InClass nets: independent classifier networks for nonparametric estimation of conditional independence mixture models and unsupervised classification

IOP Publishing Mach. Learn.: Sci. Technol. 3 (2022) 025008 https://doi.org/10.1088/2632-2153/ac6483

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
13 December 2021

REVISED
1 April 2022

ACCEPTED FOR PUBLICATION
5 April 2022

PUBLISHED
9 May 2022

PAPER

InClass nets: independent classifier networks for nonparametric estimation of conditional independence mixture models and unsupervised classification

Konstantin T Matchev[1,†] **and Prasanth Shyamsundar**[2,*,†]

[1] Institute for Fundamental Theory, Physics Department, University of Florida, Gainesville, FL 32611, United States of America
[2] Fermi National Accelerator Laboratory, Fermilab Quantum Institute, PO Box 500, Batavia, IL 60510, United States of America
[*] Author to whom any correspondence should be addressed.
[†] Authors are listed alphabetically by last name.

E-mail: prasanth@fnal.gov

Keywords: mixture models, conditional independence, mutual information, nonparametric estimation, unsupervised learning, semi-supervised learning, neural networks

Abstract

Conditional independence mixture models (CIMMs) are an important class of statistical models used in many fields of science. We introduce a novel unsupervised machine learning technique called the independent classifier networks (InClass nets) technique for the nonparameteric estimation of CIMMs. InClass nets consist of multiple independent classifier neural networks (NNs), which are trained simultaneously using suitable cost functions. Leveraging the ability of NNs to handle high-dimensional data, the conditionally independent variates of the model are allowed to be individually high-dimensional, which is the main advantage of the proposed technique over existing non-machine-learning-based approaches. Two new theorems on the nonparametric identifiability of bivariate CIMMs are derived in the form of a necessary and a (different) sufficient condition for a bivariate CIMM to be identifiable. We use the InClass nets technique to perform CIMM estimation successfully for several examples. We provide a public implementation as a Python package called RainDancesVI.

Contents

1. Introduction

In this section we shall introduce the general problem of *nonparametric* estimation of conditional independence mixture models (CIMMs), discuss related work, and briefly describe our machine learning based estimation technique. In section 1.1 we define CIMMs and discuss their different estimation paradigms, namely parametric, semi-parametric and non-parametric. In sections 1.2 and 1.3 we review related ideas in the literature, and in section 1.4 we provide a high-level overview of our technique, before filling in the technical details in the subsequent sections.

1.1. Conditional independence mixture models

In many fields of science one encounters multivariate models which consist of several distinct sub-populations or components, say C in number. Each component $i \in \{1,\ldots,C\}$ has its own characteristic probability density function $f^{(i)}(\mathcal{X})$ of the relevant multi-dimensional variable $\mathcal{X}$. Such models are referred to as multivariate finite mixture models [1], and the probability density of $\mathcal{X}$ under such a model is given by

$$\mathcal{P}(\mathcal{X}) = \sum_{i=1}^{C} w_i f^{(i)}(\mathcal{X}), \quad \text{with} \quad \sum_{i=1}^{C} w_i = 1, \quad \text{and} \quad w_i \geqslant 0 \quad \forall i \in \{1,\ldots,C\}, \tag{1}$$

where the non-negative weights w_i parameterize the mixing proportions of the individual components. An important special case of these finite mixture models is that of the conditional independence multivariate finite mixture models[3] [3, 4], which for brevity we will simply refer to as CIMMs. Under this special case, the variable $\mathcal{X}$ is parameterized using V 'variates' as $\mathcal{X} \equiv (x_1,\ldots,x_V)$ such that for each component i, the density function $f^{(i)}$ factorizes into a product of distributions for the individual variates as

$$f^{(i)}(\mathcal{X}) = \prod_{v=1}^{V} f_v^{(i)}(x_v), \qquad \forall i \in \{1,\ldots,C\}, \tag{2}$$

so that (1) becomes

$$\mathcal{P}(\mathcal{X}) = \sum_{i=1}^{C} w_i \prod_{v=1}^{V} f_v^{(i)}(x_v). \tag{3}$$

Here $f_v^{(i)}(x_v)$ is the unit-normalized probability density of x_v within component i—the top index (i) denotes the component and the bottom index v denotes the variate. In our treatment, the individual variates x_v are themselves allowed to be multi-dimensional. In other words, V is not the dimensionality of the data, but rather the number of groups the attributes in $\mathcal{X}$ can be partitioned into so that they are (conditionally) independent of each other within the given component i a datapoint belongs to. This is similar to the treatment, for example, in [3, 5, 6]. In particular, the technique we develop below will be applicable in situations where the variates x_v are high-dimensional ($\dim(x_v) \gg 1$). Unless otherwise stated, henceforth a 'mixture model' shall refer to the CIMM of (3).

1.1.1. Applications of conditional independence mixture models

CIMMs have applications in situations where the correlations and dependence between different variables in the data are explained in terms of a latent or hidden *confounding* variable which influences the observed variables. This is referred to as Latent Structure Analysis (LSA) [7, 8]. In particular, when the confounding variable is discrete or categorical, it can be interpreted as representing the *class* a given datapoint belongs to. Such models are referred to as latent class models and their study and analysis is referred to as latent class analysis (LCA) [9].

The connection between CIMMs and LCMs can be seen in a straightforward manner as follows. We can sample a datapoint as per the mixture model in (3) by first generating the component index $i \in \{1,\ldots,C\}$ as per the multinomial probability distribution induced by the weights w_i, and then sampling $(x_1,\ldots,x_V)$ as per the distribution $f^{(i)}(\mathcal{X})$ within component i. Now, the component index i can be interpreted as the latent variable that explains the dependence between the random variates $\{x_1,\ldots,x_V\}$ in the mixture.

CIMMs, LCA, and mixture models in general have applications in a wide range of fields, including econometrics [10, 11], social sciences [12–15], bioinformatics [16], astronomy and astrophysics [17–23], high energy physics [24–26], and many others.

[3] In the literature, these models are also referred to as finite mixtures of product measures [2].

1.1.2. Nonparametric estimation of conditional independence mixture models

Estimation of a CIMM is simply the process of estimating the weights w_i and functions $f_v^{(i)}$ (assuming the number of components C is known) from a dataset sampled from the joint distribution $\mathcal{P}(\mathcal{X})$ of the variates under the mixture model, see (3).

Under 'parametric' estimation of mixture models (conditionally independent or otherwise), one assumes that each of the distributions $f^{(i)}(\mathcal{X})$ is from an appropriately chosen parametric class of distributions, e.g. multivariate Gaussians. The choice of the class of functions assumed to contain the true $f^{(i)}(\mathcal{X})$ is informed by practitioner's prior knowledge of the problem at hand. This reduces the problem of estimating the mixture to the more tractable problem of estimating the weights w_i and the parameter values corresponding to the true $f^{(i)}(\mathcal{X})$. This weight and parameter estimation from the data is typically approached as a maximum likelihood estimation problem [27] (often tackled using the expectation–maximization algorithm [28]) or within a Bayesian approach [29].

'Semi-parametric' estimation of mixture models has been studied in many works, including [3, 30–32]. In this paper, however, we are interested in the 'nonparametric' estimation of CIMMs, i.e. no parametric forms will be assumed for the functions $f_v^{(i)}(x_v)$. Nonparametric estimation has been addressed by several works recently [4–6, 31, 33–38]. In this paper, we introduce a novel machine-learning-based-approach, called the InClass nets technique, to split the dataset into its different components in a nonparametric way. This splitting naturally leads to the estimation of the mixture model. In order to perform the splitting, the InClass nets technique directly exploits the fact that the variates x_v are mutually independent of each other within each component. Earlier known approaches nonparametric CIMM estimation either (a) discretize the data-space into bins and estimate the representative value of the component functions $f_v^{(i)}$ in those bins [33], (b) estimate each component distribution using a smoothed kernel-based approach [4–6, 31, 34–36], or (c) express each component distribution in terms of a set of basis functions [38]. Due to the curse of dimensionality, these approaches are applicable only when the individual variates x_v are low-dimensional. Due to neural networks (NNs) ability to handle high dimensional data, our technique can tackle situations where the individual variates x_v are high-dimensional—this is the biggest advantage offered by our machine-learning-based-technique over existing approaches. This opens up the possibility of using CIMMs for hitherto unfeasible applications.

The estimation of mixture models is closely tied to the concept of 'identifiability' of mixture models. A statistical model is said to be identifiable if it is theoretically possible to estimate the model (i.e. uniquely identify the parameters and functions that describe it) based on an infinite dataset sampled from it. A model will not be identifiable if two or more parameterizations of the model are observationally indistinguishable even with an infinite dataset.

In the situations where the CIMM is identifiable, our technique estimates the true w_i and $f_v^{(i)}$. On the other hand, when the model is not identifiable, our technique will yield one of the parameterizations that best fits the available data. In section 4.1, we provide some new results on the (nonparametric) identifiability of bivariate ($V = 2$) CIMMs, to supplement existing results on nonparametric mixture model identifiability [2, 5, 33, 39–45].

1.2. Unsupervised classification in machine learning

In this paper, we approach the estimation of CIMMs as a classification problem—classifying the datapoints in a given dataset into the different categories will lead to the estimation of the mixture model in a straightforward way.

This classification needs to be performed in an unsupervised manner since the dataset being analyzed does not contain labels for the component each datapoint belongs to. In this way, our method has connections to unsupervised clustering techniques like k-means clustering [46] and other density-based clustering techniques [47]. However, our approach does not rely on the different components being spatially clustered to perform the classification.

The intuition behind our method can be understood as follows: In supervised classification, the target class labels associated with the training datapoints serve as the supervisory signal for training the classifier. In the absence of target labels, a quantity that is dependent on or shares mutual information with, the (unavailable) target label can be used as the supervisory signal. Now, lets say we are training a classifier that bases its decision or output only on the first variate x_1. The other variates $\{x_2, \ldots, x_V\}$ can serve as the supervisory signal, since they contain information regarding the component i the datapoint belongs to. Our approach can be thought of as training V classifiers, one for each of the V variates, with each classifier relying on the other $V - 1$ variates to act as the supervisory signal for training.

There are a few ways of interpreting and actualizing this intuition [48–50]. For example, in [50], NNs were trained without supervision to classify images, using a training dataset consisting of pairs of images,

where the images in a given pair are from the same category. In the InClass nets approach developed in this paper, the NN architecture and training cost functions we develop will primarily be geared towards estimating CIMMs, which, as shown below in section 3.3, can nevertheless be used for classification similar to the technique of [50]. In section 5.2, we also discuss a straightforward extension of the technique in [50] to handle n-tuples of data, where the components of the n-tuple could be from different sample spaces (as opposed to pairs of images from the same sample space of images).

The idea of using a quantity that shares information with the true labels as the supervisory signal has been employed previously in weakly supervised classification techniques like 'Learning from Label Proportions' (LLPs) [51–53] and 'Classification Without Labels' (CWoLa) [54]. LLP and CWoLa learn to distinguish between different classes of datapoints, using multiple mixed datasets which differ in the mixing proportions of the classes—the identity of the mixed dataset a given datapoint belongs to serves as the supervisory signal, since it contains information about the class the datapoint belongs to (due to the mixing proportions in different mixtures being different). While CWoLa and LLP are not fully unsupervised techniques (since they still require a label indicating which mixture a training datapoint belongs to), they are applicable even in situations where the distribution of the feature $\mathcal{X}$ within a given class i does not factorize as $\prod_{v=1}^{V} f_v^{(i)}$.

1.3. Other related work

The idea of separating a mixture into its components using a mutual-information-based technique is similar in spirit to Independent Component Analysis (ICA) [55]. However, ICA solves a signal separation problem where multiple mixtures with different mixing weights for the components are provided—this is different from the problem of separating data from a single CIMM into its components.

Bayesian nonparametric methods have been applied in the context of mixture models to select the number of components C in the mixture model using the data itself [56]. In this technique, the individual components of the mixture themselves are parameterized. In contrast, our technique assumes that the number of components C is *a priori* known (we briefly discuss how C could be estimated in section 4.3), but the individual component distributions are left nonparameterized.

Mixture models have applications in data analysis in high energy physics [24–26], where datasets are mixtures of 'events' (datapoints) produced under different 'processes' (categories). The ${}_s\mathcal{P}lot$ technique [57], which is popular in data analysis in high energy physics, is used to analyze bivariate CIMMs where the distribution of one of the variables (referred to as the discriminating variable) is known *a priori*. In such situations, the ${}_s\mathcal{P}lot$ technique can estimate the distribution of the other variable, referred to as the control variable. On the other hand, the InClass nets approach introduced in this paper is capable of estimating the mixture model 'without any knowledge of the distributions of any of the variables'. In section 4.5, we describe how the InClass nets approach can be modified to incorporate additional information about the distributions of some of the variates.

1.4. Synopsis

1.4.1. Review of parametric model estimation using maximum likelihood estimation

Our technique for nonparametric mixture model estimation is closely related to the estimation of parametric models using maximum likelihood estimation (MLE). Under MLE, we have a parametric class of probability distributions $\{\mathcal{P}_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta\}$, parameterized by $\boldsymbol{\theta}$ (possibly multi-dimensional). We are provided a dataset $\{\mathcal{X}_1, \ldots, \mathcal{X}_N\}$ of size N sampled from an unknown data-distribution $\mathcal{P}_{\boldsymbol{\theta}^*}$, which is known to belong to the parametric class. The goal is to determine the value of $\boldsymbol{\theta}^*$ (assuming the model is identifiable). The MLE estimator for $\boldsymbol{\theta}^*$ is given by

$$\hat{\boldsymbol{\theta}}_{\text{MLE}} = \underset{\boldsymbol{\theta}\in\Theta}{\arg\max} \left[\frac{1}{N} \sum_{a=1}^{N} \ln \mathcal{P}_{\boldsymbol{\theta}}(\mathcal{X}_a) \right] = \underset{\boldsymbol{\theta}\in\Theta}{\arg\min} \left[-\frac{1}{N} \sum_{a=1}^{N} \ln \mathcal{P}_{\boldsymbol{\theta}}(\mathcal{X}_a) \right]. \tag{4}$$

The asymptotic consistency of this estimator follows from noting that

$$\operatorname*{plim}_{N\to\infty} \hat{\boldsymbol{\theta}}_{\text{MLE}} = \underset{\boldsymbol{\theta}\in\Theta}{\arg\min} \left[E_{\mathcal{P}_{\boldsymbol{\theta}^*}} \left[-\ln \mathcal{P}_{\boldsymbol{\theta}}(\mathcal{X}) \right] \right] \tag{5a}$$

$$= \underset{\boldsymbol{\theta}\in\Theta}{\arg\min} \left[E_{\mathcal{P}_{\boldsymbol{\theta}^*}} \left[\ln \mathcal{P}_{\boldsymbol{\theta}^*}(\mathcal{X}) - \ln \mathcal{P}_{\boldsymbol{\theta}}(\mathcal{X}) \right] \right] \tag{5b}$$

$$= \underset{\boldsymbol{\theta}\in\Theta}{\arg\min} \left[\text{KL} \left[\mathcal{P}_{\boldsymbol{\theta}^*} \,||\, \mathcal{P}_{\boldsymbol{\theta}} \right] \right], \tag{5c}$$

where plim denotes convergence in probability, and $\mathrm{KL}\left[\mathcal{P}_{\boldsymbol{\theta}^*} \,||\, \mathcal{P}_{\boldsymbol{\theta}}\right]$ is the Kullback–Leibler divergence from $\mathcal{P}_{\boldsymbol{\theta}}$ to $\mathcal{P}_{\boldsymbol{\theta}^*}$, which is minimized when $\mathcal{P}_{\boldsymbol{\theta}}$ equals $\mathcal{P}_{\boldsymbol{\theta}^*}$ almost everywhere.

