables and collected into vector $\mathbf{Y}$. The joint pdf of $\mathbf{Y}$, given that $\mathbf{Z} = \mathbf{z}$ is known and fixed, can be obtained by exploiting the well-known property that conditional Gaussian variables also follow a Gaussian distribution [31] (Appendix C), that is,

$$f_{\mathbf{Y}|\mathbf{Z}}(\mathbf{y}|\mathbf{z}) = \frac{f(\mathbf{x})}{f_{\mathbf{Z}}(\mathbf{z})} = \prod_{i=1}^{m} \left(\frac{f_i(y_i)}{\phi(w_i(y_i))} \right) \phi_m \left(\mathbf{w_Y}; \mathbf{c_{YZ}} \, \mathbf{c_{ZZ}}^{-1} \, \mathbf{w_Z}, \mathbf{c_{YY}} - \mathbf{c_{YZ}} \, \mathbf{c_{ZZ}}^{-1} \, \mathbf{c_{YZ}}^T \right), \quad (11)$$

where

$$\mathbf{c} = \begin{pmatrix} \mathbf{c_{YY}} & \mathbf{c_{YZ}} \\ \mathbf{c_{YZ}}^T & \mathbf{c_{ZZ}} \end{pmatrix}, \mathbf{w_Y} = (w_1(z_1), \ldots, w_m(z_m))^T, \text{ and } \mathbf{w_Z} = (w_{m+1}(z_{m+1}), \ldots, w_d(z_d))^T.$$

2.2.6. Entropy and K-L Divergence

Some information on the theoretic properties of the YADA model can also be established. The (differential) joint entropy of a YADA model is given by

$$- \int_{\mathbb{R}^d} f(\mathbf{x}) \log f(\mathbf{x}) \, d\mathbf{x}, \quad (12)$$

where f is the joint pdf defined by Equation (7), and $\log f(\mathbf{x})$ can be obtained from Equation (10).

Suppose that we have two YADA models with joint pdfs $f^{(i)}$ and $f^{(j)}$; the relative entropy or Kullback-Liebler (K-L) divergence is a type of statistical distance between the two models. The K-L divergence of model i from reference model j is given by

$$d_{\mathrm{KL}}(i, j) = \int_{\mathbf{R}^d} f^{(i)}(\mathbf{x}) \log \left(\frac{f^{(i)}(\mathbf{x})}{f^{(j)}(\mathbf{x})} \right) d\mathbf{x}. \quad (13)$$

Using Equations (7) and (10), we have

$$d_{\mathrm{KL}}(i, j) = \int_{\mathbf{R}^d} \prod_{k=1}^{d} \left(\frac{f_k^{(i)}(x_k)}{\phi(w_k^{(i)}(x_k))} \right) \left[\sum_{k=1}^{d} \left(\ln \left(\frac{f_k^{(i)}(x_k)}{f_k^{(j)}(x_k)} \right) + \ln \left(\frac{\phi(w_k^{(j)}(x_k))}{\phi(w_k^{(i)}(x_k))} \right) \right) - \right.$$
$$\frac{1}{2} \ln \left(\frac{\det(\mathbf{c}^{(i)})}{\det(\mathbf{c}^{(j)})} \right) - \frac{1}{2} (\mathbf{w}^{(i)})^T (\mathbf{c}^{(i)})^{-1} \mathbf{w}^{(i)} +$$
$$\left. \frac{1}{2} (\mathbf{w}^{(j)})^T (\mathbf{c}^{(j)})^{-1} \mathbf{w}^{(j)} \right] \phi_d(\mathbf{w}^{(i)}; \mathbf{c}^{(i)}) \, d\mathbf{x}, \quad (14)$$

where $\mathbf{c}^{(i)}$ and $\mathbf{w}^{(i)}$ represent covariance matrix $\mathbf{c}$ and vector $\mathbf{w}$ defined above for model i. The K-L divergence is not symmetric, meaning that $d_{\mathrm{KL}}(i, j) \neq d_{\mathrm{KL}}(j, i)$ in general.

Equation (14) may be difficult to compute in practice, but there is an alternative. Because of the connection between each feature and its Gaussian image, i.e., Equation (3),

we can also compute the K-L divergence in the space of multivariate normal random variables as

$$d_{\text{KL}}(i,j) = \frac{1}{2}\left(\text{tr}\left((\mathbf{c}^{(j)})^{-1}\mathbf{c}^{(i)} \right) - \ln\left(\frac{\det(\mathbf{c}^{(i)})}{\det(\mathbf{c}^{(j)})} \right) - d \right), \tag{15}$$

where $\text{tr}(\mathbf{c})$ denotes the trace of matrix $\mathbf{c}$, and d is the number of features.

3. YADA for Machine Learning

In this section, we discuss using the YADA model in the context of machine learning applications. The topics of training a YADA model and utilizing it for classification are presented in Sections 3.1 and 3.2, respectively. As described in Section 3.3, explanations for predicted labels can also be provided; a separate model for explainability is not needed. An approach for assessing confidence in ML model predictions with YADA is presented in Section 3.4, and Section 3.5 contains a discussion on creating synthetic data from a trained YADA model. Examples using standard machine learning datasets are included throughout for demonstration.

3.1. Training

Suppose we have training data $\{(\mathbf{x}, y)_j, \ j = 1, \ldots, n\}$, where each $\mathbf{x} = (x_1, \ldots, x_d)^T$ is a vector of d features, and each $y \in \{0, \ldots, \kappa - 1\}$ is a class label. In this section, we describe how a YADA model can be trained with these data, that is, how the data can be used to learn the functions $h_1, \ldots, h_d$ and the correlation matrix $\mathbf{c}$ defined by Equations (3) and (4), respectively.

We first partition the training data per class, then train κ YADA models, one for each class. Each model represents the probability distribution for a single class; we are not modeling one class versus all of the other classes. For each class, we follow five steps for training:

1. Estimate F_i, the marginal cdf for each of the d features. This can be performed, for example, using the empirical cdf or kernel-based methods such as the kernel density estimator with a Gaussian kernel. The former is appropriate if it is especially important to maintain the support of each random variable observed in the data, while the latter is important if interpolation within and extrapolation beyond the support of the observed data are desired. Further, YADA has the ability to use the empirical cdf combined with an extrapolation model for the left and right tails; see Appendix C.

2. Estimate the corresponding inverse marginal cdf for each feature. If a kernel density estimator was used for the previous step, we can solve for the inverse using, for example, interpolating spline functions.

3. Compute the Gaussian image of each feature in the training set using Equation (2); this will produce n samples of a random vector that we assume follows a d-variate normal distribution.

4. Compute the sample covariance matrix $\mathbf{c}$ of the data produced in the previous step.

5. Compute the inverse and log-determinant of $\mathbf{c}$. In practice, we compute the pseudo-inverse and pseudo-determinant to handle any numerical issues caused by collinearity in the data. For any features with zero variance, IID Gaussian noise with small variance can be added to the data.

As mentioned, we first partition the training data per class, then train κ YADA models, one for each class. This approach is useful when applying YADA for classification, where it is necessary to evaluate the likelihood that a test point comes from the YADA model for each of the labels; this will be discussed in Section 3.2. As in other ML methods, the quality of the model depends on the quality and amount of the training data. If data of one class outnumber those of another class, YADA models for any underrepresented classes will

have lower quality. However, the amount of data required by YADA is significantly less than that typically required for DL models.

It is possible to instead train a single YADA model to all of the data regardless of class label. As discussed in Section 3.5, this approach is useful for synthetic data generation when, in addition to producing synthetic data on features alone, we also want to produce synthetic data on the corresponding labels.