1.4.2. Blueprint for nonparametric estimation of CIMMs
The preceding review of MLE suggests the following approach to estimating CIMMs nonparametrically using a dataset sampled from the data-distribution $\mathcal{P}^*$.

(a) Search through the space of CIMMs $\mathcal{P}$ of the form given in (3), and
(b) Minimize an appropriate objective function which depends on $\mathcal{P}$ and the available data. Asymptotically, minimizing the objective function should be equivalent to minimizing the Kullback–Leibler divergence from $\mathcal{P}$ to the data distribution $\mathcal{P}^*$.

We will use NNs to parameterize CIMMs. More specifically, we will use V different NN-based classifiers (one for each variate), along with the marginal distributions of the individual variates in the data, to parameterize CIMMs. We refer to this as the Independent Pseudo Classifiers (IPCs) representation of CIMMs (section 2.1.1). We will show that every CIMM has a (non-unique) IPC representation. Thus the task of searching through the space of CIMMs has been converted into the task of searching through the space of classifiers, i.e, training the NN-based classifiers using an appropriate cost function.

Next, in section 2.1.4, we will derive an expression for the KL divergence from the distribution $\mathcal{P}$ (in the IPC representation) to the data distribution $\mathcal{P}^*$ (up to a constant term independent of $\mathcal{P}$). This is the quantity to be minimized in order to estimate $\mathcal{P}^*$. After deriving the expression for $\mathrm{KL}\left[\mathcal{P}^* \,||\, \mathcal{P}\right]$, one can find a data-sample-based estimator for the same by replacing expectations over $\mathcal{P}^*$ with sample means. This leads to a cost function which only depends on the NN-classifier 'outputs' for the various 'input' datapoints in the sample. Minimizing this cost function is asymptotically equivalent to minimizing $\mathrm{KL}\left[\mathcal{P}^* \,||\, \mathcal{P}\right]$. In section 2.1, we will also describe how the weights w_i and component distributions $f_v^{(i)}$ of the estimated mixture model can be extracted from the classifier networks.

The rest of the paper is organized as follows. In section 2.2 we use the bivariate case as a simple case study to summarize the results from section 2.1, in the order in which they will be used in a typical analysis. We provide a public implementation of InClass nets as a Python package called `RainDancesVI` and use it to validate our InClass nets technique with several worked out examples in section 3. In section 4.1 we then derive some new results on the nonparametric identifiability of bivariate CIMMs, in the form of a necessary and a (different) sufficient condition for a bivariate CIMM to be identifiable. In sections 4 and 5 we discuss our technique in the context of science applications, and provide possible variations and extensions, before finally summarizing in section 6.

2. Methodology

2.1. Independent classifier networks (InClass nets)

For the purpose of nonparametric estimation of CIMMs, we introduce a new NN architecture which we shall call 'Independent Classifier networks' or 'InClass nets' for short. Under InClass nets, the V variates $\{x_1,\ldots,x_V\}$ of the input $\mathcal{X}$ are fed into V independent NNs—one variate for each independent network. Each of the V networks returns a multi-class classifier output. More explicitly, for each $v\in\{1,\ldots,V\}$, the vth classifier network returns a vector $\left(\eta_v^{(1)}(x_v),\ldots,\eta_v^{(C)}(x_v)\right)$, whose ith component can 'roughly' be interpreted as the probability that a datapoint belongs to category i, conditional only on its x_v value.

The outputs of the independent classifiers are constrained to obey

$$\eta_v^{(i)}(x_v)\geqslant 0\,,\qquad \forall(i,v)\in\{1,\ldots,C\}\times\{1,\ldots,V\}\,, \tag{6a}$$

$$\sum_{i=1}^{C}\eta_v^{(i)}(x_v)=1\,,\qquad \forall v\in\{1,\ldots,V\}\,, \tag{6b}$$

possibly using the softmax output layer[4] [58] as follows:

$$\eta_v^{(i)}(x_v)=\mathtt{softmax}^{(i)}\left(z_v^{(1)},\ldots,z_v^{(C)}\right)\,,\qquad \forall(i,v)\in\{1,\ldots,C\}\times\{1,\ldots,V\}\,, \tag{7}$$

[4] For the case of $C=2$, one can also simply use a one-dimensional output layer constrained to be in $[0,1]$, with $(\mathtt{output},1-\mathtt{output})$ serving as $(\eta^{(1)},\eta^{(2)})$.

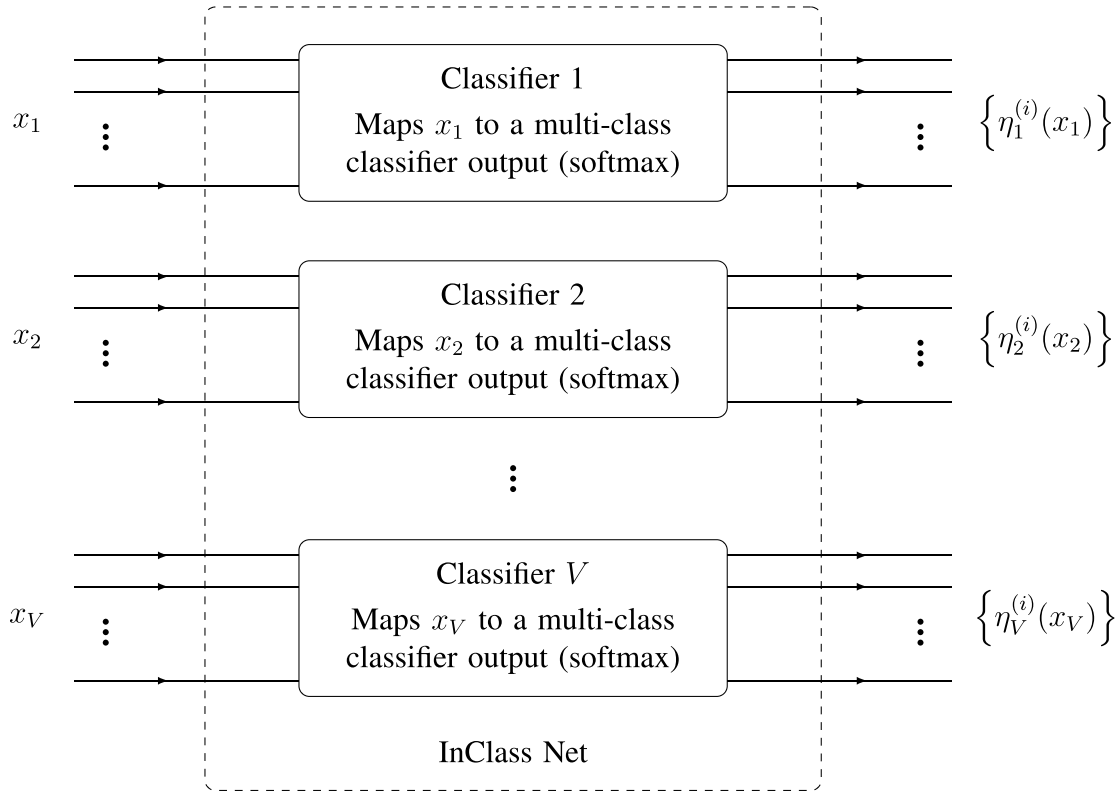


Figure 1. Basic architecture of independent classifier networks (InClass nets). The V variates $\{x_1,\ldots,x_V\}$ of the input $\mathcal{X}$ are fed into V independent NNs, each of which returns a multi-class classifier output $\eta_v^{(i)}(x_v)$ for $v\in\{1,\ldots,V\}$.

where the $z^{(i)}$-s are the inputs to the final output layer (which performs the softmax operation) of the corresponding network. Alternatively, if softmax is used as an *activation function* of the final layer, then the $z^{(i)}$-s represent the outputs of the layer before applying the activation function. The `softmax` function is defined as

$$\texttt{softmax}^{(i)}\left(z_v^{(1)},\ldots,z_v^{(C)}\right)\equiv\frac{\exp\left(z_v^{(i)}\right)}{\sum\limits_{j=1}^{C}\exp\left(z_v^{(j)}\right)}\,. \tag{8}$$

Figure 1 illustrates the basic architecture of InClass nets. In the next few sections, we will build the framework for estimating CIMMs using InClass nets.

Recall that the variates x_v can be multi-dimensional. In particular, InClass nets can handle high-dimensional data types like images. The choice of architecture for the individual classifier networks can be influenced by the nature of the input data the classifier will handle.

For the purposes of this paper, we have restricted the output dimensionality of the individual classifiers to be the same (equal to C). We have also restricted the inputs $\{x_1,\ldots,x_V\}$ of the individual classifiers to form a non-overlapping partition of the features or attributes in $\mathcal{X}$, which is in line with the structure of CIMMs. However, InClass nets can have applications outside mixture model estimation as well, and for those purposes it may be appropriate to lift the above restrictions. For example, InClass nets can be used to perform unsupervised 'multi-label' classification, where the outputs of different classifiers correspond to different labels. In this case, the different classifiers can have different output dimensionalities and the inputs to these networks can also potentially have overlapping features. In section 5.2 we will briefly indicate how the multi-label variant of InClass nets can be trained to perform unsupervised classification by maximizing the mutual information between the classifier outputs.

2.1.1. Parameterizing mixture models with InClass nets

In this section we will show how CIMMs can be parametrized using InClass nets. The parametrization will be done using the IPCs representation of mixture models which will be introduced in section 2.1.1. But as a useful lead-up, let us first introduce the Constrained Independent Classifiers (CICs) representation.

2.1.1.1. Constrained independent classifiers representation

The mixture model in (3) is completely specified by the mixture weights w_i and the distributions $f_v^{(i)}$. Recall that they satisfy

$$w_i \geqslant 0\,,\;\; f_v^{(i)}(x_v) \geqslant 0\,, \qquad \forall (i,v) \in \{1,\ldots,C\} \times \{1,\ldots,V\}\,, \tag{9a}$$

$$\sum_{i=1}^{C} w_i = 1\,, \tag{9b}$$

$$\int dx_v\, f_v^{(i)}(x_v) = 1\,, \qquad \forall (i,v) \in \{1,\ldots,C\} \times \{1,\ldots,V\}\,. \tag{9c}$$

The goal of this paper is to develop a machine learning technique to fit a mixture model to the given data in an agnostic, nonparametric, manner. In other words, we will estimate the weights w_i and the distributions $f_v^{(i)}$, without assuming, *a priori*, any parameterized forms (like Gaussians, exponentials, etc) for $f_v^{(i)}$. We will approach this as an unsupervised multi-class classification problem—classifying the data into different components will automatically result in an estimation of the mixture model[5]. To this end, let us rewrite the mixture model distribution in terms of the marginal distributions $\mathcal{P}_v(x_v)$ of the individual variates and multi-class classifiers $\alpha_v^{(i)}(x_v)$ given by

$$\mathcal{P}_v(x_v) = \int dx_1 \ldots \int dx_{v-1} \int dx_{v+1} \ldots \int dx_V\, \mathcal{P}(\mathcal{X}) \tag{10a}$$

$$= \sum_{i=1}^{C} w_i f_v^{(i)}(x_v)\,, \qquad \forall v \in \{1,\ldots,V\}\,, \tag{10b}$$

$$\alpha_v^{(i)}(x_v) = \frac{w_i\, f_v^{(i)}(x_v)}{\mathcal{P}_v(x_v)}\,, \qquad \forall (i,v) \in \{1,\ldots,C\} \times \{1,\ldots,V\}\,. \tag{10c}$$

$\mathcal{P}_v$ is the probability density of the vth variate in the full mixture and can be directly accessed from a dataset sampled from $\mathcal{P}$. $\alpha_v^{(i)}(x_v)$ can be interpreted as the probability that an observed datapoint is from component i conditional on the value of x_v. The vector function $\left(\alpha_v^{(1)}(x_v),\ldots,\alpha_v^{(C)}(x_v)\right)$ can be interpreted as the output of a multi-class 'classifier' that returns the probability of a datapoint $\mathcal{X}$ to have come from the different components based only on the vth variate. At this point, one might already notice an emerging connection with InClass nets, which we shall crucially exploit below. The marginals density functions $\mathcal{P}_v$ and the multi-class classifiers $\alpha_v^{(i)}$ satisfy

$$\mathcal{P}_v(x_v) \geqslant 0\,,\;\; \alpha_v^{(i)}(x_v) \geqslant 0\,, \quad \forall (i,v) \in \{1,\ldots,C\} \times \{1,\ldots,V\}\,, \tag{11a}$$

$$\int dx_v\, \mathcal{P}_v(x_v) = 1\,, \quad \forall v \in \{1,\ldots,V\}\,, \tag{11b}$$

$$\sum_{i=1}^{C} \alpha_v^{(i)}(x_v) = 1\,, \quad \forall v \in \{1,\ldots,V\}\,, \tag{11c}$$

$$\int dx_v\, \mathcal{P}_v(x_v)\, \alpha_v^{(i)}(x_v) = \int dx_u\, \mathcal{P}_u(x_u)\, \alpha_u^{(i)}(x_u)\,, \;\; \forall (i,v,u) \in \{1,\ldots,C\} \times \{1,\ldots,V\}^2\,, \tag{11d}$$

where the integrals in (11*d*) are simply equal to the weight w_i of the ith component. There is a one-to-one map[6] from the description of the mixture model in terms of the w_i-s and $f_v^{(i)}$-s satisfying (9) to the

[5] Directly modeling the distributions $f_v^{(i)}$ is possible using generative networks, but classifier outputs are more robust quantities, e.g. they are invariant under invertible transformations of the x_v-s, and are typically easier to learn in machine learning.

[6] This is not a statement on the identifiability of conditional independence mixture models. Identifiability of mixture models will be briefly discussed in section 4.1.

description in terms of $\mathcal{P}_v$-s and $\alpha_v^{(i)}$-s satisfying (11). This can be seen from the existence of the inverse transform shown below:

$$w_i = \int dx_v\, \mathcal{P}_v(x_v)\, \alpha_v^{(i)}(x_v) \equiv E_{\mathcal{P}}\left[\alpha_v^{(i)}\right], \qquad \forall (i,v) \in \{1,\ldots,C\} \times \{1,\ldots,V\}, \tag{12a}$$

$$f_v^{(i)}(x_v) = \frac{\mathcal{P}_v(x_v)\, \alpha_v^{(i)}(x_v)}{w_i}, \qquad \forall (i,v) \in \{1,\ldots,C\} \times \{1,\ldots,V\}, \tag{12b}$$

where $E_{\mathcal{P}}[\cdots]$ represents the expectation value of $\cdots$ under the model. The probability density of $\mathcal{X}$ under the corresponding mixture model is given by

$$\mathcal{P}(\mathcal{X}) = \sum_{i=1}^{C} w_i \prod_{v=1}^{V} \frac{\mathcal{P}_v(x_v)\, \alpha_v^{(i)}(x_v)}{w_i} \tag{13a}$$

$$= \left[\prod_{v=1}^{V} \mathcal{P}_v(x_v)\right] \left[\sum_{i=1}^{C} w_i^{1-V} \prod_{v=1}^{V} \alpha_v^{(i)}(x_v)\right]. \tag{13b}$$

As mentioned earlier, the marginal distributions $\mathcal{P}_v$ can be directly estimated from the data. The functions $\alpha_v^{(i)}$ can 'potentially' be modeled using InClass nets. The only hurdle is that while the outputs $\alpha_v^{(i)}$ of the V NNs can be constrained to obey (11*a*) and (11*c*) using the softmax output layer (as seen in (6)), the constraint in (11*d*) in general will **not** be satisfied by independent classifiers. We will handle this difficulty next in section 2.1.1. We will refer to the description in terms of $\mathcal{P}_v$-s and $\alpha_v^{(i)}$-s satisfying (11*a*)–(11*d*) as the CICs representation of the mixture model.

2.1.1.2. Independent pseudo classifiers representation

To accommodate the fact that independent classifiers will not obey the constraint (11*d*) of the CICs representation, we introduce the IPCs representation in terms of pseudo marginals $\mathcal{Q}_v$ and pseudo classifiers $\beta_v^{(i)}$ which only satisfy the equivalents of constraints (11*a*–11*c*):

$$\mathcal{Q}_v(x_v) \geqslant 0\,, \;\; \beta_v^{(i)}(x_v) \geqslant 0\,, \qquad \forall (i,v) \in \{1,\ldots,C\} \times \{1,\ldots,V\}\,, \tag{14a}$$

$$\int dx_v\, \mathcal{Q}_v(x_v) = 1\,, \qquad \forall v \in \{1,\ldots,V\}\,, \tag{14b}$$

$$\sum_{i=1}^{C} \beta_v^{(i)}(x_v) = 1\,, \qquad \forall v \in \{1,\ldots,V\}\,. \tag{14c}$$

The mixture weights under the IPC representation are given by

$$w_i = \frac{\tilde{w}_i}{\sum\limits_{j=1}^{C} \tilde{w}_j}, \qquad \forall i \in \{1,\ldots,C\}\,, \tag{15}$$

where the unnormalized weights $\tilde{w}_i$-s are given by

$$\tilde{w}_i = \left[\prod_{v=1}^{V} \int dx_v\, \mathcal{Q}_v(x_v)\, \beta_v^{(i)}(x_v)\right]^{1/V} = \left[\prod_{v=1}^{V} E_{\mathcal{Q}}\left[\beta_v^{(i)}\right]\right]^{1/V} \equiv \left[\prod_{v=1}^{V} \varphi_v^{(i)}\right]^{1/V}, \qquad \forall i \in \{1,\ldots,C\}\,, \tag{16}$$

where $\varphi_v^{(i)} \equiv E_{\mathcal{Q}}\left[\beta_v^{(i)}\right]$ represents the expectation value of $\beta_v^{(i)}$ under the distribution $\mathcal{Q}_v$. The distributions $f_v^{(i)}$ within the different components are given under the IPC representation by

$$f_v^{(i)}(x_v) = \frac{\mathcal{Q}_v(x_v)\, \beta_v^{(i)}(x_v)}{E_{\mathcal{Q}}\left[\beta_v^{(i)}\right]} = \frac{\mathcal{Q}_v(x_v)\, \beta_v^{(i)}(x_v)}{\varphi_v^{(i)}}. \tag{17}$$

In (15), we have used $\tilde{w}_i$, which is defined in (16) as the geometric mean[7] of $\varphi_v^{(i)}$-s, as the actual mixture weight w_i, after an appropriate scaling to make the weights add up to 1 across all components. We will refer to $\varphi_v^{(i)}$ as the pseudo weight of component i corresponding to variate v. Using (15) and (17), we can write the probability density function for the mixture model in the IPC representation as

$$\mathcal{P}(\mathcal{X}) = \sum_{i=1}^{C} w_i \prod_{v=1}^{V} \frac{\mathcal{Q}_v(x_v)\,\beta_v^{(i)}(x_v)}{\tilde{w}_i} \tag{18a}$$

$$= \left[\prod_{v=1}^{V} \mathcal{Q}_v(x_v)\right] \frac{\sum_{i=1}^{C} \tilde{w}_i^{1-V} \prod_{v=1}^{V} \beta_v^{(i)}(x_v)}{\sum_{i=1}^{C} \tilde{w}_i}, \tag{18b}$$

where the $\tilde{w}_i$-s can be written in terms of $\mathcal{Q}_v$-s and $\beta_v^{(i)}$-s using (16). We will now make the following observations relevant to our goal of fitting a mixture model to data using InClass nets:

(a) IPC describes a CIMM. Note that by construction, the mixture weights in (15) and the distributions in (17) are non-negative and normalized to 1.

(b) The pseudo marginals and pseudo classifiers do not necessarily correspond to the true marginals and classifiers. However, the true marginals and classifiers of the CIC representation can be extracted from the IPC representation as follows

$$\mathcal{P}_v(x_v) = \frac{\mathcal{Q}_v(x_v)}{\sum_{i=1}^{C} \tilde{w}_i} \sum_{i=1}^{C} \frac{\beta_v^{(i)}(x_v)\,\tilde{w}_i}{\varphi_v^{(i)}}, \tag{19a}$$

$$\alpha_v^{(i)}(x_v) = \left[\sum_{j=1}^{C} \frac{\beta_v^{(j)}(x_v)\,\tilde{w}_j}{\varphi_v^{(j)}}\right]^{-1} \frac{\beta_v^{(i)}(x_v)\,\tilde{w}_i}{\varphi_v^{(i)}}. \tag{19b}$$

These results follow from plugging in (15)–(17) in (10).

(c) The IPC representation of a mixture model is not unique. Unlike the CIC representation, we cannot find a unique map from the weights w_i and $f_v^{(i)}$ to the pseudo marginals and pseudo classifiers. This is because of the additional degrees of freedom due to the removal of the constraints in (11*d*).

(d) Every mixture model has an IPC representation[8] in which the pseudo marginals match the true marginals of the model. This can be seen from the fact that the true marginals $\mathcal{P}_v$ and classifiers $\alpha_v^{(i)}$ from the CIC representation of a mixture model can used as the pseudo marginals $\mathcal{Q}_v$ and pseudo classifiers $\beta_v^{(i)}$ under the IPC representation to get the same model.

Observation (d) means that in order to fit a mixture model to data, we can restrict ourselves to IPC representations of the mixture models with the pseudo marginals set to the marginals of the data. The only remaining unknowns in the IPC representation are the pseudo classifiers $\beta_v^{(i)}(x_v)$ which we can parameterize using an InClass net, identifying $\beta_v^{(i)}$ with the network output $\eta_v^{(i)}$. Next, we will develop the technique to fit a mixture model parameterized with an InClass net to a given dataset.

2.1.2. Fitting mixture models to data with InClass nets

In this section we will construct a cost function which can be used to train InClass nets to fit mixture models to the given data. Let the data to which we want to fit a mixture model be sampled from the true underlying distribution $\mathcal{P}^*(\mathcal{X})$ with true marginals $\mathcal{P}_v^*(x_v)$. As per observation (d) in the previous section, we restrict

[7] It is also possible to use other mean functions, including generalized means, instead of the geometric mean here. However we use the geometric mean, since it subsequently leads to simple expressions, since $\prod_{v=1}^{V} \varphi_v^{(i)}$ can be replaced with $\tilde{w}_i^V$, for example, as in (18*a*). This simplification also leads to computational advantages during NN training.

[8] Not necessarily unique, even after imposing the constraint that the pseudo marginals match the true marginals.

our attention to IPC representations with $\mathcal{Q}_v \equiv \mathcal{P}^*_v$. Using (16) and (18*b*), we can write the probability density of $\mathcal{X}$ under this restricted class of mixture models as

$$\mathcal{P}(\mathcal{X}) = \left[\prod_{v=1}^{V} \mathcal{P}^*_v(x_v)\right] \frac{\displaystyle\sum_{i=1}^{C} \left[\prod_{v=1}^{V} \beta_v^{(i)}(x_v) \left(E_{\mathcal{P}^*}\left[\beta_v^{(i)}\right]\right)^{(1-V)/V}\right]}{\displaystyle\sum_{i=1}^{C} \left[\prod_{v=1}^{V} \left(E_{\mathcal{P}^*}\left[\beta_v^{(i)}\right]\right)^{1/V}\right]}, \tag{20}$$

where $E_{\mathcal{P}^*}$ refers to the expectation value under the true distribution of the data. The best-fitting $\mathcal{P}$ can be estimated by minimizing the Kullback–Leibler (KL) divergence from $\mathcal{P}$ to $\mathcal{P}^*$ given by

$$\mathrm{KL}\left[\mathcal{P}^* \,||\, \mathcal{P}\right] = \int d\mathcal{X}\, \mathcal{P}^*(\mathcal{X}) \log\left[\frac{\mathcal{P}^*(\mathcal{X})}{\mathcal{P}(\mathcal{X})}\right]. \tag{21}$$

As discussed in section 1.4, minimizing the KL divergence (over some class of distributions) is equivalent to, and commonly known in some disciplines as, maximizing the likelihood in the large statistics limit. Using the expression for $\mathcal{P}$ from (20), we can rewrite (21) as

$$\mathrm{KL}\left[\mathcal{P}^* \,||\, \mathcal{P}\right] = \int d\mathcal{X}\, \mathcal{P}^*(\mathcal{X}) \log\left[\frac{\mathcal{P}^*(\mathcal{X})}{\left[\prod_{v=1}^{V} \mathcal{P}^*_v(x_v)\right]} \frac{\left[\prod_{v=1}^{V} \mathcal{P}^*_v(x_v)\right]}{\mathcal{P}(\mathcal{X})}\right] \tag{22a}$$

$$= \int d\mathcal{X}\, \mathcal{P}^*(\mathcal{X}) \log\left[\frac{\mathcal{P}^*(\mathcal{X})}{\left[\prod_{v=1}^{V} \mathcal{P}^*_v(x_v)\right]}\right] - \int d\mathcal{X}\, \mathcal{P}^*(\mathcal{X}) \log\left[\frac{\mathcal{P}(\mathcal{X})}{\left[\prod_{v=1}^{V} \mathcal{P}^*_v(x_v)\right]}\right] \tag{22b}$$

$$= C^*(x_1, \ldots, x_V) - E_{\mathcal{P}^*}\left[\log\left\{\frac{\displaystyle\sum_{i=1}^{C} \left[\prod_{v=1}^{V} \beta_v^{(i)} \left(E_{\mathcal{P}^*}\left[\beta_v^{(i)}\right]\right)^{(1-V)/V}\right]}{\displaystyle\sum_{i=1}^{C} \left[\prod_{v=1}^{V} \left(E_{\mathcal{P}^*}\left[\beta_v^{(i)}\right]\right)^{1/V}\right]}\right\}\right], \tag{22c}$$

where $C^*(x_1, \ldots, x_V)$ is the total correlation [59, 60] of the V variates in the data, which is one of the generalizations of mutual information to more than two variables. It is given by the KL divergence from the product distribution $\prod_v \mathcal{P}^*_v(x_v)$ to the joint distribution $\mathcal{P}^*(\mathcal{X})$ as

$$C^*(x_1, \ldots, x_V) = \int d\mathcal{X}\, \mathcal{P}^*(\mathcal{X}) \log\left[\frac{\mathcal{P}^*(\mathcal{X})}{\mathcal{P}^*_1(x_1)\, \mathcal{P}^*_2(x_2)\, \ldots\, \mathcal{P}^*_V(x_V)}\right]. \tag{23}$$

Note that the C^* term in (22*c*) is independent of the state of the InClass net under consideration. This means that the second term in (22*c*) can be used as a cost function for the network to minimize in order to minimize the KL divergence, and hence fit the mixture model parameterized by the InClass net to the data. Noting the similarity between the two terms in (22*b*) and drawing inspiration from the naming of 'cross entropy', we introduce the 'negative cross total correlation' cost function (`neg_ctc_cost`) defined as

$$\texttt{neg\_ctc\_cost} = \mathrm{KL}\left[\mathcal{P}^* \,||\, \mathcal{P}\right] - C^*(x_1, \ldots, x_V) \tag{24a}$$

$$= -E_{\mathcal{P}^*}\left[\log\left\{\frac{\displaystyle\sum_{i=1}^{C} \left[\prod_{v=1}^{V} \beta_v^{(i)} \left(E_{\mathcal{P}^*}\left[\beta_v^{(i)}\right]\right)^{(1-V)/V}\right]}{\displaystyle\sum_{i=1}^{C} \left[\prod_{v=1}^{V} \left(E_{\mathcal{P}^*}\left[\beta_v^{(i)}\right]\right)^{1/V}\right]}\right\}\right]. \tag{24b}$$

Note that $\beta_v^{(i)}$ are functions of the corresponding input variate x_v. Despite the complicated appearance, this cost function provides a viable approach to learning the underlying mixture model from data. Let us make the following observations in the context of training InClass nets using this cost function, with outputs $\eta_v^{(i)}$ of the network identified with $\beta_v^{(i)}$.

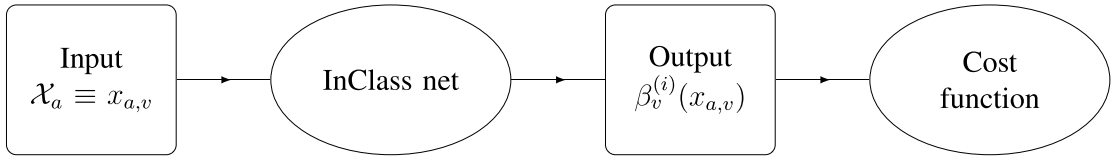


Figure 2. A flowchart illustrating the training of InClass nets. The indices $a \in \{1,\ldots,N_{\text{bat}}\}$, $v \in \{1,\ldots,V\}$, and $i \in \{1,\ldots,C\}$ correspond to samples, variates, and components, respectively. The diagram shows how the cost function in (25) is computed for a batch of N_{bat} datapoints, after they are sent through the InClass net. The output η of the InClass net is identified with the pseudo classifier β in the cost function.