Once trained, comparing the YADA models can provide useful information about how well separated the training data are among the different classes. This measure can provide a baseline for trustworthiness; if the classes are close together and hence difficult to separate, we should perhaps be skeptical of any predictions by ML models trained on these data. To illustrate, we consider the Modified National Institute of Standards and Technology (MNIST) dataset of images of handwritten digits 0, 1, . . . , 9 [34]. The dataset contains 60,000 images for training and 10,000 images for testing, and they are approximately evenly distributed over the 10 classes. Each image contains 28×28 grayscale pixels which, when scaled, can be interpreted as $28^2 = 784$ continuous features that take values in the interval $[0, 1]$. We trained a YADA model for each class, and then, as described in Section 2.2.6, computed the K-L divergence amongst the 10 models. The results are illustrated in Figure 5. The left panel illustrates the K-L divergence between the 10 models as a matrix colored by the value of the divergence, with values ranging from 0 along the diagonal to a maximum of approximately 350; the divergence is greatest for the YADA models of class 0 and class 1. The right panel illustrates a "class separation score", which is simply a symmetric version of the K-L divergence, i.e., $(d_{\mathrm{KL}}(i, j) + d_{\mathrm{KL}}(j, i))/2$, with each being normalized by the entropy of the reference distribution (see Equation (12)). The class separation scores take values between zero and one and are illustrated as a bar plot, sorted in decreasing order. The greatest separation is between classes 0 and 1; the least separation is between classes 7 and 9.
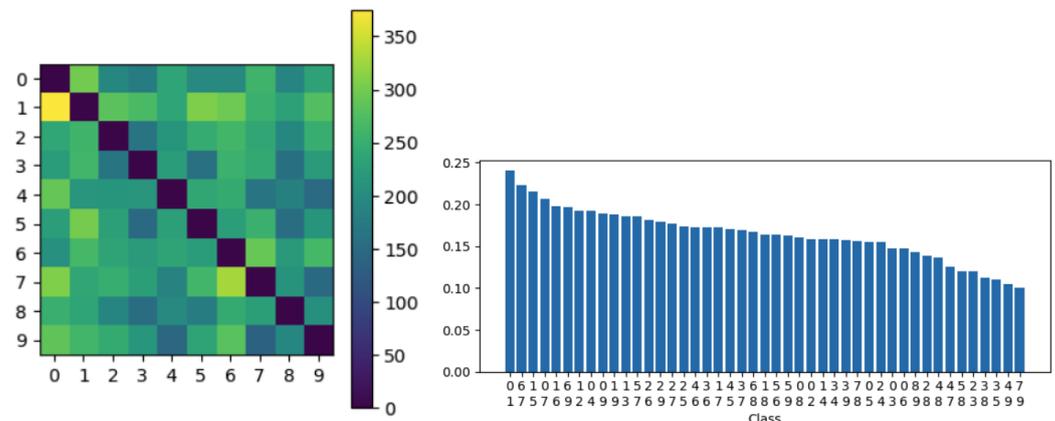
Figure 5. The K-L divergence between 10 YADA models trained to the MNIST dataset (**left**). The (**right**) panel illustrates the class separation score, which is a symmetric and normalized version of the K-L divergence, sorted in descending order.

3.2. Classification

Suppose we have followed the steps outlined in Section 3.1, resulting in κ-trained YADA models, one for each class label. We distinguish YADA models for different classes by using a superscript to denote the class label for all relevant model properties. For example, $\mathbf{c}^{(i)}$ will denote the sample covariance obtained in training step 4 and $f^{(i)}$ represents the joint PDF defined by Equation (7), both for the YADA model for class i. Let $\mathbf{x}^*$ denote a feature vector with an unknown label. In this section, we present ways to use the collection of YADA models to predict the label for $\mathbf{x}^*$.

3.2.1. Likelihood

The probability that $\mathbf{x}^*$ comes from class i is given by a version of Bayes' formula, i.e.,

$$\Pr(\text{class } i|\mathbf{x}^*) = \frac{\Pr(\mathbf{x}^*|\text{class } i)\Pr(\text{class } i)}{\sum\limits_{j=0}^{\kappa-1}\Pr(\mathbf{x}^*|\text{class } j)\Pr(\text{class } j)} = \frac{f^{(i)}(\mathbf{x}^*)\,\pi^{(i)}}{\sum\limits_{j=0}^{\kappa-1} f^{(j)}(\mathbf{x}^*)\,\pi^{(j)}}, \tag{16}$$

where $\pi^{(i)} = \Pr(\text{class } i)$ is the prior probability that an unlabeled test point belongs to class i, and $f^{(i)}(\mathbf{x}^*)$ defined by Equation (7) can be interpreted as the joint likelihood that $\mathbf{x}^*$ comes from class i. Typically, the prior probabilities are assumed to be the relative frequency of each class in the training dataset; non-informative priors, where each class is assumed equally likely, can also be used. To address any numerical issues, for calculations, we instead compute the log of Equation (16), i.e.,

$$\ln\left(\Pr(\text{class } i|\mathbf{x}^*)\right) \propto \ell^{(i)}(\mathbf{x}^*) + \ln(\pi^{(i)}),$$

where $\ell^{(i)} = \ln(f^{(i)})$ is given by Equation (10), and $a \propto b$ means that a is linearly proportional to b.

To demonstrate classification, we trained three YADA models on the Fisher iris data, one per class label, then evaluated Equation (16) throughout the feature space to determine the classification regions for each label, that is, the regions

$$\mathcal{R}_i = \{\mathbf{x} : \Pr(\text{class } i|\mathbf{x}) > \Pr(\text{class } j|\mathbf{x}), \forall j \neq i\}. \tag{17}$$

Hence, region $\mathcal{R}_i$ is the set of all points in feature space that YADA will predict as coming from class i. Figure 6 illustrates 2D projections of regions $\mathcal{R}_0$ (blue), $\mathcal{R}_1$ (orange), and $\mathcal{R}_2$ (green). The training data are also shown for reference. Note that in the plot on the right, the white area represents a part of the feature space where the likelihood of all three models is extremely small, and YADA cannot provide a prediction for test points that fall in this area.
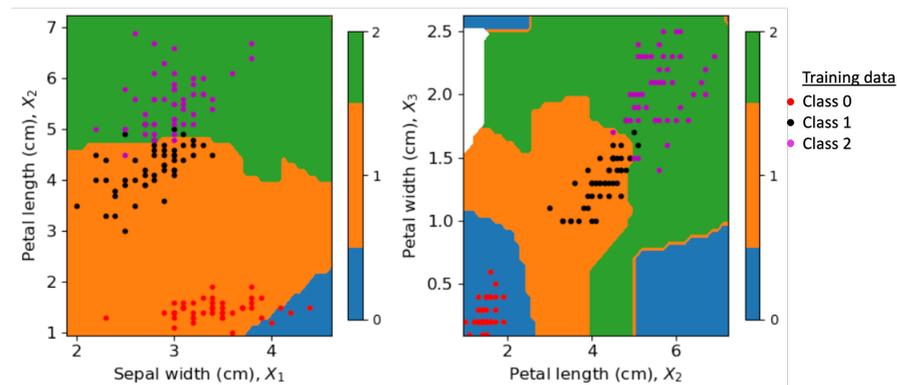
Figure 6. Two-dimensional projections of the YADA classification regions $\mathcal{R}_i$, $i = 0, 1, 2$, for the Fisher iris dataset.

3.2.2. Mahalanobis Distance

One alternative approach for classification is to utilize the Mahalanobis distance, a measure of the distance of a point to a distribution. Given a test point $\mathbf{x}^*$, we can calculate its Gaussian image $\mathbf{g}^{(i)}$ assuming it originates from class i; $\mathbf{g}^{(i)}$ is a vector of d coordinates, with each coordinate obtained by using Equation (2), i.e.,

$$g_j^{(i)} = \Phi^{-1}\left(F_j^{(i)}(x_j^*)\right),$$

where $F_j^{(i)}$ is the cdf for feature j assuming class i.

For classification, we assign $\mathbf{x}^*$ to class i if it is closest to the YADA model for class i, that is, if $m_i(\mathbf{x}^*) \leq m_j(\mathbf{x}^*)$, $j = 0, \ldots, \kappa - 1$, where

$$m_i(\mathbf{x}^*) = \left((\mathbf{g}^{(i)})^T (\mathbf{c}^{(i)})^{-1} \mathbf{g}^{(i)} \right)^{1/2} \tag{18}$$

is the Mahalanobis distance of $\mathbf{x}^*$ from the YADA model for class i.

3.2.3. Ensemble Method with Dimension Reduction

A second alternative to using the likelihood approach is to classify using an ensemble of reduced-order YADA models. This approach can be beneficial if the computational cost of using all features is too great, or when there exists some collinearity between the features that results in a poorly conditional covariance matrix. Let $I' \subseteq \{1, \ldots, d\} = I_d$ be a subset of d features, with $|I'| = d' \leq d$. The d' features can be selected at random or by some feature importance metric. From Equation (7), the joint PDF of the reduced-order feature vector is obtained by integrating out the dependence on those features not contained in I', i.e.,

$$\int_{\mathbb{R}^{d-d'}} f(\mathbf{x}) \prod_{i \in I_d \setminus I'} \mathrm{d}x_i = \prod_{i \in I'} \left(\frac{f_i(x_i)}{\phi(w_i(x_i))} \right) \phi_{d'}(\mathbf{w}'; \mathbf{c}'), \tag{19}$$

where $\mathbf{w}'$ is a vector with d' coordinates $w_i(x_i)$, $i \in I'$, and $\mathbf{c}'$ is a $d' \times d'$ covariance matrix obtained by retaining the rows and columns from the original covariance matrix $\mathbf{c}$ defined by Equation (7) with indices contained in I'.