(a) The cost function depends only on the outputs $\beta_v^{(i)}$ of the network. More precisely, the cost function depends on the distribution of the network output. It does not need the input data to be labelled to learn the mixture model, and the only supervisory signal exploited by the training process is the joint-distribution of the input data.

(b) The cost function for a given state of the InClass net can be estimated using a (mini-)batch of training samples by approximating the expectation values $E_{\mathcal{P}^*}[\cdots]$ with sample means, as shown below:

`neg_ctc_cost_from_data`

$$= -\frac{1}{N_{\text{bat}}}\sum_{a=1}^{N_{\text{bat}}}\left[\log\left\{\frac{\sum_{i=1}^{C}\left[\prod_{v=1}^{V}\beta_v^{(i)}(x_{a,v})\left(\frac{1}{N_{\text{bat}}}\sum_{b=1}^{N_{\text{bat}}}\left[\beta_v^{(i)}(x_{b,v})\right]\right)^{(1-V)/V}\right]}{\sum_{i=1}^{C}\left[\prod_{v=1}^{V}\left(\frac{1}{N_{\text{bat}}}\sum_{b=1}^{N_{\text{bat}}}\left[\beta_v^{(i)}(x_{b,v})\right]\right)^{1/V}\right]}\right\}\right], \tag{25}$$

where a and b are sample indices, and $x_{a,v}$ is the vth variate of the ath datapoint. The batch size N_{bat} should be large enough to perform a good estimation of $E_{\mathcal{P}^*}\left[\beta_v^{(i)}\right]$.

These observations allow us to train the NNs. The computation of the cost function for a batch of events is illustrated in figure 2. After training the InClass net, (15)–(18) and (17) can be used to extract the fitted model, with the pseudo marginals $\mathcal{Q}_v$ set to the true marginals $\mathcal{P}_v^*$. The classifiers $\alpha_v^{(i)}(x_v)$ can be extracted from the pseudo classifiers $\beta_v^{(i)}(x_v)$ using (19*b*). If one is interested in classifying the individual datapoints based on the full information $\mathcal{X}$, an aggregate classifier can be constructed, based on (18*b*), as

$$\alpha_{\text{aggregate}}^{(i)}(\mathcal{X}) = \frac{\tilde{w}_i^{1-V}\prod_{v=1}^{V}\beta_v^{(i)}(x_v)}{\sum_{j=1}^{C}\tilde{w}_j^{1-V}\prod_{v=1}^{V}\beta_v^{(j)}(x_v)}. \tag{26}$$

Note that if there is a mismatch between the model learned by the InClass net and the true distribution the data is sampled from, then classifying the data using the aggregate classifier will not necessarily lead to components within which the x_v-s are independent.

2.2. Bivariate case

When analyzing real data with CIMMs, a common difficulty is the identification of a suitable partitioning of the attributes of $\mathcal{X}$ into variates x_v so that the distribution within each component would factorize to a good approximation. In this sense, a higher number of (conditionally independent) variates represents stronger assumptions about the underlying model. This makes the bivariate case ($V = 2$) extremely important. The bivariate case is also difficult from an identifiability point of view—data distributed according to a conditional independence bivariate mixture model, in general, will not uniquely identify the model, since several different mixture models can lead to the same overall probability density $\mathcal{P}(\mathcal{X})$. In section 4.1, we will present some new results on the identifiability of conditional independence bivariate mixture models. In particular, we will provide the conditions under which bivariate mixture models are identifiable.

Despite being the most difficult case in terms of identifiability, the bivariate case lets us gain some useful intuition, as demonstrated with several examples in section 3 below. But first, in preparation for section 3, let us summarize the results from the previous sections for the bivariate case, in the order in which a typical analysis might use them.

2.2.1. Notation

The expressions from the previous sections become easier to follow if we explicitly write out the two variates, thus avoiding the product notation. To this end, let us simplify the notation by giving names x and y to our two variates x_1 and x_2, resulting in

$$x \equiv x_1\,, \quad \mathcal{P}_x \equiv \mathcal{P}_1\,, \quad \mathcal{Q}_x \equiv \mathcal{Q}_1\,, \quad \mathcal{P}_x^* \equiv \mathcal{P}_1^*\,, \quad \alpha_x^{(i)} \equiv \alpha_1^{(i)}\,, \quad \beta_x^{(i)} \equiv \beta_1^{(i)}\,, \quad f_x^{(i)} \equiv f_1^{(i)}\,, \tag{27a}$$

$$y \equiv x_2\,, \quad \mathcal{P}_y \equiv \mathcal{P}_2\,, \quad \mathcal{Q}_y \equiv \mathcal{Q}_2\,, \quad \mathcal{P}_y^* \equiv \mathcal{P}_2^*\,, \quad \alpha_y^{(i)} \equiv \alpha_2^{(i)}\,, \quad \beta_y^{(i)} \equiv \beta_2^{(i)}\,, \quad f_y^{(i)} \equiv f_2^{(i)}\,. \tag{27b}$$

Under this notation, the CIMM of (3) becomes simply

$$\mathcal{P}(x,y) = \sum_{i=1}^{C} w_i\, f_x^{(i)}(x)\, f_y^{(i)}(y)\,. \tag{28}$$

2.2.2. Cost function

Noting that total correlation is a generalization of mutual information for more than two variables, we will refer to the negative cross total correlation cost function of (24*b*) in the bivariate special case as the 'negative cross mutual information' cost function (`neg_cmi_cost`). Under our new notation, it is given by

$$\texttt{neg\_cmi\_cost} = -E_{\mathcal{P}^*}\left[\log\left\{\frac{\displaystyle\sum_{i=1}^{C}\frac{\beta_x^{(i)}\,\beta_y^{(i)}}{\sqrt{E_{\mathcal{P}^*}\left[\beta_x^{(i)}\right]\,E_{\mathcal{P}^*}\left[\beta_y^{(i)}\right]}}}{\displaystyle\sum_{i=1}^{C}\sqrt{E_{\mathcal{P}^*}\left[\beta_x^{(i)}\right]\,E_{\mathcal{P}^*}\left[\beta_y^{(i)}\right]}}\right\}\right], \tag{29}$$

where, as before, $E_{\mathcal{P}^*}$ represents the expectation over the true distribution $\mathcal{P}^*(x,y)$ from which the data is sampled. As in (25), the expectations over $\mathcal{P}^*$ can be estimated using sample means to train the NNs using this cost function.

2.2.3. Extracting the learned mixture model from the trained network

After training the InClass net, the trained $\beta_x^{(i)}$ and $\beta_y^{(i)}$ cannot directly be interpreted as classifiers based on x and y since they may correspond to different mixture weights. In order to extract the learned mixture model (and the corresponding classifiers), we can first estimate the pseudo weights $\varphi_x^{(i)}$ and $\varphi_y^{(i)}$ from the data as

$$\varphi_x^{(i)} = E_{\mathcal{P}^*}\left[\beta_x^{(i)}\right], \qquad \varphi_y^{(i)} = E_{\mathcal{P}^*}\left[\beta_y^{(i)}\right]. \tag{30}$$

Now, using (16) and (19*b*), the marginals and classifiers for the model represented by the InClass net can be constructed as

$$\mathcal{P}_x(x) = \mathcal{P}_x^*(x)\,\frac{\displaystyle\sum_{i=1}^{C}\beta_x^{(i)}(x)\sqrt{\frac{\varphi_y^{(i)}}{\varphi_x^{(i)}}}}{\displaystyle\sum_{i=1}^{C}\sqrt{\varphi_x^{(i)}\,\varphi_y^{(i)}}}, \qquad \alpha_x^{(i)}(x) = \frac{\beta_x^{(i)}(x)\sqrt{\dfrac{\varphi_y^{(i)}}{\varphi_x^{(i)}}}}{\displaystyle\sum_{j=1}^{C}\beta_x^{(j)}(x)\sqrt{\frac{\varphi_y^{(j)}}{\varphi_x^{(j)}}}}, \tag{31a}$$

$$\mathcal{P}_y(y) = \mathcal{P}_y^*(y)\,\frac{\displaystyle\sum_{i=1}^{C}\beta_y^{(i)}(y)\sqrt{\frac{\varphi_x^{(i)}}{\varphi_y^{(i)}}}}{\displaystyle\sum_{i=1}^{C}\sqrt{\varphi_x^{(i)}\,\varphi_y^{(i)}}}, \qquad \alpha_y^{(i)}(y) = \frac{\beta_y^{(i)}(y)\sqrt{\dfrac{\varphi_x^{(i)}}{\varphi_y^{(i)}}}}{\displaystyle\sum_{j=1}^{C}\beta_y^{(j)}(y)\sqrt{\frac{\varphi_x^{(j)}}{\varphi_y^{(j)}}}}. \tag{31b}$$

From (15) and (16), the component weights of the learned model are given by

$$w_i = \frac{\sqrt{\varphi_x^{(i)}\,\varphi_y^{(i)}}}{\sum_{j=1}^{C}\sqrt{\varphi_x^{(j)}\,\varphi_y^{(j)}}} \tag{32}$$

and from (17), the distributions $f_x^{(i)}$ and $f_y^{(i)}$ within each component are given by

$$f_x^{(i)}(x) = \frac{\mathcal{P}_x^*(x)\,\beta_x^{(i)}(x)}{\varphi_x^{(i)}}, \qquad f_y^{(i)}(y) = \frac{\mathcal{P}_y^*(y)\,\beta_y^{(i)}(y)}{\varphi_y^{(i)}}. \tag{33}$$

The corresponding joint distribution is given by

$$\mathcal{P}(x,y) = \mathcal{P}_x^*(x)\,\mathcal{P}_y^*(y)\,\frac{\sum_{i=1}^{C}\frac{\beta_x^{(i)}(x)\,\beta_y^{(i)}(y)}{\sqrt{\varphi_x^{(i)}\,\varphi_y^{(i)}}}}{\sum_{i=1}^{C}\sqrt{\varphi_x^{(i)}\,\varphi_y^{(i)}}}. \tag{34}$$

Note that after estimating $\mathcal{P}_x^*$, $\mathcal{P}_y^*$, $\varphi_x^{(i)}$, and $\varphi_y^{(i)}$ from the dataset, the mixture model can be read off directly from the InClass net using (32) and (33).

2.2.4. Aggregate classifier

From (26), the aggregate classifier that classifies the individual datapoints based on the full information (x, y) is given by

$$\alpha^{(i)}_{\text{aggregate}}(x,y) = \frac{\frac{\beta_x^{(i)}(x)\,\beta_y^{(i)}(y)}{\sqrt{\varphi_x^{(i)}\,\varphi_y^{(i)}}}}{\sum_{j=1}^{C}\frac{\beta_x^{(j)}(x)\,\beta_y^{(j)}(y)}{\sqrt{\varphi_x^{(j)}\,\varphi_y^{(j)}}}}. \tag{35}$$

3. Results

We provide a public, `tensorflow`-based [61], implementation of InClass nets as a Python 3 package called `RainDancesVI` [62]. The package provides a) routines for wrapping the classifier networks of individual variates into InClass nets, and b) cost functions for training them. It also provides utilities for extracting the model learned by the network post-training. In this section we will demonstrate the working of InClass nets using several toy examples [63] analyzed using `RainDancesVI`. In each case, we assume that the number of components C in the mixture is known *a priori*. The examples considered below are meant for illustration purposes, and were deliberately chosen to require no domain knowledge. At the same time, there are many potential applications of the method to real experimental data, e.g. in astro-particle physics for studying dark matter kinematic substructure in the Milky Way [22], which we are currently pursuing in a separate project.

3.1. Mixture of two independent bivariate Gaussians ($V = 2, C = 2$)

In the first example, we consider the mixture of two independent bivariate Gaussians. In the first component, x and y are both (independently) normally distributed with mean -1 and standard deviation 1.5. The second component is identical, except x and y both have mean $+1$. The mixture weights are taken to be $w_1 = 0.4, w_2 = 0.6$. Table 1 summarizes the mixture model specification and figure 3 shows the normalized joint distributions of (x, y) under each of the two components as heatmaps. Figure 4 shows the normalized joint distribution of (x, y) under the mixture model and our InClass net will estimate the mixture model based on data generated as per this distribution.

The classifier networks $\beta_x^{(i)}$ and $\beta_y^{(i)}$ were constructed using `keras` with the `tensorflow` backend. The NNs are distinct, but have identical architectures. The networks are fairly simple, consisting of three sequential dense layers of 32 nodes using the rectified linear unit (ReLU) [64] activation function. The output layer is a dense layer with two nodes (since $C = 2$), with the softmax activation function. The

Table 1. The mixture model specification for the example considered in section 3.1.

i	w_i	$f_x^{(i)}$	$f_y^{(i)}$
1	0.4	$\mathcal{N}(\text{mean} = -1, \text{SD} = 1.5)$	$\mathcal{N}(\text{mean} = -1, \text{SD} = 1.5)$
2	0.6	$\mathcal{N}(\text{mean} = +1, \text{SD} = 1.5)$	$\mathcal{N}(\text{mean} = +1, \text{SD} = 1.5)$

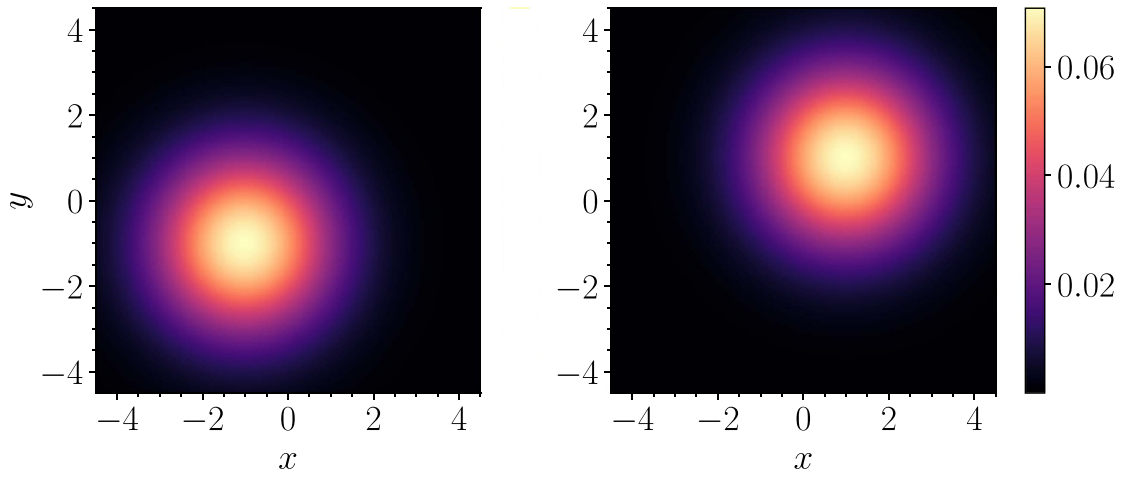


Figure 3. Heatmaps of the normalized joint distributions of (x, y) under component 1 (left panel) and component 2 (right panel) for the example considered in section 3.1.