For the ensemble classification of unlabeled test point $\mathbf{x}^*$, we follow three steps:

1. Choose $d' \leq d$ features at random from the original set to make up the reduced feature set, I';
2. Remove the corresponding $d - d'$ features from the test point $\mathbf{x}^*$ that do not belong to I';
3. Predict the label for the reduced-order version of $\mathbf{x}^*$ using the methods defined in Sections 3.2.1 or 3.2.2 with the reduced-order YADA model with joint pdf defined by Equation (19).

By repeating the above steps numerous times, we obtain an ensemble of label predictions, and we can interpret the empirical distribution over the class labels to be the probabilities that $\mathbf{x}^*$ belongs to each class.

3.3. Explanations

Recall Equation (16), where the numerator term $f^{(i)}(\mathbf{x}^*) \pi^{(i)}$ can be interpreted as the posterior joint likelihood that test point $\mathbf{x}^*$ comes from class i. The posterior marginal likelihoods are given by

$$f_j^{(i)}(x_j^*) \pi^{(i)}, \; j = 1, \ldots, d, \tag{20}$$

where $f_j^{(i)}$ is the marginal pdf of feature j defined by Equation (6), assuming class i, and x_j^* is the value of the jth feature at the test point. The values defined by Equation (20) approximately explain how each feature contributes to the posterior probability that $\mathbf{x}^*$ comes from class i. This is only an approximation; the exact relationship between the marginal and joint likelihoods is given by Equation (7), and is more complex.

One approach to explainability is to simply plot the scores defined by Equation (20). To illustrate, we consider a dataset containing measurements of 570 human cancer cells [35]. Thirty cell features are recorded, such as cell area and perimeter; each measurement has a label of either benign (B) or malignant (M). We train a YADA model for each class, and then use the marginal likelihoods defined by Equation (6) as a means for explaining class predictions. Figure 7 illustrates the marginal likelihoods of a test point with true label

M; the marginal likelihood that the point comes from both classes are illustrated for each feature. The likelihood that the point comes from class M is greater for nearly all 30 features, but most significantly for features 14 (smoothness_se) and 19 (fractal_dimension_se).
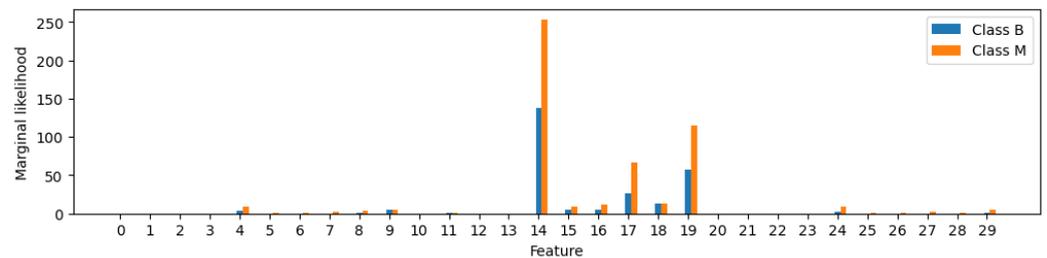
Figure 7. Marginal likelihoods of a test point from the cancer dataset with class label M.

A second approach to explainability using YADA, particularly useful when the number of features d is large, is to examine a subset of the d posterior marginal likelihoods, for example, a subset of the 10 largest posterior marginal likelihoods, or those that are larger than the average of all d posterior marginal likelihoods, because the members of this subset contribute more to the posterior probability. To illustrate, recall the MNIST dataset [34] introduced previously. We trained a YADA model for each class, then predicted labels for each image in the test set. The first two test images are illustrated in Figure 8. In each image, 14 pixels are highlighted (white); these are the features that contribute the most to the joint likelihood score. They can be interpreted as explaining which pixels were most important to the classifier. We observe that pixels near the outline of a digit are important, but pixels with zero values are also indicative.
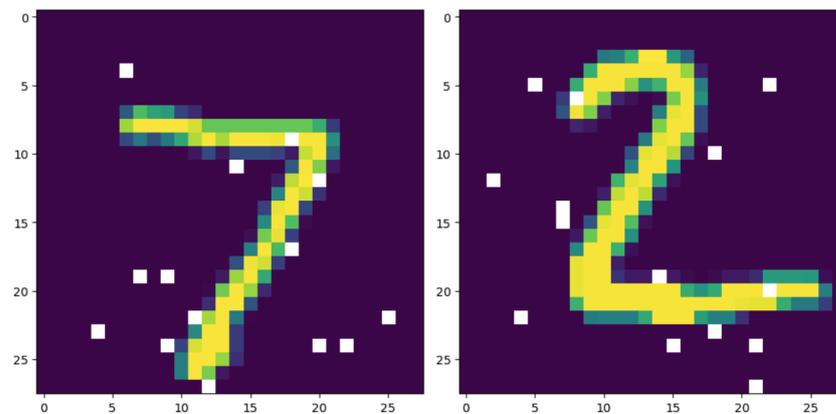
Figure 8. Explanations for two MNIST test images. The white pixels are the features that contribute the most to the joint likelihood score and can serve as explanations.

3.4. Prediction Confidence

Let $\mathbf{x}^*$ denote an unlabeled test point with predicted label y^*; this prediction can originate from YADA and the methods from Section 3.2, or from a completely separate ML model such as a trained neural network. There are a variety of sources of uncertainty that impact the outputs of ML models [36,37]. High uncertainty leads to low confidence in the predicted label. However, ML (and particularly DL) models are generally overconfident in their predictions [9,38]. We can use YADA to provide an independent confidence score for any test point.

In this section, we present a method to assess the confidence in the predicted label y^* that is a function of the distance of the test point to the training data. Computing the confidence values can be used to (1) quantify the uncertainty of a prediction from an ML model on inference data for tasks such as out-of-distribution (OOD) detection (when inference data are different from the data used for training) [9]; (2) identify outliers and

anomalies in training data that could have adverse effects when used for training [39]; and (3) detect concept drift when the target data have changed over time and invalidates the ML model [40]. As a method for detecting OOD data, we show that YADA is an effective method for measuring confidence; we utilize the Mahalanobis distance of $\mathbf{x}^*$ from a trained YADA model, as defined by Equation (18). Two approaches are considered.

3.4.1. Empirical Approach

One approach is to compute the Mahalanobis distance of $\mathbf{x}^*$ from the YADA model trained on the data with predicted class label y^*, then compare this distance with the population of distances of the training data from that same model. For example, if $y^* = i$ is the predicted class label, we would compare $m_i(\mathbf{x}^*)$ defined by Equation (18) with the population of distances $m_i(\{\mathbf{x}^{(i)}\})$ of all training data for class i. Intuitively, if $m_i(\mathbf{x}^*)$ is close to the mean of the population, we can have higher confidence in the predicted label y^* than if $m_i(\mathbf{x}^*)$ is in the tail of the population.

3.4.2. Approach Based on Theory

Let $\mathbf{G} = (G_1, \ldots, G_d)^T$ be a Gaussian random vector with mean vector $\boldsymbol{\mu}$ and covariance matrix $\mathbf{c}$. It is well known that the random variable $M^2 = (\mathbf{G} - \boldsymbol{\mu})^T \mathbf{c}^{-1}(\mathbf{G} - \boldsymbol{\mu})$ follows a chi-squared distribution with d degrees-of-freedom. For applications, we instead have samples of vector $\mathbf{G}$, which (approximately) follows a d-variate normal distribution, and we can compute the sample mean vector $\hat{\boldsymbol{\mu}}$ and sample covariance matrix $\hat{\mathbf{c}}$. This result provides a theoretical means of determining the likelihood that the Mahalanobis distance $m_i(\mathbf{x}^*)$ came from the theoretical distribution for the Mahalanobis distance for class i.

Using this approach, we can derive the following as a measure for the confidence that test point $\mathbf{x}^*$ comes from class i

$$\operatorname{conf}_i(\mathbf{x}^*) \propto \exp\left(-\frac{1}{2}\left(m_i(\mathbf{x}^*)^2 - d + 1\right)\right), \tag{21}$$

where d is the number of features in the model; this expression is based on the fact that the square of the Mahalanobis distance of a multivariate normal random vector follows the chi-squared distribution with d degrees-of-freedom.