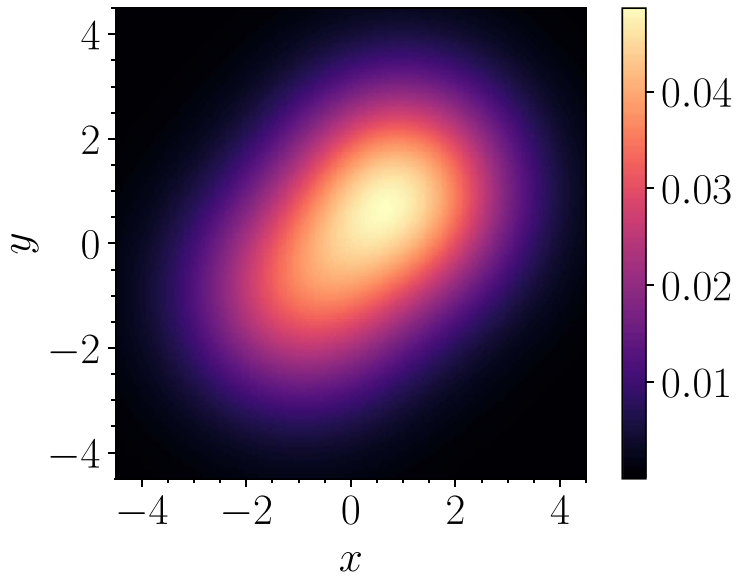


Figure 4. Heatmap of the normalized joint distribution of (x, y) under the mixture model defined in table 1.

individual classifier networks were then wrapped into an InClass net using the `RainDancesVI` package. The resulting network has a total of 4484 trainable parameters.

We trained the InClass net to minimize the negative cross mutual information cost function (29), using 100 000 datapoints sampled from the distribution depicted in figure 4. The optimization was performed for 15 epochs with the `Adam` [65] optimizer (with default hyperparameters) using a batch size of 50. After training the network, we used the same dataset to estimate the pseudo weights $\varphi_x^{(i)}$ and $\varphi_y^{(i)}$ and the mixture weights w_i using (30) and (32). Note that the estimation of mixture models can only be performed up to permutations of the components indexed by i. For clarity of the presentation, unless otherwise stated, the components of the true mixture model will be matched with the respective closest candidates from the machine-learned components. The results of the estimation of the mixture weights are summarized in table 2, which demonstrates an excellent agreement between the true and estimated values.

Table 2. Results of the estimation of the mixture weights for the example considered in section 3.1.

i	Estimated $\varphi_x^{(i)}$	Estimated $\varphi_y^{(i)}$	Estimated w_i	True w_i
1	0.4055	0.4048	0.4051	0.4
2	0.5945	0.5952	0.5949	0.6

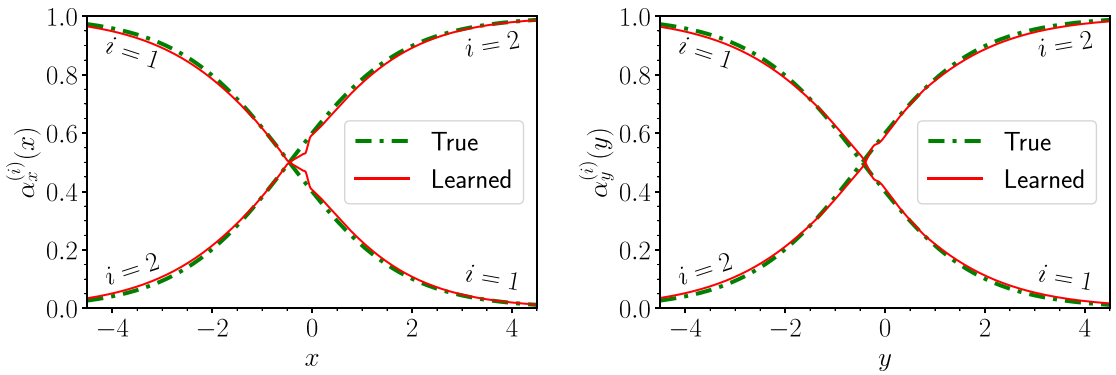


Figure 5. The classifiers $\alpha_x^{(i)}(x)$ (left panel) and $\alpha_y^{(i)}(y)$ (right panel) learned by the network (red solid lines) and the corresponding true classifiers (green dash-dot lines).

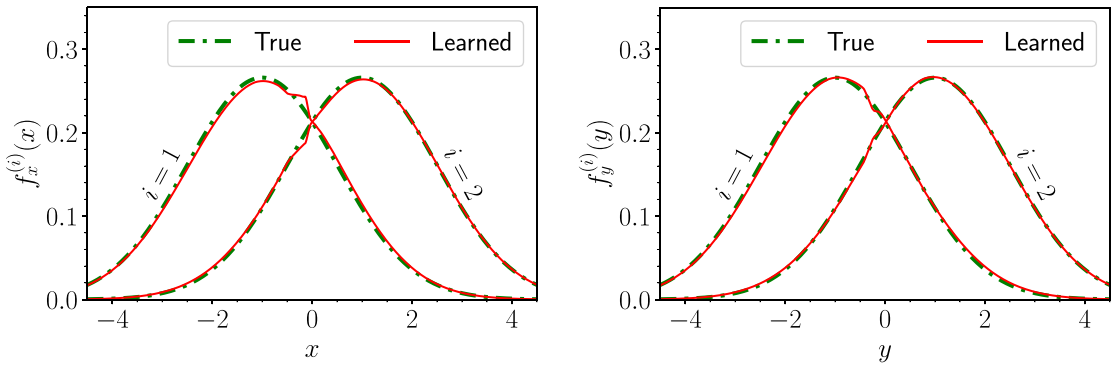


Figure 6. The distributions $f_x^{(i)}$ (left panel) and $f_y^{(i)}$ (right panel). The estimated (true) distributions are shown with red solid (green dash-dot) lines.

The solid red curves in figure 5 depict the classifiers $\alpha_x^{(i)}(x)$ (left panel) and $\alpha_y^{(i)}(y)$ (right panel) learned by the network—they are extracted from $\beta_x^{(i)}$ and $\beta_y^{(i)}$ with the help of (31). For comparison, the true classifiers based on the exact functional forms of the component distributions are also shown as green dash-dot curves. The red solid lines and the green dash-dot lines almost coincide, which validates our method.

Next, we used (33) to estimate the distributions $f_x^{(i)}$ and $f_y^{(i)}$. The resulting distributions are shown with red solid lines in the left and right panels of figure 6, respectively. In applying (33), for simplicity we used the exact expressions for the marginal distributions of x and y in the mixture. In a typical example, the exact expressions for the marginals will not be available, but can be easily estimated from the data, say using a histogram or kernel density estimation [66, 67]. In figure 6, we also show the true $f_x^{(i)}$ and $f_y^{(i)}$ as green dash-dot curves. The good agreement between the true $w_i, f_x^{(i)}$, and $f_y^{(i)}$ and their estimates shown in table 2 and figure 6, demonstrates that the InClass net has successfully estimated the mixture model. Finally, we use (35) to estimate the aggregate classifier $\alpha_{\text{aggregate}}^{(i)}(x,y)$ which is shown as a heatmap in the left panel of figure 7. For comparison, in the right panel we show the true aggregate classifier based on the exact functional forms of the component distributions $f^{(i)}(x,y)$. As expected, the two heatmaps are in very good agreement.

3.2. The checkerboard mixture ($V = 2, C = 2$)

Now we will look at an artificial toy example which was instrumental in the conception and development of the InClass nets technique, see figures 8 and 9. Figure 8 shows the joint distribution of (x, y) for a 'checkerboard' mixture under which the datapoints are uniformly distributed on the bright squares of a 4×4

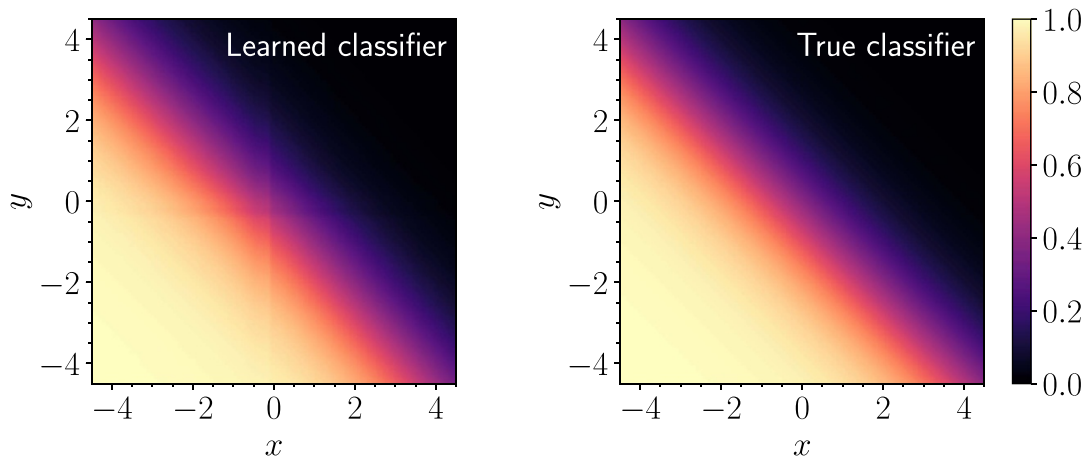


Figure 7. The estimated aggregate classifier $\alpha^{(i)}_{\text{aggregate}}(x,y)$ from (35) (left panel) and the true aggregate classifier (right panel).

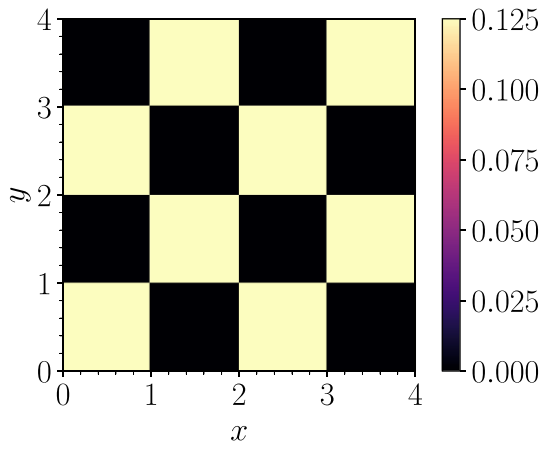


Figure 8. Heatmap illustrating the joint distribution of (x, y) for the 'checkerboard' mixture example considered in section 3.2. The datapoints are uniformly distributed on the bright squares of a 4×4 checkerboard spanning the region $0 \leqslant x, y < 4$, while the dark squares have zero density.

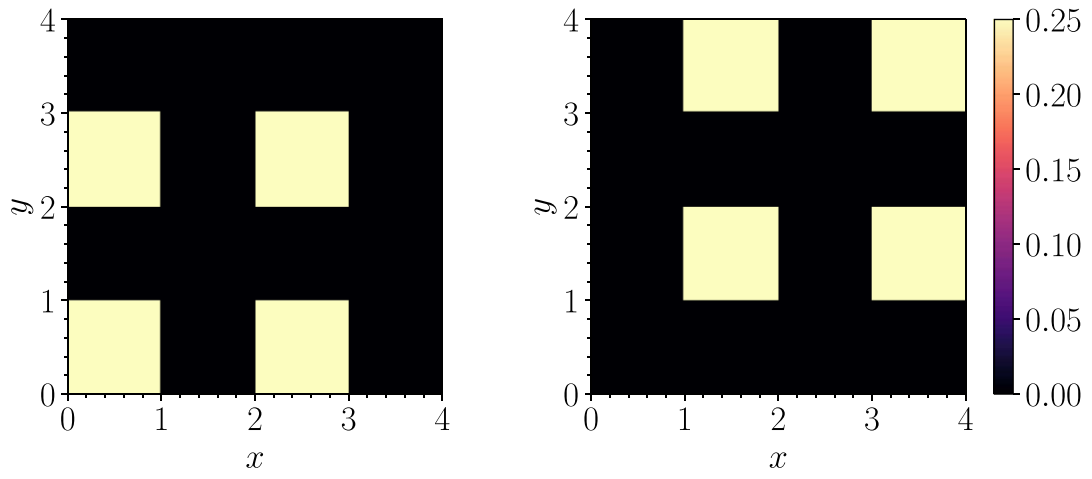


Figure 9. Heatmaps of the normalized joint distributions of (x, y) under component 1 (left panel) and component 2 (right panel) for the 'checkerboard' mixture shown in figure 8.

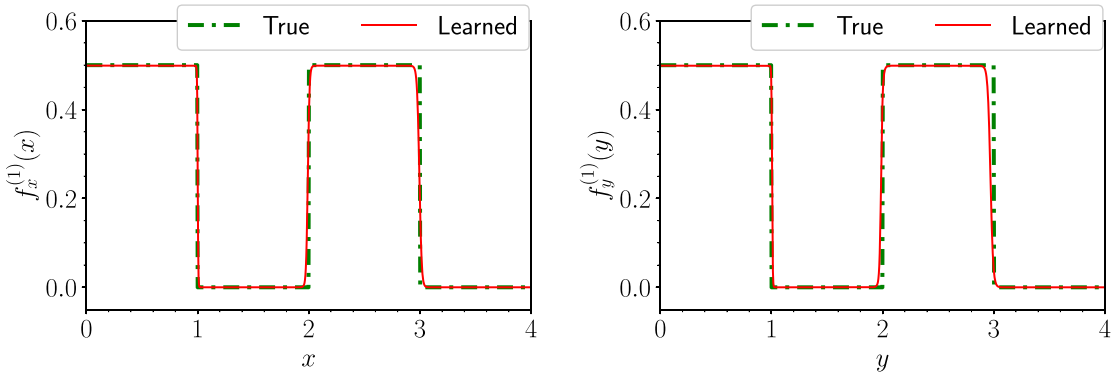


Figure 10. The distributions $f_x^{(1)}$ (left panel) and $f_y^{(1)}$ (right panel) for the 'checkerboard' example considered in section 3.2. The estimated (true) distributions are shown with red solid (green dash-dot) lines. We only show the first component in this figure for the sake of visual clarity (see text).

checkerboard spanning the region $0 \leqslant x, y < 4$, while the dark squares have zero density. For concreteness, the vertical (horizontal) boundaries between cells are assigned to the cell on the right (top). It is easy to see that x and y are, individually, uniformly distributed between 0 and 4. It can also be seen that x and y, despite being uncorrelated, are not mutually independent in the mixture, since x lies within $[0,1) \cup [2,3)$ if and only if y does as well.