3.4.3. Applications

To illustrate these concepts for prediction confidence, we considered three different applications. First, we trained three YADA models on the Fisher iris dataset, one model per class label. We then treated each of the 150 data points as test points, and classified them using the likelihood approach. The confidence in these predictions as computed by Equation (21) are illustrated by Figure 9.
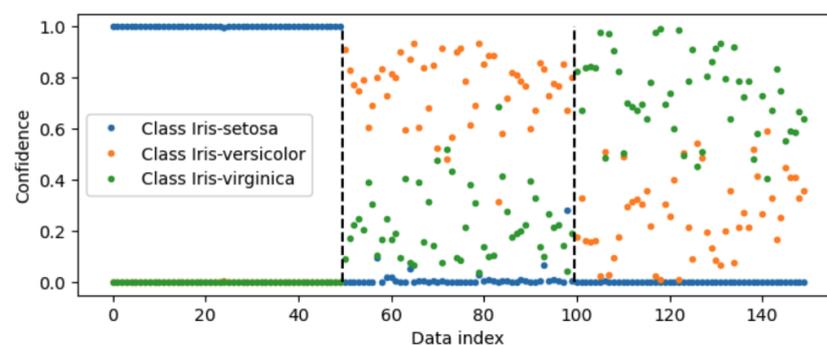
Figure 9. Prediction confidence $\operatorname{conf}_i(\mathbf{x}^*)$ defined by Equation (21) for Fisher iris data.

The data are ordered so that points 1–50 are from class Iris-setosa, points 51–100 are from Iris-versicolor, and points 101–150 are from class Iris-virginica. For points 1–50, the

prediction confidence that these points come from the Iris-setosa class is very high, and the confidence for the other two classes is near zero. For points 51–100, the prediction confidence is, in general, the greatest for the (correct) Iris-versicolor class, but the confidence that these points come from Iris-virginica is not zero, so there is some uncertainty in the predicted class labels for these points. Similar results can be observed for points 100–150.

A second example is based on the MNIST dataset of images of handwritten digits 0, 1, . . . , 9 [34]. We trained a YADA model for each class, then predicted labels for each image in the test set; we also computed confidence values for each predicted label. The five test images with the greatest confidence values are illustrated by the top row in Figure 10. The bottom row illustrates the test images of a '2' and '6' that had the greatest (left) and least (right) confidence values.
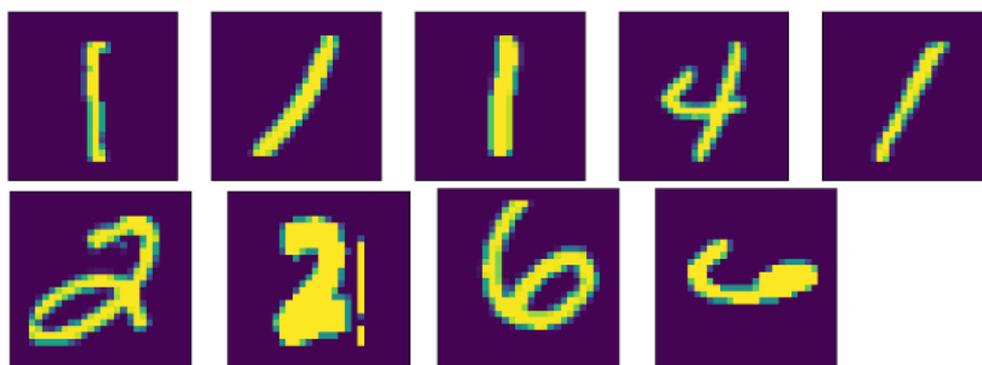
Figure 10. Prediction confidence for the MNIST dataset. The top row illustrates the 5 test images with the greatest confidence values. The bottom row illustrates the test images with the greatest and least confidence for both the '2' class (left two images) and the '6' class (right two images).

For a third example, we utilize a DL model (ResNet50 [41]) pre-trained on the CIFAR-10 dataset and compare how well YADA detects OOD data. CIFAR-10 [42] is a 10-class dataset for general object detection containing 50,000 training samples. For this study, data from CIFAR-10 is considered to be in-distribution and any other data are OOD. We then consider the following benchmark object recognition datasets in an OOD study: (1) CIFAR-100 [42], which is similar to CIFAR-10 except that it has 100 classes; (2) Tiny ImageNet (TIN) [43], an object detection dataset that is a subsample of the ImageNet database of over 14 million images (with any intersection of images from CIFAR-10 removed); (3) MNIST [44]; (4) SVHN [45], a dataset of mostly house numbers representing digits in natural images; (5) Texture [46], a collection of textural images in the wild; and (6) Places365 [47], a scene recognition dataset. We apply the pre-trained DL model and framework for executing experiments from OpenOOD [48] that compare other state-of-the-art OOD detection methods. In particular, we use the values from the last layer in the neural network as input to the OOD detection methods.

We limited our approach to state-of-the-art OOD detection methods. Other probabilistic methods could have been used but would require a higher computational overhead. Thus, rather than increasing the computational overhead, we chose to focus on the selected methods. Other DL models could have been used and may provide some variation in results. However, the provided example highlights the ability of YADA to quantify the uncertainty of OOD data with similar performance as other state-of-the-art techniques.

We compute the Mahalanobis distance as our confidence measure and use it for OOD detection in YADA and compare against the following: traditional Mahalanobis distance (MDist) [49], Virtual Logit Matching (ViM) [50], and Deep Nearest Neighbor (DNN) [51]. YADA models trained using both the empirical distribution and a kernel density estimator are considered. The results are summarized in Table 1, which correspond to area under the corresponding ROC curve. In all cases except for the Texture dataset, YADA outperforms the traditional Mahalanobis distance (MDist), suggesting that transforming the data to the

MVN space (meeting the assumptions of Mahalanobis distance) is beneficial. YADA is competitive with the other OOD methods considered.

Table 1. AUC values for detecting OOD data. CIFAR-10 is considered in-distribution against several OOD datasets. Across all datasets, YADA is competitive with state-of-the-art approaches.

	MDist	**ViM**	**DNN**	**YADA with Empirical cdf**	**YADA with KDE**
CIFAR-100	85.95	87.45	89.75	87.35	88.31
TIN	87.63	89.68	91.71	89.32	87.89
MNIST	88.61	94.27	94.41	91.97	96.62
SVHN	91.83	94.48	93.01	90.96	92.72
Texture	93.78	94.77	93.02	91.63	86.86
Places365	86.63	89.19	92.10	90.02	88.76

3.5. Synthetic Data Generation

Synthetic data have a variety of uses, such as creating additional training or testing data, or to compensate for label and/or feature imbalance. In particular, YADA can be used to produce synthetic but realistic images for image-related applications. Because YADA is a probabilistic model, it is straightforward to generate synthetic data. There are several ways to perform this, as will be described in the following sections.

Recall from Section 3.1 that we can take two approaches to train a YADA model. First, we can partition the data according to class label, which produces one trained YADA model for each class. To produce synthetic data for this case, we apply the procedures described below for each YADA model separately. Second, we train a single YADA model on all the available data regardless of class. In this approach, we simply treat the label as an additional "feature" and build and train a single model on $d + 1$ features.

3.5.1. Random Sampling

The first approach to creating synthetic data is by simple random sampling, where samples are drawn independently at random from the trained joint distribution of a YADA model, i.e., Equation (7). This is the most straightforward and probably the most common way to produce synthetic data. An algorithm to create random samples of a YADA model requires two steps:

1. Create independent samples of a d-variate Gaussian vector with zero mean and covariance matrix $\mathbf{c}$;
2. Map the samples of $\{G_i\}$ to samples of $\{X_i\}$ using Equation (3).

There are numerous methods to execute step (1) provided by various statistical packages.

3.5.2. Conditional Sampling

There may be scenarios where we want to create synthetic data with one or more of the features held constant at a fixed value. For example, we may want to produce synthetic images but know that pixels along the boundary should always be of one color; a conditional sampling approach can be used to achieve this.

Suppose the vector $\mathbf{X}$ of d features is re-ordered so that all $d - m$ features with known fixed values are collected into vector $\mathbf{Z}$ and the remaining m features are collected into vector $\mathbf{Y}$, i.e., $\mathbf{X} = (\mathbf{Y}, \mathbf{Z})$. An algorithm to create samples of $\mathbf{Y}|\mathbf{Z} = \mathbf{z}$ requires six steps:

1. Re-arrange the columns and rows of the covariance matrix $\mathbf{c}$ to be consistent with the re-ordered feature vector;
2. Partition $\mathbf{c}$ as described in Section 2.2.5 to create sub-matrices $\mathbf{c_{YY}}$, $\mathbf{c_{YZ}}$, and $\mathbf{c_{ZZ}}$;
3. Compute $\mathbf{w_Z}$, the Gaussian image of $\mathbf{z}$, using Equation (2);
4. Create samples of an m-variate Gaussian vector with mean $\mathbf{c_{YZ}} \, \mathbf{c_{ZZ}^{-1}} \, \mathbf{w_Z}$ and covariance matrix $\mathbf{c_{YY}} - \mathbf{c_{YZ}} \, \mathbf{c_{ZZ}^{-1}} \, \mathbf{c_{YZ}^{T}}$;
5. Map these samples to samples of $(Y_1, \ldots, Y_m)^T$ using Equation (3);

6. Assemble the full feature vector by appending the fixed values **z**, and then retract the re-order step.

3.5.3. High Probability Sampling

For some applications, it may be of interest to create synthetic data from samples that occur with high probability, i.e., samples "near" the mean of the joint distribution instead of out in the tails. Let $p \in [0,1]$ be the target probability value. The idea is to modify the algorithm for the random sampling presented above to include an intermediate step, where we compute the Mahalanobis distance of each sample from the origin, and keep only those samples with distances less than $r^2 = (\chi_d^2)^{-1}(p)$, where χ_d^2 denotes the cdf of the chi-squared distribution with d degrees-of-freedom. Hence, if p is close to zero, many samples are rejected and only those very close to the origin are retained. Likewise, all samples are retained when $p = 1$, i.e., random sampling.