As shown in figure 9, the checkerboard mixture can be separated into two equally weighted components within which x and y are mutually independent. Under the first component, x and y both lie within $[0,1) \cup [2,3)$, and under the second component x and y both lie within $[1,2) \cup [3,4)$. Note that each of these components has four spatially disconnected regions—the classification cannot be achieved using spatial clustering techniques. This example also naturally evokes the intuition of the variates x and y serving as each other's supervisory signal, since the value of either x or y uniquely determines the component the datapoint belongs to.

Let us now analyze this toy example using an InClass net. All the details of the network training process are identical to the analysis of the example in section 3.1, including the network architectures, the size of the training dataset, the choice of optimizer, batch size and epoch count. The estimated mixture weights are $w_1 = 0.501, w_2 = 0.499$, which is in excellent agreement with their true values of $w_1 = w_2 = 0.5$. Figure 10 shows, in solid red curves, the distributions of x (left panel) and y (right panel) under the first component learned by the network, using the same procedure as in section 3.1. We only show the first component in this figure for the sake of clarity—the second component fills the gaps in the univariate distributions of x and y so that $w_1 f_x^{(1)} + w_2 f_x^{(2)}$ and $w_1 f_y^{(1)} + w_2 f_y^{(2)}$ are constant. For comparison, the true 'rectangular wave' distributions are also shown as green dash-dot curves, which are also seen to agree with the estimates.

3.3. Semi-supervised training on MNIST data ($V = 2, C = 10$)

The biggest advantage offered by a machine learning based technique over existing non-machine-learning techniques for nonparametric mixture model estimation is the possibility of tackling high-dimensional data. As a proof of concept, in this section we will train an InClass net to classify images of handwritten digits from the MNIST database [68], with the classes corresponding to the digits 0–9. With this example, we will focus more on the data classification aspect of this paper than the mixture model estimation.

The MNIST dataset contains 28px × 28px grayscale images of handwritten digits. Each image also has an associated label indicating the digit contained in the image. We will construct a bivariate mixture model out of the MNIST dataset, where each datapoint is a pair of images. A single datapoint of the dataset will be sampled by first choosing a class between 0 and 9 uniformly at random, and then sampling two images[9] containing that digit uniformly from the MNIST dataset (with replacement). This gives us a bivariate CIMM with ten classes of equal mixture weights—note that within each component (or class), the two images are mutually independent of each other. Figure 11 illustrates the kind of data the InClass net will see, with five randomly chosen datapoints from the mixture model (one in each column).

For analyzing this dataset, instead of creating two different classifiers for the variates, we use the same NN for classifying both x and y. Viewed differently, the networks classifying x and y are identical in architecture

[9] Alternatively, one can generate image pairs by sampling the first image from the MNIST dataset, and applying a random transformation on the sampled image to get the second, as in [32].

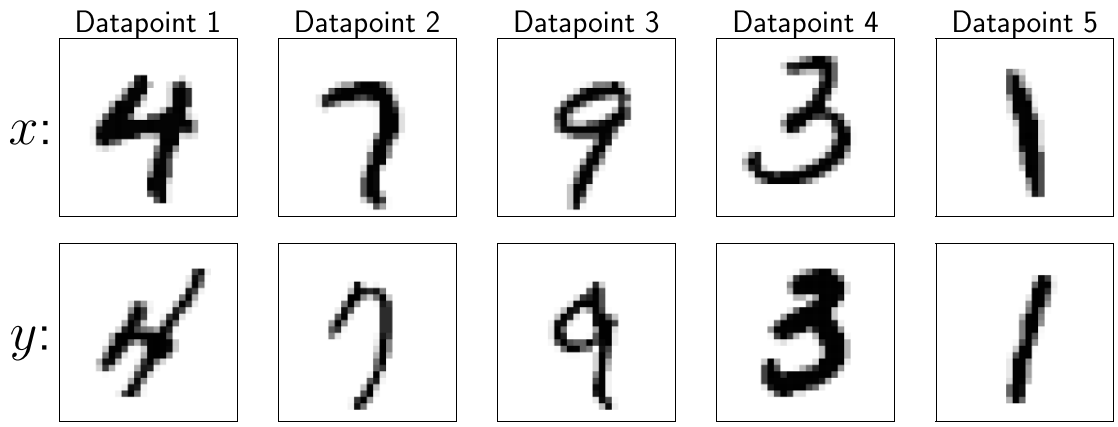


Figure 11. Five representative datapoints from the dataset used to train the InClass net in the example considered in section 3.3. Each datapoint (x, y) is a pair of images containing the same digit.

and share their weights as well, and only differ in the input (output) they receive (return). The network uses a sequential architecture and contains, in order, a layer to flatten the 28×28 image data, three dense layers each with the ReLU activation function and 32 nodes, and finally a dense layer with the softmax activation function and ten nodes (since $C = 10$). This network has a total of 27 562 trainable parameters.

In principle, an InClass net can be trained without supervision to distinguish the digits. However, considering the large number of input dimensions and classes, without supervision, our network is expected to have difficulties 'discovering' new classes in the data, and will end up in bad local minima of the cost function. We will discuss some ways of overcoming this difficulty in appendix B.

In this example, we addressed this issue by taking a semi-supervised approach: We 'seeded' the classes (digits) in the network by performing a supervised training over a small dataset with noisy labels. For this purpose, we used a training dataset containing 2000 images. The noisy label associated with each image matches its true label with probability 0.6, and matches one of the other nine incorrect labels (chosen uniformly) with probability 0.4. The network was trained using the categorical cross-entropy loss function with the `Adam` optimizer for 30 epochs (batch size 20).

After this pre-training, we trained the network further using our `neg_ctc_cost` function on 100 000 pairs of images from our mixture model[10]. Ten percent of the 100 000 datapoints were set aside as a validation dataset to monitor the evolution of the network performance, though no hyperparameter optimization was actively performed using the validation data. The training was done using the `Adam` optimizer with a batch size of 100 for 20 epochs.

Finally, we evaluated the performance of the classifier on a testing dataset of 10 000 single images unseen by the network (either during training or during validation). The performance is illustrated as a confusion matrix in the left panel of figure 12. Each row of the confusion matrix shows the output of the network averaged over test images containing a given digit (true label), both as a heatmap and as numerical values within each cell of the matrix. Recall that our network output, for each image, is 10-dimensional and can be interpreted as the probabilities assigned by the network to the different classes. Because the classes were pre-seeded into the network in a supervised manner, they matched with the true classes without requiring any manual reassignment.

For completeness, in the middle panel of figure 12, we show the confusion matrix of the network after the supervised pre-training performed on the noisily labelled data. Note that the training that resulted in the performance improvement from the middle panel to the left panel was completely unsupervised. We will discuss this semi-supervised training approach in the context of real world applications in section 5.3.

For comparison, in the right panel of figure 12, we show the confusion matrix of a classifier network with an identical architecture that was trained in a fully supervised manner, using noise-free labels (training was performed until the performance on the validation dataset saturated). As can be seen, the network trained using our unsupervised technique (left panel) achieves a comparable performance to the fully supervised classifier (right panel). Their prediction accuracies are 95.07% and 95.79%, respectively[11].

[10] The 200 000 images were all sampled with replacement from a set of 60 000 total images in the MNIST training dataset—repetitions will occur within the dataset.

[11] Note that both training techniques can result in further improved accuracies using more sophisticated network architectures.

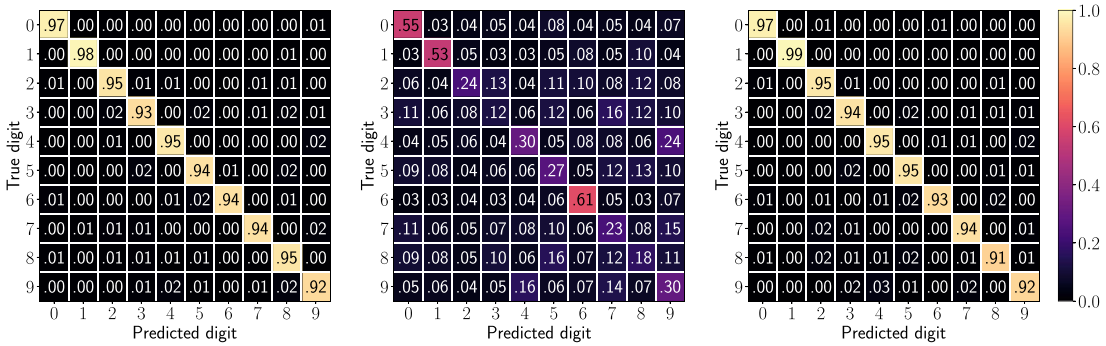


Figure 12. Confusion matrix of the neural network after full training (left panel) and after noisy pre-training (middle panel). Each cell of the confusion matrix shows the average probability assigned by the network for images from a given true class (y-axis) to belong to a given predicted class (x-axis). For comparison, we also include the confusion matrix of a classifier with an identical network architecture that was trained with full supervision using the true labels.

Table 3. The mixture model specification for the example considered in section 3.4. The last column shows the mixture weights estimated by the InClass net technique.

i	w_i	$f_x^{(i)}$	$f_y^{(i)}$	$f_z^{(i)}$	Estimated w_i
1	0.22	$\mathcal{N}(\text{mean} = -1, \text{SD} = 1.5)$	$\mathcal{N}(\text{mean} = -1, \text{SD} = 1.5)$	$\mathcal{N}(\text{mean} = -1, \text{SD} = 1.5)$	0.228
2	0.28	$\mathcal{N}(\text{mean} = +1, \text{SD} = 1.5)$	$\mathcal{N}(\text{mean} = +1, \text{SD} = 1.5)$	$\mathcal{N}(\text{mean} = 0, \text{SD} = 1.5)$	0.268
3	0.18	$\mathcal{N}(\text{mean} = -1.5, \text{SD} = 1.5)$	$\mathcal{N}(\text{mean} = +1.5, \text{SD} = 1.5)$	$\mathcal{N}(\text{mean} = +1, \text{SD} = 1.5)$	0.187
4	0.32	$\mathcal{N}(\text{mean} = +1.5, \text{SD} = 1.5)$	$\mathcal{N}(\text{mean} = -1.5, \text{SD} = 1.5)$	$\mathcal{N}(\text{mean} = +2, \text{SD} = 2.5)$	0.318

3.4. Mixture of four independent trivariate Gaussians ($V = 3, C = 4$)

In this example, we will demonstrate that the InClass nets technique works for the estimation of mixture models with more than two variates as well. We consider the mixture of four independent trivariate Gaussians, with the third variate denoted by z. Table 3 summarizes the mixture model specification. The classifier networks $\beta_x^{(i)}$, $\beta_y^{(i)}$, and $\beta_z^{(i)}$ have a similar architecture to the classifier architectures used in section 3.1, except that the output layer has four nodes, since $C = 4$. The InClass net constructed out of the classifiers has a total of 6924 trainable parameters. We trained the InClass net with the `neg_ctc_cost` function, using 1000 000 datapoints for 15 epochs and a batch size of 500 (the other details of the training process remained the same as in section 3.1), and estimated the mixture model. The estimated mixture weights, shown in the last column of table 3, are in good agreement with the true weights of the components (second column in table 3). The estimated distributions $f_x^{(i)}$, $f_y^{(i)}$, and $f_z^{(i)}$ of the variates x, y, and z, respectively, are shown in figure 13 as red solid curves, along with the true distributions depicted as green dash-dot curves. In all twelve cases (4 components $\times$ 3 variates) we observe good agreement between the true and estimated distribution.

4. Discussion

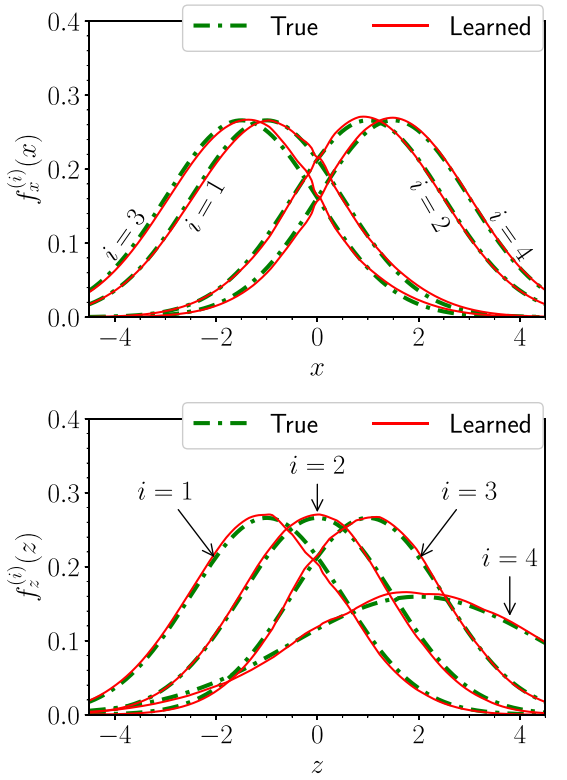
In this section, we will discuss some considerations which might be relevant in the context of science applications of the InClass nets technique introduced in this paper.

4.1. Identifiability of conditional independence bivariate mixture models

Identifiability of a statistical model is concerned with whether the parameters and functions that describe the model are uniquely identifiable from an infinite sample of datapoints produced from the model. The identifiability of mixture models is an important concept, especially in the context of using the techniques introduced in this paper for science applications.

For a given statistical model, the definition of what it means to estimate the model is usually chosen to be practically useful. In the context of estimating nonparametric CIMMs, the definition allows for the following 'leniencies':

- The number of components C is assumed to be known *a priori*. Otherwise, any CIMM will be unidentifiable since a given component i can always be split into several new components which share the same distribution



i	True w_i	Estimated w_i
1	0.22	0.228
2	0.28	0.268
3	0.18	0.187
4	0.32	0.318

Figure 13. The distributions $f_x^{(i)}$ (top-left panel), $f_y^{(i)}$ (top-right panel), and $f_z^{(i)}$ (bottom-left panel) for the example considered in section 3.4. The estimated (true) distributions are shown with red solid (green dash-dot) lines.

$f^{(i)}(\mathcal{X})$ (with weights adding up to the weight of the 'parent' component). Similarly, zero weight components can always be added without affecting the data distribution.

- The model only needs to be (and can only ever be) estimated up to permutations of the component indices.
- The distribution $f_v^{(i)}$ is considered to be the same as the distribution $g_v^{(i)}$ if $f_v^{(i)}(x_v) = g_v^{(i)}(x_v)$ 'almost surely'. In other words, $f_v^{(i)}$ and $g_v^{(i)}$ are allowed to be different over a set of probability measure 0. This is required for the nonparametric case which allows arbitrary $f_v^{(i)}$-s.