An algorithm to create p-probability samples requires five steps:

1. Create one sample of a d-variate Gaussian vector with zero mean and covariance matrix **c**;
2. Compute m^2, the square of the Mahalanobis distance of this sample to the origin;
3. Retain the sample if $m^2 \leq r^2$; otherwise, reject it;
4. Repeat steps 1–3 until the desired number of samples have been retained;
5. Map the retained samples of $\{G_i\}$ to samples of $\{X_i\}$ using Equation (3).

3.5.4. Applications

In this section, we illustrate the different sampling methods for several example applications. First, consider again the cancer cell data [35] introduced in Section 3.3. Here, we trained a YADA model for class B and another for class M, then used them to create 500 synthetic data points; random sampling was used. Figure 11 illustrates scatter plots for six random pairings of the thirty available features. Each panel illustrates both the 569 training data ('x') and 500 synthetic data ('o') for both benign (B) and malignant (M) class labels.
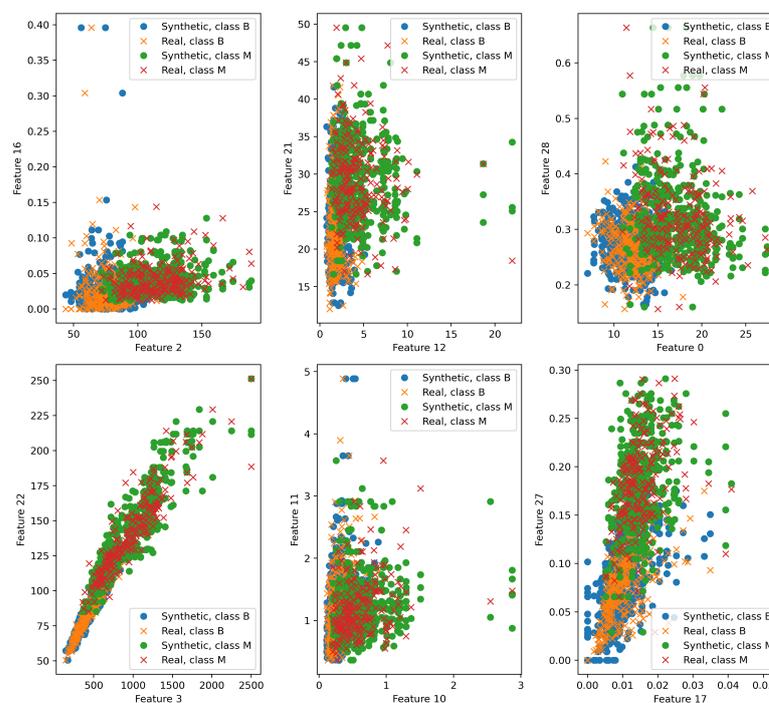
Figure 11. Scatter plots of the cancer dataset for 6 random pairings of the 30 available features. Each panel illustrates the 569 real training data ('×') and 500 synthetic data ('•') for both benign ('B') and malignant ('M') class labels.

As a second example, recall the MNIST dataset of images of handwritten digits 0, 1, . . . , 9 [34]. We trained a YADA model for each class, then used them to create synthetic images of handwritten digits. Within the dataset, there are numerous features (pixels) that are blank for all images considered (e.g., the four corners of an image). Conditional sampling was therefore utilized to ensure that the synthetic images also had these same blank pixels. Figure 12 illustrates some of the results; original training images are shown in the top row, and synthetic images of the same class are shown in the bottom row. These results demonstrate that capturing the marginal distributions and pairwise correlations of the training data is sufficient to produce adequate synthetic images of handwritten digits.
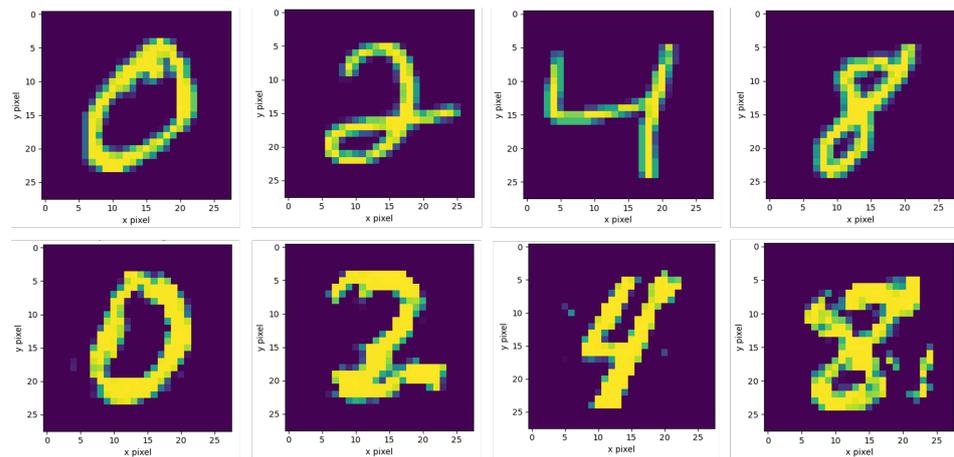
Figure 12. Images of handwritten digits: MNIST training data (**top** row) and synthetic images produced by YADA (**bottom** row).

A third example illustrates high probability sampling; Figure 13 illustrates 500 synthetic data from YADA models trained to the Fisher iris dataset. From left to right, the scatter plots show data for two of the features assuming probability values $p = 1$, $p = 0.5$, $p = 0.1$, and $p = 0.005$. As p approaches zero, the samples approach increasingly closer to the sample mean, denoted by a black 'x'; the case wherein $p = 1$ is identical to simple random sampling.
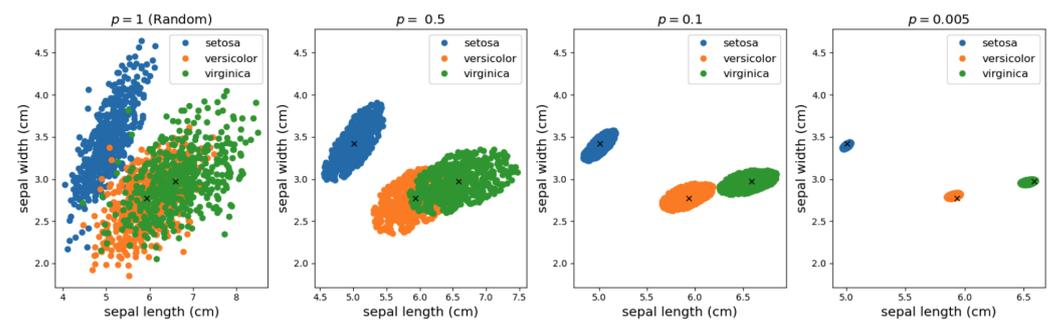
Figure 13. Synthetic data using high probability sampling for the Fisher iris data.

As a final example, we consider the bone marrow transplant dataset [52] containing 36 features from 187 pediatric patients, including age, body mass, dosage, and recovery time. An MLP classifier was trained on these data to predict patient survival status; the mean accuracy of the classifier for a 5-fold cross validation analysis was 70%. We also trained a YADA model on these data and applied random sampling to produce an additional 250 synthetic training data. Once re-trained, the mean accuracy of the classifier improved to 83%.

4. Conclusions

While deep learning (DL) methods have enjoyed much success in recent years, they remain overly complex, over-parameterized, overconfident in predicting unknown labels, and difficult to explain and/or interpret. This motivated our work on a probabilistic model for ML applications we refer to as Yet Another Discriminant Analysis (YADA). YADA is a general-purpose probabilistic model that can be used for a wide variety of ML tasks that DL models do not perform well, including (1) providing explanations; (2) quantifying the uncertainty in a prediction; and (3) out-of-distribution detection. Further, YADA has few parameters and provides a mechanism to predict labels with explanations while also providing an associated confidence score from a single model (typically, predictions, explanations, and confidence are not all provided by a single model). YADA can represent continuous or discrete-valued features with arbitrary marginal distributions that exhibit nonzero correlations. It is based on a mathematically rigorous foundation so that joint distributions, joint likelihood functions, joint entropy, K-L divergence, and other properties are readily derived. In addition, realistic synthetic data can be generated using YADA in a variety of ways. The YADA algorithm is based on the translation random vector or Gaussian copula, which has seen widespread use in finance, engineering, climate science, and other disciplines, but its use for ML applications remains limited. As mentioned, YADA can be less accurate for classification tasks than many DL models and, for this reason, we do not make any direct comparisons on classification accuracy.

We view YADA as a first step in developing probabilistic models based on a mathematically rigorous foundation, and see several areas for improvement and future work. Currently, YADA operates on raw features, whereas DL algorithms are able to extract features through the layers of a neural network. Integrating feature learning could help improve the performance of YADA. Also, YADA is limited based on only modeling correlations between pairs of features. Improving modeling more complex feature interactions would increase the fidelity of YADA. We also mention that DL has received significant engineering research to improve hardware and software stacks the make training and inference more efficient. Similar gains would be helpful for scaling YADA, which is limited by inverting the covariance matrix.

Our intention is that this paper provides a foundation for further improvement in mathematically based probabilistic models that are simple enough to understand rather than start from overly complex models and trying to extract meaning from them. This paper serves as an alternative approach starting from more simplistic results that are largely overlooked by the ML community.

Author Contributions: R.V.F.J. contributed to conceptualization, methodology, software, validation, formal analysis, and all writing. M.R.S. contributed to conceptualization, methodology, software, validation, writing—review and editing, project administration, and funding acquisition. J.B.I. contributed to software and validation. E.J.W. contributed to software and validation. All authors have read and agreed to the published version of the manuscript.

Data Availability Statement: We utilized datasets that are readily available to the public, including (1) the Fisher iris dataset [53] designed to quantify the morphological variation in iris flowers of 3 related species; (2) the Palmer Archipelago penguin dataset [54], consisting of 5 different measurements of 344 penguins of 3 different species; (3) a dataset containing 30 measurements of 570 human cancer cells [35] to determine whether the cells in the dataset are benign or malignant; (4) a linguistics dataset of phonemes [30], which is the smallest unit of speech distinguishing 1 word (or word element) from another; and (5) the bone marrow transplant dataset [52] containing 36

features from 187 pediatric patients, including age, body mass, dosage, and recovery time. Several image datasets were also studied, including (6) The Modified National Institute of Standards and Technology (MNIST) image dataset of handwritten digits [34]; (7) CIFAR-10 [42], a 10-class dataset for general object detection containing 50,000 training samples; (8) CIFAR-100 [42], which is similar to CIFAR-10 except that it has 100 classes; (9) Tiny ImageNet (TIN) [43], an object detection dataset that is a subset of ImageNet; (10) SVHN [45], a dataset of mostly house numbers representing digits in natural images; (11) Texture [46], which contains a collection of textural images in the wild; and (12) Places365 [47], a general scene recognition dataset.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Discrete-Valued Random Variables with YADA

Let X be a discrete random variable that takes values $x_1 < \cdots < x_r$ with probabilities $p_1, \ldots, p_r$, where $\sum_{i=1}^{r} p_i = 1$. In this section, we show that X can be expressed in terms of $G \sim N(0,1)$, a standard Gaussian random variable, as was performed in Sections 2.1 and 2.2, assuming X was continuous. In particular, we develop results that are analogous to Equations (1), (2), and (6) below.

The cdf and probability mass function (pmf) of X are given by

$$F(x) = \sum_{i=1}^{r} p_i \mathbb{1}(x \geq x_i) \quad \text{and} \quad f(x) = \sum_{i=1}^{r} p_i \mathbb{1}(x = x_i), \tag{A1}$$

respectively, where $\mathbb{1}(A)$ is the indicator function, equal to one if event A is true and equal to zero otherwise; $f(x)$, defined by Equation (A1), is a version of Equation (6) for discrete variables. It follows that we can express X as

$$X = h(G) = \sum_{i=1}^{r} x_i \mathbb{1}(G \in \beta_i), \tag{A2}$$

where $\beta_i = (a_{i-1}, a_i]$ are sets on the real line with convention $\beta_1 = (-\infty, a_1]$ and $\beta_r = (a_{r-1}, \infty)$. We can think of $\{\beta_i\}$ and $\{a_i\}$ as bins and bin boundaries, respectively, with boundaries defined by

$$a_i = \Phi^{-1}(p_1 + \cdots p_i), \ i = 1, \ldots, r - 1.$$

As proof that Equation (A2) is a valid representation for X, note that

$$\begin{aligned}
\Pr(X = x_i) &= \Pr(h(G) = x_i) \\
&= \Pr(G \in \beta_i) \\
&= \Phi(a_i) - \Phi(a_{i-1}) \\
&= (p_1 + \cdots p_i) - (p_1 + \cdots p_{i-1}) \\
&= p_i, \ i = 1, \ldots, r - 1.
\end{aligned}$$

For the special case of $i = r$, $\Pr(X = x_r) = \Pr(G \in \beta_r) = 1 - \sum_{i=1}^{r-1} p_i = p_r$.

The mapping defined by Equation (A2) is a quantization of the Gaussian distribution and represents a version of Equation (1) for arbitrary discrete variables. For example, take $r = 2$ with $x_1 = 0$, $x_2 = 1$, $p_1 = q$, and $p_2 = 1 - q$. Then, by Equation (A2), $X = \mathbb{1}(G > a)$ is a binomial random variable with parameter $q \in (0,1)$, where $a = \Phi^{-1}(p)$.

We interpret the inverse of Equation (A2) as

$$G = w(X) = \sum_{i=1}^{r} \tilde{G}_i \mathbb{1}(X = x_i), \tag{A3}$$

where $\tilde{G}_i$ is a truncated normal random variable with support β_i, that is,

$$\Pr(\tilde{G}_i \leq z) = \frac{1}{p_i}(\Phi(z) - \Phi(a_{i-1})), \; z \in \beta_i = (a_{i-1}, a_i].$$

This is a version of Equation (2) for the case where X is a discrete random variable.

Appendix B. Derivative of $w(x)$

Recall that $w(x) = h^{-1}(x)$ is the inverse mapping introduced by Equation (2) for the case of continuous-valued features. The joint pdf of the feature vector defined by Equation (7) depends on $w'(x)$, the derivative of this function, which we derive below.

First, note that $\Phi \colon \mathbb{R} \to (0,1)$ and $\Phi^{-1} \colon (0,1) \to \mathbb{R}$, the cdf of the $N(0,1)$ random variable and its inverse, are strictly increasing continuous bijective functions. If we let $z = \Phi^{-1}(y)$, then $y = \Phi(z)$ and

$$\frac{\mathrm{d}y}{\mathrm{d}z} = \Phi'(z) = \phi(z),$$

where $\phi(z)$ denotes the pdf of the $N(0,1)$ random variable. Then,

$$\frac{\mathrm{d}}{\mathrm{d}y}\Phi^{-1}(y) = \frac{\mathrm{d}z}{\mathrm{d}y} = \frac{1}{\phi(z)} = \frac{1}{\phi \circ \Phi^{-1}(y)}$$

It follows (by the chain rule) that

$$w'(x) = (\Phi^{-1} \circ F(x))' = ((\Phi^{-1})' \circ F(x)) \cdot F'(x) = \frac{f(x)}{\phi \circ \Phi^{-1}(F(x))}$$

$$= \frac{f(x)}{\phi(w(x))}, \tag{A4}$$

where f and F are the pdf and cdf of random variable X defined by Equation (1). Further, $w'(x) \geq 0$ because both f and ϕ are pdfs and, therefore, non-negative.

Appendix C. Tail Modeling with YADA

Suppose we have samples $x_1, \ldots, x_n$ of random variable X, and assume that the samples have been ordered in ascending order. The empirical cdf built from these data is equal to zero for all $x < x_1$ and equal to one for all $x \geq x_n$. Sometimes, it is useful to use models that allow us to extrapolate outside of the interval defined by the observed data, i.e., for $x < x_1$ and/or $x > x_n$, the left and right tails of the distribution of X.

We introduce a method, referred to as tail modeling, to extrapolate outside of the observed data. The idea is simply to choose a distribution with a known functional form for the left and/or right tails, and calibrate these distributions using the available data to enforce continuity at $x = x_1$ and $x = x_n$.