The $(V=1, C\geqslant 2)$ case (univariate) is always unidentifiable nonparametrically. For the $(V\geqslant 3, C=2)$ case, [33] provided certain regularity conditions under which instances of CIMMs are identifiable, and [42] generalized the result to the $(V\geqslant 3, C\geqslant 2)$ case. The result from [42] states that an instance of a CIMM with $V\geqslant 3$ and $C\geqslant 2$ is identifiable if the functions $\left\{f_v^{(1)},\ldots,f_v^{(C)}\right\}$ are linearly independent, for all $v=1,\ldots,V$.

This leaves the bivariate case $(V=2)$, which is the main focus of this section. reference [33] showed that in the $(V=2, C=2)$ case, instances of nonparametric CIMMs are not identifiable in general. In particular it was shown that for any instance of a two component bivariate nonparametric CIMM, there exists a two-parameter family of instances which leads to the same distribution of the observed variables (x, y). The authors also noted that non-negativity conditions will introduce constraints on the allowed values for the two parameters. Extending this result from [33], we derive the following two theorems which provide a sufficient and a (different) necessary condition for instances of nonparametric CIMMs with $(V=2, C\geqslant 2)$ to be identifiable. The two conditions coincide for the $C=2$ case. We relegate the proof of the theorems to appendix A.

Theorem 1 (Necessary condition). *A nonparametric conditional independence bivariate $(V=2)$ mixture model with $C\geqslant 2$ components of the form given in (28) is uniquely identifiable up to permutations of the component-identities only if the following necessary condition is satisfied:*

$$\operatorname{ess\,sup}\left[\frac{w_i f_t^{(i)}(t)}{w_i f_t^{(i)}(t)+w_j f_t^{(j)}(t)}\right] \equiv \operatorname{ess\,sup}\left[\frac{\alpha_t^{(i)}(t)}{\alpha_t^{(i)}(t)+\alpha_t^{(j)}(t)}\right] = 1\,, \qquad \forall (i,j)\in\{1,\ldots,C\}^2\ :\ i\neq j,\ \ \forall t\in\{x,y\}\,, \tag{36}$$

where $\operatorname{ess\,sup}[\mathtt{func}(t)]$ *represents the essential supremum of* $\mathtt{func}(t)$. □

Theorem 2 (Sufficient condition). *A nonparametric conditional independence bivariate ($V=2$) mixture model with $C \geqslant 2$ components of the form given in (28) is uniquely identifiable up to permutations of the component-identities if the following sufficient condition is satisfied:*

$$\operatorname{ess\,sup}\left[\frac{w_i f_t^{(i)}(t)}{\mathcal{P}_t(t)}\right] \equiv \operatorname{ess\,sup}\left[\alpha_t^{(i)}(t)\right] = 1\,, \qquad \forall i \in \{1,\ldots,C\}\,, \forall t \in \{x,y\}\,, \tag{37}$$

where, as before, ess sup$[\texttt{func}(t)]$ represents the essential supremum of $\texttt{func}(t)$. □

The essential supremum can be thought of as an adaptation of the notion of supremum of a function, allowing for ignoring the behaviour of the function over regions with a total probability measure[12] of zero. These conditions can 'roughly' be interpreted as follows: The sufficient condition (37) will be satisfied if, for every component i and variate x or y, there exists some region in the phase space of the variate where component i *completely* dominates the mixture, i.e. all the datapoints in that region are from component i. The necessary condition (36) will be satisfied if, for every pair of components $i \neq j$ and variate x or y, there exists some region in the phase space of the variate where component i 'completely' dominates the mixture of components i and j.

Let us now revisit the examples considered earlier in section 3 from the point of view of identifiablilty. Considering the successful estimation of mixture models and/or classifier training in those examples, we can expect them to be identifiable. For the mixture of two independent bivariate Gaussians, figure 5 shows how, for both x and y, the true and reconstructed classifier output for the first (second) component approaches 1 for increasingly negative (positive) values. This ensures that the sufficient condition for identifiability (37) is satisfied. Similarly, for the checkerboard mixture, by construction, there are regions in x and y which contain points from only component 1 or only component 2, see figure 9.

Recall that in our treatment, the individual variates x and y are themselves allowed to be multi-dimensional. In the special case of one-dimensional variates x and y, typically a component will only dominate the mixture in either the left tail or the right tail of the other components. This means that for most natural examples with one-dimensional x and y, it is unlikely for the mixture to be identifiable for more than two components. On the other hand, this limitation does not apply to higher dimensional variates x and y which our InClass nets specialize in. For instance, the sufficient condition (37) for the mixture model constructed out of the MNIST dataset becomes: 'For every digit d, there must exist some region in the space of images, within which the images look unmistakably like the digit d'. This condition is naturally expected to be satisfied, considering the reliability of good handwritten communication.

4.1.1. Reduced identifiability due to limited statistics

The unique estimation of mixture model instances guaranteed by theorem 2 can only be achieved with an infinite dataset. There will always be an uncertainty associated with estimation performed using finite datasets [69]. The level of this uncertainty is related to (among other things) how close the conditions (36) and (37) are to being satisfied within the region of sample space covered sufficiently by the finite dataset at hand. In this sense, the result in theorem 2 is useful from a practical point of view. To illustrate this, we repeated the two Gaussians example from section 3.1, with the same setup, but with much fewer datapoints, namely 5000 instead of 100 000. With fewer datapoints, the dataset is less likely to probe the tails of the x and y distributions, where a single component dominates. As expected, this time the estimation of the weights is slightly worse—we obtained $w_1 = 0.44$ and $w_2 = 0.56$, to be compared with the true values of $w_1 = 0.4$ and $w_2 = 0.6$. The results for the classifiers $\alpha_x^{(i)}(x)$ and $\alpha_y^{(i)}(y)$ and for the component distributions $f_x^{(i)}$ and $f_y^{(i)}$ are shown in figures 14 and 15, respectively. Comparing to the analogous high statistics figures 5 and 6, we see that the estimation has generally succeeded (after all, the model was identifiable), but is not perfect and suffers from statistical uncertainties.

4.1.2. Unidentifiable situations

When a CIMM instance is not identifiable, our technique will yield one of the parameterizations (weights and functions) that best fits the available data. Note that the unidentifiability of an instance of a nonparametric CIMM is not a weakness of our InClass nets approach, but rather a statement on the impossibility of the task of unique estimation.

4.2. Uncertainty quantification

An important aspect of estimating a model (or equivalently, fitting a model to the available data), is providing an uncertainty on the estimate. Uncertainties in parametric estimation are conceptually

[12] It is understood that the relevant probability measure in (36) and (37) is the one that corresponds to the mixture model itself.

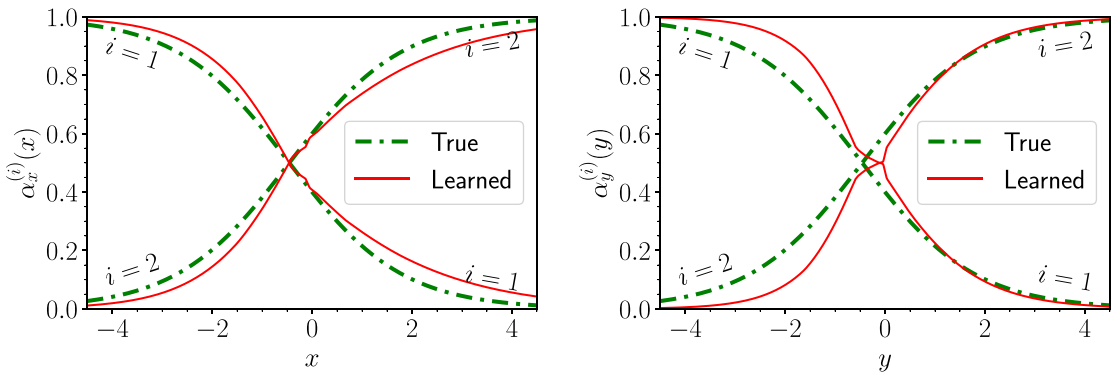


Figure 14. The same as figure 5, but using only 5000 events for the estimation of the mixture model considered in section 3.1.

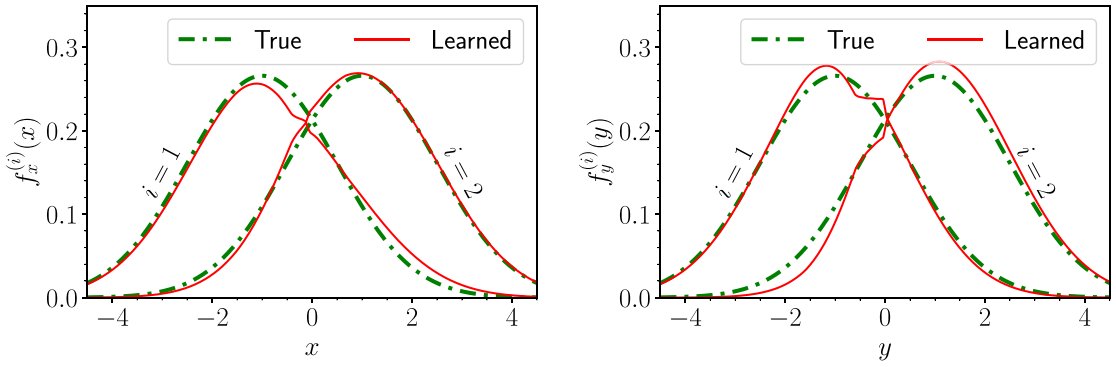


Figure 15. The same as figure 6, but using only 5000 events for the estimation of the mixture model considered in section 3.1.

straightforward—they correspond to the (possibly correlated) uncertainties in the estimated values of the parameters. The corresponding approach in the context of nonparametric models (which allow arbitrary functions), would be to treat either the NN outputs $\beta_v^{(i)}$ or the estimated distributions $f_v^{(i)}$ as Gaussian processes [70]. For a Gaussian process $G(\texttt{input})$, the value of G at any finite set of `input` values is taken to be randomly distributed according to a multivariate normal distribution. This allows us to assign uncertainty estimates on the value of G at individual `input` points, while also accounting for the correlations in the uncertainties between the values at different `input` points. It has been shown that Gaussian processes can be modeled using Bayesian NNs with wide layers [71–73]. Using wide Bayesian NNs as the individual classifiers of the InClass net, one can obtain robust uncertainties on the estimated mixture model. In many scenarios, one is simply interested in visualizing a band of uncertainty around the estimated $\beta_v^{(i)}$-s, $\alpha_v^{(i)}$-s, or $f_v^{(i)}$-s and even narrow Bayesian NNs may be sufficient for this purpose.

Note that the Gaussian process approach will work when the instance of the CIMM at hand is identifiable, and the uncertainty in the estimated model arises only from the finiteness of the dataset being analyzed. It is presently unclear whether Bayesian NNs can capture the degree(s) of freedom in the model specification which are introduced by the unidentifiability of the CIMM instance.

4.3. Estimating the number of components C

In many applications, one does not *a priori* know the number of components in the CIMM [37, 74]. In other situations, the assumption that the distribution of the data can be written as a CIMM may not necessarily be valid. In such situations, by training different InClass nets with different values of C, one may be able to a) verify the validity of the conditional independence assumption, and b) estimate C.

Note that increasing the number of components increases the fitting ability of a CIMM. More concretely, every CIMM instance with C components can be thought of as a CIMM instance with $C' > C$ components (with $C' - C$ additional zero-weight components). As a result, an InClass net with more components should strictly perform better (in terms of the minimum cost value achieved), up to network training deficiencies and statistical fluctuations due to the finiteness of the training dataset. However, the improvement (in the minimum cost achieved) resulting from increasing C is expected to diminish beyond a certain point.

In particular, if the true probability distribution of the data $\mathcal{P}^*(\mathcal{X})$ can be modeled as a CIMM, then there exists a minimum number of components C_{min} required to express $\mathcal{P}^*(\mathcal{X})$ in the form

$$\mathcal{P}^*(\mathcal{X}) = \sum_{i=1}^{C_{\text{min}}} w_i \prod_{v=1}^{V} f_v^{(i)}(x_v)\,. \tag{38}$$

Increasing C from 1 to C_{min} will show an improvement in `neg_ctc_cost` value, but beyond C_{min}, the performance is expected to saturate. This feature, if observed, can simultaneously a) confirm that the data is consistent with the conditional independence assumption, and b) provide an estimate of C_{min}. Note that the C_{min} value identified in this way is only an estimate—inferring the presence of a component with a small mixing weight, or the presence of two components with very similar distributions $f^{(i)}(\mathcal{X})$ may be statistically limited by the amount of data available. If the actual number of components is *a priori* unknown, then the C_{min} estimate can serve as an Occam's razor estimate of C.

On the other hand, if such a sharp saturation of network performance is not observed at a particular value of C, and the saturation is more gradual, this could be a sign of a Latent Factor Model—the underlying latent variable that explains the dependence of the different variates could be continuous instead of being the discrete category label i.

4.4. Minimum possible value of `neg_ctc_cost`

When estimating C_{min} using the method described in section 4.3, one relies on observing a saturation in the value of the minimum cost achieved. However, such a saturation could also result from deficiencies in the architecture and/or training of the network. It it therefore useful to have an estimate of the minimum possible `neg_ctc_cost` achievable by the best fitting model. Recall from (24*a*), that

$$\texttt{neg\_ctc\_cost} = \text{KL}\left[\mathcal{P}^* \,\middle|\middle|\, \mathcal{P}\right] - C^*(x_1,\ldots,x_V)\,, \tag{39}$$

where $\text{KL}\left[\mathcal{P}^* \,\middle|\middle|\, \mathcal{P}\right]$ is the KL divergence from the distribution represented by the InClass net $\mathcal{P}$ to the true distribution $\mathcal{P}^*$ and $C^*(x_1,\ldots,x_V)$ is the total correlation of the variates under the true distribution. Since, the KL divergence is manifestly non-negative and equals 0 only when $\mathcal{P}^*$ is equivalent to $\mathcal{P}$, we have the following inequality

$$\texttt{neg\_ctc\_cost} \geqslant -C^*(x_1,\ldots,x_V)\,, \tag{40}$$

where the equality is achieved when $\mathcal{P}^*$ matches $\mathcal{P}$ *almost surely*. Thus, the negative total correlation $-C^*$ provides a (theoretically achievable) lower-bound on the negative cross total correlation `neg_ctc_cost`. From (23) the total correlation C^* is given by

$$C^*(x_1,\ldots,x_V) = \int d\mathcal{X}\; \mathcal{P}^*(\mathcal{X})\; \log\left[\frac{\mathcal{P}^*(\mathcal{X})}{\mathcal{P}_1^*(x_1)\;\mathcal{P}_2^*(x_2)\;\ldots\;\mathcal{P}_V^*(x_V)}\right], \tag{41}$$

where $\mathcal{P}_v^*$ represents the marginal distribution of x_v in the data. For low dimensional data, C^* can be estimated directly using this formula, after first estimating the distributions $\mathcal{P}^*(\mathcal{X})$ and $\mathcal{P}_v^*(x_v)$.