We consider three different tail models, selected because they offer different rates of decay in the tails; other tail models can of course be added if necessary. The tail models considered include the following:

- Gaussian/normal tails, with pdf and cdf:

$$f_G(x; \mu, \sigma) = \frac{1}{\sigma}\phi\left(\frac{x - \mu}{\sigma}\right) \quad \text{and} \quad F_G(x; \mu, \sigma) = \Phi\left(\frac{x - \mu}{\sigma}\right), \tag{A5}$$

where μ and $\sigma > 0$ are parameters, and

$$\phi(x) = \frac{1}{\sqrt{2\pi}}e^{-x^2/2} \quad \text{and} \quad \Phi(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{-u^2/2}\,\mathrm{d}u$$

denote the pdf and cdf, respectively, for the standard normal random variable. We note that the tails of the normal pdf decay to zero as e^{-x^2} for $x \to \pm\infty$.

- Exponential tails, with pdf and cdf:

$$f_E(x; \mu, b) = \frac{1}{2b} e^{-|x-\mu|/b} \quad \text{and}$$

$$F_E(x; \mu, b) = \frac{1}{2} + \frac{1}{2}\text{sgn}(x - \mu)\left(1 - e^{-|x-\mu|/b}\right), \tag{A6}$$

where μ and $b > 0$ are parameters. The tails of this pdf decay to zero as $e^{-|x|}$ for $x \to \pm\infty$, which is slower than the decay rate of the Gaussian pdf.

- Log-normal tails, with pdf and cdf:

$$f_L(x; \mu, \sigma, c) = \left(\frac{1}{2\sigma|x-c|}\right)\phi\left(\frac{\ln(|x-c|) - \mu}{\sigma}\right)$$

$$F_L(x; \mu, \sigma, c) = \frac{1}{2}\left[1 + \text{sgn}(x - c)\Phi\left(\frac{\ln(|x-c|) - \mu}{\sigma}\right)\right], \tag{A7}$$

where μ, c, and $\sigma > 0$ are parameters. The tails of this pdf decay at a rate in between that of the Gaussian and exponential pdfs.

Given the data $x_1, \ldots, x_n$, we implement these tail models as follows:

1. Use the empirical cdf for $x_1 \leq x < x_n$;
2. Compute $\hat{\mu}$, the sample mean of the data;
3. For any values $x < x_1$ (the left tail), set the pdf and cdf to f_1 and F_1 as specified in Table A1;
4. For any values $x \geq x_n$ (the right tail), set the pdf and cdf to f_n and F_n as specified in Table A1.

Note that we can choose to model only the left tail, only the right tail, or both. Further, the functional forms of the left and right tail models need not be identical.

To illustrate the tail modeling approach, suppose we have $n = 20$ random samples of X. The cdf and left tail models are illustrated by the left panel in Figure A1; a log scale is applied to increase the visibility of the tail. The complementary cdf and right tail models are shown in the right panel of Figure A1. The blue line indicates the empirical cdf, which drops immediately to zero for $x < x_1 \approx -3.1$ and for $x > x_n \approx 1.2$. The three tail models allow for extrapolating the cdf outside of the range $[x_1, x_n]$. The exponential model (green) exhibits the slowest decay to zero, followed by the log-normal (red) and Gaussian (orange) models.

Table A1. The left and right tail models for the pdf and cdf.

Type	$f_1(x), x < x_1$	$f_n(x), x > x_n$	$F_1(x), x < x_1$	$F_n(x), x > x_n$	Parameters
Gaussian	$f_G(x; \hat{\mu}, \sigma_1)$	$f_G(x; \hat{\mu}, \sigma_n)$	$F_G(x; \hat{\mu}, \sigma_1)$	$F_G(x; \hat{\mu}, \sigma_n)$	$\sigma_1 = \dfrac{x_1 - \hat{\mu}}{\Phi^{-1}\left(\frac{1}{n}\right)}, \sigma_n = \dfrac{x_n - \hat{\mu}}{\Phi^{-1}\left(1 - \frac{1}{2n}\right)}$
Exponential	$f_E(x; \hat{\mu}, b_1)$	$f_E(x; \hat{\mu}, b_n)$	$F_E(x; \hat{\mu}, b_1)$	$F_E(x; \hat{\mu}, b_n)$	$b_1 = \dfrac{x_1 - \hat{\mu}}{\ln\left(\frac{2}{n}\right)}, b_n = \dfrac{x_n - \hat{\mu}}{\ln(n)}$
Log-normal	$f_L(x; \hat{\mu}, b_1)$	$f_L(x; \hat{\mu}, b_n)$	$F_L(x; \hat{\mu}, b_1)$	$F_L(x; \hat{\mu}, b_n)$	$b_1 = \dfrac{x_1 - \hat{\mu}}{\ln\left(\frac{2}{n}\right)}, b_n = \dfrac{x_n - \hat{\mu}}{\ln(n)}$
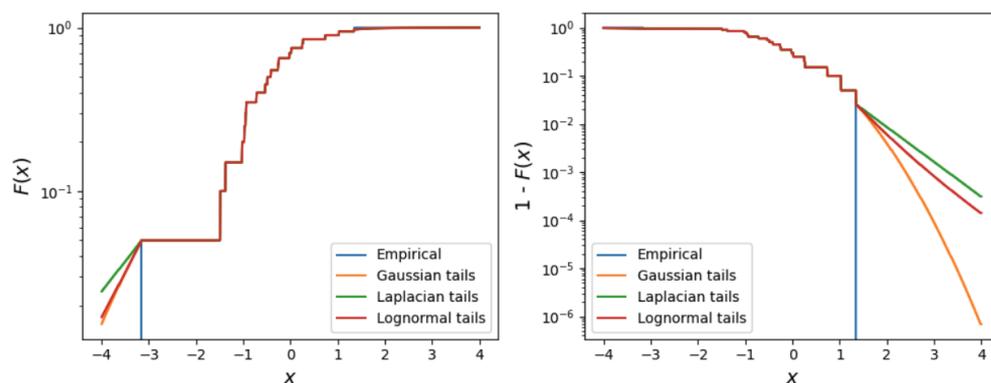
Figure A1. Example models for the left and right tails of the cdf.

References

1. Tang, H.; Houthooft, R.; Foote, D.; Stooke, A.; Xi Chen, O.; Duan, Y.; Schulman, J.; DeTurck, F.; Abbeel, P. # Exploration: A study of count-based exploration for deep reinforcement learning. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17), Long Beach, CA, USA, 4–9 December 2017; pp. 2750–2759.
2. Yamada, I.; Asai, A.; Shindo, H.; Takeda, H.; Matsumoto, Y. LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention. *arXiv* **2020**, arXiv:2010.01057. [CrossRef]
3. Li, C.; Li, L.; Geng, Y.; Jiang, H.; Cheng, M.; Zhang, B.; Ke, Z.; Xu, X.; Chu, X. YOLOv6 v3.0: A Full-Scale Reloading. *arXiv* **2023**, arXiv:2301.05586. [CrossRef]
4. Dumitru, R.G.; Peteleaza, D.; Craciun, C. Using DUCK-Net for polyp image segmentation. *Sci. Rep.* **2023**, *13*, 9803. [CrossRef] [PubMed]
5. Selvaraju, R.R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; Batra, D. Grad-CAM: Visual Explanations From Deep Networks via Gradient-Based Localization. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017.
6. Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and Harnessing Adversarial Examples. *arXiv* **2015**, arXiv:1412.6572. [CrossRef]
7. Ilyas, A.; Santurkar, S.; Tsipras, D.; Engstrom, L.; Tran, B.; Madry, A. Adversarial examples are not bugs, they are features. In Proceedings of the 33rd International Conference on Neural Information Processing Systems (NIPS'19), Vancouver, BC, Canada, 8–14 December 2019; pp. 125–136.
8. Guo, C.; Pleiss, G.; Sun, Y.; Weinberger, K.Q. On calibration of modern neural networks. In Proceedings of the International Conference on Machine Learning. PMLR, Sydney, Australia, 6–11 August 2017; pp. 1321–1330.
9. Hendrycks, D.; Mazeika, M.; Dietterich, T. Deep Anomaly Detection with Outlier Exposure. *arXiv* **2019**, arXiv:1812.04606. [CrossRef]
10. Ribeiro, M.T.; Singh, S.; Guestrin, C. Model-Agnostic Interpretability of Machine Learning. *arXiv* **2016**, arXiv:1606.05386. [CrossRef]
11. Lundberg, S.M.; Erion, G.G.; Lee, S.I. Consistent Individualized Feature Attribution for Tree Ensembles. *arXiv* **2019**, arXiv:1802.03888. [CrossRef]
12. Jobin, A.; Ienca, M.; Vayena, E. The global landscape of AI ethics guidelines. *Nat. Mach. Intell.* **2019**, *1*, 389–399. [CrossRef]
13. Gal, Y.; Ghahramani, Z. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In Proceedings of the International Conference on Machine Learning. PMLR, New York, NY, USA, 20–22 June 2016; pp. 1050–1059.
14. Lakshminarayanan, B.; Pritzel, A.; Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17), Long Beach, CA, USA, 4–9 December 2017; pp. 6405–6416.
15. Grigoriu, M. Crossings of Non-Gaussian Translation Processes. *J. Eng. Mech.* **1984**, *110*, 610–620. [CrossRef]
16. Arwade, S.R. Translation vectors with non-identically distributed components. *Probabilistic Eng. Mech.* **2005**, *20*, 158–167. [CrossRef]
17. Elidan, G. Copulas in Machine Learning. In *Proceedings of the Copulae in Mathematical and Quantitative Finance, Cracow, Poland, 10–11 July 2013*; Jaworski, P., Durante, F., Härdle, W.K., Eds.; Springer: Berlin/Heidelberg, Germany, 2013; pp. 39–60.
18. Kosowski, R.L.; Neftci, S.N. *Principles of Financial Engineering*, 3rd ed.; Academic Press: New York, NY, USA, 2015.
19. Targino, R.S.; Peters, G.W.; Shevchenko, P.V. Sequential Monte Carlo Samplers for capital allocation under copula-dependent risk models. *Insur. Math. Econ.* **2015**, *61*, 206–226. [CrossRef]
20. Lapuyade-Lahorgue, J.; Xue, J.H.; Ruan, S. Segmenting Multi-Source Images Using Hidden Markov Fields With Copula-Based Multivariate Statistical Distributions. *IEEE Trans. Image Process.* **2017**, *26*, 3187–3195. [CrossRef] [PubMed]
21. Qian, D.; Wang, B.; Qing, X.; Zhang, T.; Zhang, Y.; Wang, X.; Nakamura, M. Drowsiness Detection by Bayesian-Copula Discriminant Classifier Based on EEG Signals During Daytime Short Nap. *IEEE Trans. Biomed. Eng.* **2017**, *64*, 743–754. [CrossRef]

22. Meyer, D.; Nagler, T.; Hogan, R.J. Copula-based synthetic data augmentation for machine-learning emulators. *Geosci. Model Dev.* **2021**, *14*, 5205–5215. [CrossRef]

23. Schölzel, C.; Friederichs, P. Multivariate non-normally distributed random variables in climate research - Introduction to the copula approach. *Nonlinear Process. Geophys.* **2008**, *15*, 761–772. [CrossRef]

24. Field, R.V., Jr.; Constantine, P.; Boslough, M. Statistical surrogate models for prediction of high-consequence climate change. *Int. J. Uncertain. Quantif.* **2013**, *3*, 341–355. [CrossRef]

25. Yuan, Z.; Wang, J.; Worrall, D.M.; Zhang, B.B.; Mao, J. Determining the Core Radio Luminosity Function of Radio AGNs via Copula. *Astrophys. J. Suppl. Ser.* **2018**, *239*, 33. [CrossRef]

26. Carrillo, J.A.; Nieto, M.; Velez, J.F.; Velez, D. A New Machine Learning Forecasting Algorithm Based on Bivariate Copula Functions. *Forecasting* **2021**, *3*, 355–376. [CrossRef]

27. Gawlikowski, J.; Tassi, C.R.N.; Ali, M.; Lee, J.; Humt, M.; Feng, J.; Kruspe, A.; Triebel, R.; Jung, P.; Roscher, R.; et al. A survey of uncertainty in deep neural networks. *Artif. Intell. Rev.* **2023**, *56*, 1513–1589. [CrossRef]

28. Ganian, R.; Korchemna, V. The complexity of Bayesian network learning: Revisiting the superstructure. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 430–442.

29. Nataf, A. Determination des distribution don't les marges sont donnees. *Comptes Rendus L'AcadÉMie Des Sci.* **1962**, *225*, 42–43.

30. The Phoneme Dataset. Available online: https://www.kaggle.com/datasets/timrie/phoneme (accessed on 19 September 2023).

31. Grigoriu, M. *Applied Non-Gaussian Processes*; P T R Prentice-Hall: Englewood Cliffs, NJ, USA, 1995.

32. Field, Jr., R.V.; Grigoriu, M. A method for the efficient construction and sampling of vector-valued translation random fields. *Probabilistic Eng. Mech.* **2012**, *29*, 79–91. [CrossRef]

33. Papoulis, A.; Pillai, S.U. *Probability, Random Variables, and Stochastic Processes*, 4th ed.; McGraw-Hill, Inc.: New York, NY, USA, 2002.

34. The MNIST Dataset. Available online: https://www.kaggle.com/datasets/hojjatk/mnist-dataset (accessed on 29 October 2024).

35. The Cancer Dataset. Available online: https://www.kaggle.com/datasets/erdemtaha/cancer-data (accessed on 19 September 2023).

36. Hüllermeier, E.; Waegeman, W. Aleatoric and Epistemic Uncertainty in Machine Learning: An Introduction to Concepts and Methods. *Mach. Learn.* **2021**, *110*, 457–506. [CrossRef]

37. Stracuzzi, D.J.; Chen, M.G.; Darling, M.C.; Peterson, M.G.; Vollmer, C. *Uncertainty Quantification for Machine Learning*; Technical Report SAND2017-6776; Sandia National Laboratories: Albuquerque, NM, USA, 2017.

38. Wei, H.; Xie, R.; Cheng, H.; Feng, L.; An, B.; Li, Y. Mitigating neural network overconfidence with logit normalization. In Proceedings of the International Conference on Machine Learning. PMLR, Baltimore, MD, USA, 17–23 July 2022; pp. 23631–23644.

39. Smith, M.R.; Martinez, T. Improving classification accuracy by identifying and removing instances that should be misclassified. In Proceedings of the 2011 International Joint Conference on Neural Networks, San Jose, CA, USA, 31 July–5 August 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 2690–2697.

40. Lu, J.; Liu, A.; Dong, F.; Gu, F.; Gama, J.; Zhang, G. Learning under concept drift: A review. *IEEE Trans. Knowl. Data Eng.* **2018**, *31*, 2346–2363. [CrossRef]

41. Targ, S.; Almeida, D.; Lyman, K. Resnet in Resnet: Generalizing Residual Architectures. *arXiv* **2016**, arXiv:1603.08029. [CrossRef]

42. Krizhevsky, A.; Hinton, G. *Learning Multiple Layers of Features from Tiny Images*; Technical Report 0; University of Toronto: Toronto, ON, Canada, 2009.

43. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS'12)—Volume 1, Lake Tahoe, NV, USA, 3–6 December, 2012, pp. 1097–1105.

44. Deng, L. The MNIST database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Process. Mag.* **2012**, *29*, 141–142. [CrossRef]

45. Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; Ng, A.Y. Reading digits in natural images with unsupervised feature learning. In Proceedings of the NIPS Workshop on Deep Learning and Unsupervised Feature Learning, Granada, Spain, 12–17 December 2011; p. 4.

46. Kylberg, G. The Kylberg Texture Dataset v. 1.0. *External Report (Blue Series) 35*; Centre for Image Analysis, Swedish University of Agricultural Sciences and Uppsala University: Uppsala, Swiden, 2014.

47. Zhou, B.; Lapedriza, A.; Khosla, A.; Oliva, A.; Torralba, A. Places: A 10 million image database for scene recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *40*, 1452–1464. [CrossRef]

48. Yang, J.; Wang, P.; Zou, D.; Zhou, Z.; Ding, K.; Peng, W.; Wang, H.; Chen, G.; Li, B.; Sun, Y.; et al. Openood: Benchmarking generalized out-of-distribution detection. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 32598–32611.

49. Lee, K.; Lee, K.; Lee, H.; Shin, J. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18), Montreal, QC, Canada, 3–8 December 2018; pp. 7167–7177.

50. Wang, H.; Li, Z.; Feng, L.; Zhang, W. Vim: Out-of-distribution with virtual-logit matching. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 4921–4930.

51. Sun, Y.; Ming, Y.; Zhu, X.; Li, Y. Out-of-distribution detection with deep nearest neighbors. In Proceedings of the International Conference on Machine Learning. PMLR, Baltimore, MD, USA, 17–23 July 2022; pp. 20827–20840.

52. Sikora, M.; Wróbel, L.; Gudyś, A. Bone Marrow Transplant: Children. UCI Machine Learning Repository. *Clin. Transplant.* **2020**, *3*, 12–18. [CrossRef]
53. Fisher, R.A. *Iris*; UCI Machine Learning Repository: Los Angeles, CA, USA, 1988. [CrossRef]
54. Horst, A.M.; Hill, A.P.; Gorman, K.B. *Palmerpenguins: Palmer Archipelago (Antarctica) Penguin Data*; R Package Version 0.1.0; Zenodo: Geneva, Switzerland, 2020. [CrossRef]