Alternatively, for both low and high dimensional data, one can estimate C^* using supervised machine learning as follows. Let the distribution $\mathcal{Q}^*(\mathcal{X})$ be defined as

$$\mathcal{Q}^*(\mathcal{X}) \equiv \prod_{v=1}^{V} \mathcal{P}_v^*(x_v). \tag{42}$$

Note that C^* is simply the Kullback–Leibler divergence $\text{KL}\left[\mathcal{P}^* \,\middle|\middle|\, \mathcal{Q}^*\right]$ from $\mathcal{Q}^*$ to $\mathcal{P}^*$. One can produce datapoints as per the distribution $\mathcal{Q}^*(\mathcal{X})$ by independently sampling the variates $x_1,\ldots,x_V$ from the available dataset. This gives us two datasets: the original one distributed as per $\mathcal{P}^*$ and a resampled one distributed as per $\mathcal{Q}^*$. One can train a machine in a supervised manner to distinguish between these two datasets and estimate the KL divergence, and hence C^*, from the trained classifier.

4.5. Incorporating prior knowledge

In some applications, one may have additional prior knowledge about the mixture model, beyond the conditional independence assumption. It may be possible to incorporate this knowledge into the InClass net directly. For example, in the MNIST image classification example considered in section 3.3, we used the information that the variates x and y are both images of digits to use the same classifier NN for both variates.

As a different example, if the distribution of a given variate x_v is known under a given component i, then the value of $\beta_v^{(i)}$ can be set to $f_v^{(i)}(x_v)/\mathcal{P}_v^*(x_v)$ up to a multiplicative weight factor which will constitute a single, trainable parameter. For the special case where the distribution of a given variate x_v is known under every component, the classifier for the vth variate can be parameterized by only the mixture weights of the components—in this way the InClass nets technique can be applied in the situations where the ${}_s\mathcal{P}lots$ technique is currently being used in high energy physics.

As yet another example, consider the case where the weights of the different components are *a priori* known (but not the distributions of the variates within the components). Then an extra term can be added to the cost function to force the mixture weights w_i estimated by the InClass net towards the true known weights w_i^{true}. One possible form of the extra term is inspired by the cross entropy:

$$-\lambda \sum_{i=1}^{C} w_i^{\text{true}} \log(w_i)\,, \tag{43}$$

where λ is a parameter that controls the relative importance of the new term in the cost function. The additional term could be added either at the beginning, or after training the network for a few epochs (and identifying the map from the true component indices to the learned component indices). The additional term may be particularly useful in estimating unidentifiable CIMMs, where the additional knowledge of the mixture weights could help rule out the observationally indistinguishable 'fake' CIMM instances.

5. Possible variations and extensions

In this section we will discuss some potential variations and extensions of the InClass nets technique introduced in this paper whose detailed exploration is beyond the scope of this work. Additionally, in appendix C, we provide some surrogate cost functions for training CIMMs, which are already implemented in the `RainDancesVI` package.

5.1. Regularizers

Recall that the dataset at hand could be consistent with multiple CIMM instances, either due to the unidentifiability of the instance or due to the finiteness of the dataset. In such situations, one can impose additional conditions for the learned model to satisfy. For example, depending on the application at hand, one might be interested in roughly evenly weighted components. This can be encouraged by adding (to the cost function) additional regularization terms like

$$\texttt{tikhonov\_reg} = \lambda \sum_{i=1}^{C} w_i^2, \tag{44a}$$

$$\texttt{neg\_shannon\_reg} = \lambda \sum_{i=1}^{C} w_i \log w_i, \tag{44b}$$

where λ is a positive constant. If one is interested in more lopsided weight distributions for the components (possibly suppressing the weights of some components), the same regularizer terms can be used with λ set to a negative value.

5.2. Unsupervised classification with multi-label InClass nets

The InClass nets architecture introduced in this paper can have more general data mining applications beyond the estimation of CIMMs. If the datapoints in a dataset are comprised of the (possibly multi-dimensional) variates $x_1, \ldots, x_V$, the joint distribution of these variates may be understandable in terms of the classes of datapoints within the dataset, even if it does not fall under a CIMM. Furthermore, the set of classes corresponding to one variate need not necessarily be the same as the set of classes corresponding to another. In the literature, the existence of different sets of classes within the dataset falls under the realm of 'multi-label classification'.

For example, consider a dataset containing paired data: each datapoint contains the identities of a book and a movie liked by a person. The working assumption could be that there exists a classification of books and a (different) classification of movies, such that the class of books liked by a person is related to the class of movies liked by the same person. In such cases, it may be possible to simultaneously train a book and a movie classifier using the InClass nets architecture, by simply maximizing the mutual information between the classes predicted by the network.

To this end, we define the 'negative total correlation function' `neg_tc_cost` and its bivariate special case 'negative mutual information' cost function `neg_mi_cost` as

$$\texttt{neg\_tc\_cost} = -\sum_{i_1=1}^{C_1}\sum_{i_2=1}^{C_2}\cdots\sum_{i_V=1}^{C_V} E_{\mathcal{P}^*}\left[\prod_{v=1}^{V}\alpha_v^{(i_v)}\right]\log\left[\frac{E_{\mathcal{P}^*}\left[\prod_{v=1}^{V}\alpha_v^{(i_v)}\right]}{\prod_{v=1}^{V}E_{\mathcal{P}^*}\left[\alpha_v^{(i_v)}\right]}\right], \tag{45a}$$

$$\texttt{neg\_mi\_cost} = -\sum_{i=1}^{C_x}\sum_{j=1}^{C_y} E_{\mathcal{P}^*}\left[\alpha_x^{(i)}\alpha_y^{(j)}\right]\log\left[\frac{E_{\mathcal{P}^*}\left[\alpha_x^{(i)}\alpha_y^{(j)}\right]}{E_{\mathcal{P}^*}\left[\alpha_x^{(i)}\right]E_{\mathcal{P}^*}\left[\alpha_y^{(j)}\right]}\right], \tag{45b}$$

where the outputs of the InClass net $\eta_v^{(i)}$ are directly interpreted as the classifier output $\alpha_v^{(i)}$, and C_v is the number of classes for the classifier corresponding to the vth variate—note that the C_v-s need not all be equal. We point out that the `neg_tc_cost` of (45*a*) is a generalization of the cost function used in [50] for the case where a) there can be more than two variates in the data, b) the classifiers $\alpha_v^{(i)}$ are not necessarily the same for different variates, and c) the number of classes C_v could be different for different variates. Also, in this formulation, it is not required that the inputs x_v to the different classifiers have only non-overlapping attributes of the datapoint.

5.3. Semi-supervised classification with InClass nets

In the MNIST image classification example considered in section 3.3, we seeded the categories into the classifier network via supervised learning using a small, noisily labeled dataset. After the categories were seeded in, we used the unsupervised training of the InClass nets technique to further train the network.

This strategy has straightforward applications in semi-supervised learning scenarios where only a subset of the datapoints in the training dataset are labeled. For example, in the training of NNs to perform medical diagnosis [75], generating labeled datasets requires manual annotation by experts, and only a small number of labeled samples may be available. On the other hand, a large number of unlabeled samples are typically available for training purposes. If, say, two different aspects (or variates) of the medical records are expected to only be weakly dependent on each other, but a confounding factor like the presence or absence of a disease can influence both variates, then we can train a NN to perform the diagnosis leveraging both the labeled and unlabeled datasets. A hybrid cost function that incorporates a supervised classification cost function (for the labeled datapoints), as well an unsupervised cost function introduced in this paper (for the unlabeled datapoints) may be appropriate for the task.

Note that the medical diagnosis example considered here will not strictly be a CIMM. For instance, in addition to the presence or absence of the disease, the severity of a particular case is also likely to influence the medical record. It may be possible to accommodate this particular effect by having multiple labels for different severity levels. Despite not strictly being an example of conditional indepedence mixture model, training using the `neg_ctc_cost` or the `neg_tc_cost` can still potentially yield useful diagnostic tools.

6. Summary

In this paper we introduced a novel approach for the 'nonparametric' estimation of CIMMs defined by (3). In this approach, the estimation of a CIMM is treated as a multi-class classification problem, which we solve with machine-learning methods. The main results of the paper are as follows.

- We develop a specific machine-learning technique which we call the InClass nets technique. The basic architecture of InClass nets is illustrated in figure 1 and consists of a number of classifiers (one for each variate), which are realized as artificial NNs.
- In section 2.1, we show how CIMMs can be represented using InClass nets. The ability of NNs to approximate arbitrary functions allows for the 'nonparametric' modeling of the CIMM.
- We recast the problem of estimation of a CIMM as a classification problem, and construct suitable cost functions for training the individual NNs without supervision. We also provide the prescription for extracting the learned CIMM from the trained InClass nets. The efficacy of our procedure is demonstrated with several toy examples in section 3, including a high-dimensional image classification problem.
- For easy adoption of the InClass nets technique, we provide a public implementation of our method as a Python package called `RainDancesVI` [62].

- In section 4.1 we derive some new results on the nonparametric identifiability of bivariate CIMMs, in the form of a necessary and a (different) sufficient condition for a bivariate CIMM to be identifiable. The proofs of the theorems can be found in appendix A.

While performing comparative studies with previously existing (non-machine-learning) estimation techniques in the literature is beyond the scope of this work, we note that (for a given C and V) a machine-learning technique like InClass nets can potentially handle higher dimensional variates than non-machine-learning techniques. As discussed in sections 4 and 5, the InClass nets technique has many potential applications beyond the narrow focus of CIMM. Specifically, the use of machine learning opens new avenues for addressing old-standing problems in nonparametric statistics.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://gitlab.com/prasanthcakewalk/code-and-data-availability/.

Acknowledgments

The authors would like to thank M Lisanti and A Roman for useful discussions. The work of P S was supported in part by the University of Florida CLAS Dissertation Fellowship (funded by the Charles Vincent and Heidi Cole McLaughlin Endowment) and the Institute of Fundamental Theory Fellowship. This work was supported in part by the United States Department of Energy under Grant No. DE-SC0010296.

Appendix A. Proof of theorems 1 and 2

Here we will prove theorems 1 and 2. Let us begin by noting that:

- A nonparametric CIMM can be identifiable only if all the mixture weights are non-zero—if one of the mixture components has zero weight, it can be removed from the mixture and different component can be split into two.
- A nonparametric CIMM with $V=2$ cannot be identifiable if there exists a pair of components i,j for which $f_x^{(i)}(x)=f_x^{(j)}(x)$ almost surely. Otherwise, the sub-mixture of the components i and j, $w_i f_x^{(i)}(x) f_y^{(i)}(y) + w_j f_x^{(j)}(x) f_y^{(j)}(y)$, can be rewritten as a different combination of two components of total weight $w_i + w_j$ which have the same distribution of the variate x as the original components, but different mixture weights and distributions of the variate y.
- Similarly, a nonparametric CIMM with $V=2$ cannot be identifiable if there exists a pair of components i,j for which $f_y^{(i)}(y)=f_y^{(j)}(y)$ almost surely.

Congruently, neither the necessary nor the sufficient condition from theorems 1 and 2 can be satisfied if one of the mixing weights is zero, or if $\exists(i,j,t): f_t^{(i)}(t)=f_t^{(j)}(t)$ *almost surely*. Henceforth, we will only consider instances of nonparametric bivariate CIMMs for which

$$w_i > 0\,, \qquad \forall i \in \{1,\ldots,C\}\,, \tag{A.1a}$$

$$\left(f_x^{(i)} - f_x^{(j)} = 0 \text{ almost surely}\right) \;\Rightarrow\; \left(i=j\right), \tag{A.1b}$$

$$\left(f_y^{(i)} - f_y^{(j)} = 0 \text{ almost surely}\right) \;\Rightarrow\; \left(i=j\right). \tag{A.1c}$$

A.1. Two component case ($C=2$)

Let us first tackle the $C=2$ case of theorems 1 and 2. Throughout this section, equality of distributions will refer to their equality *almost surely*. From theorems 4.1 and 4.2 of [33], for every instance of parametric bivariate CIMM of the form given in (28), all the instances with the same distribution of observed data form a two-parameter family. This family of instances identified in [33] can be parameterized in terms of $\gamma \in \mathbb{R}$ and $0 \leqslant w_1' \leqslant 1$, and can be written as

$$\sum_{i=1}^{2} w_i' g_x^{(i)}(x) g_y^{(i)}(y) = \sum_{i=1}^{2} w_i f_x^{(i)}(x) f_y^{(i)}(y)\,, \tag{A.2}$$

where

$$w_2' = 1 - w_1'\,, \tag{A.3a}$$

$$g_x^{(1)}(x) = \mathcal{P}_x(x) + \gamma w_2' \sqrt{\frac{w_1\, w_2}{w_1'\, w_2'}} \left(f_x^{(1)}(x) - f_x^{(2)}(x) \right), \tag{A.3b}$$

$$g_x^{(2)}(x) = \mathcal{P}_x(x) - \gamma w_1' \sqrt{\frac{w_1\, w_2}{w_1'\, w_2'}} \left(f_x^{(1)}(x) - f_x^{(2)}(x) \right), \tag{A.3c}$$

$$g_y^{(1)}(y) = \mathcal{P}_y(y) + \frac{w_2'}{\gamma} \sqrt{\frac{w_1\, w_2}{w_1'\, w_2'}} \left(f_y^{(1)}(y) - f_y^{(2)}(y) \right), \tag{A.3d}$$

$$g_y^{(2)}(y) = \mathcal{P}_y(y) - \frac{w_1'}{\gamma} \sqrt{\frac{w_1\, w_2}{w_1'\, w_2'}} \left(f_y^{(1)}(y) - f_y^{(2)}(y) \right). \tag{A.3e}$$

Note, that the transformation $\gamma \longleftrightarrow -\gamma, w_1' \longleftrightarrow 1 - w_1'$ is equivalent to a permutation of the component indices $(1) \longleftrightarrow (2)$. Since, we are only interested in the identifiability of the CIMM instance upto this permutation, we can restrict γ to be non-negative. The only additional constraints on γ and w_1' are provided by the non-negativity of the distribution functions $g_x^{(i)}$ and $g_y^{(i)}$.

It can be verified that $\gamma = 1$ and $w_1' = w_1$ corresponds to the original CIMM instance with $g_x^{(i)} = f_x^{(i)}$ and $g_y^{(i)} = f_y^{(i)}$. Furthermore, any other set of values for γ and w_1' corresponds to a different instance, since the $w_1, w_2 > 0$ and the differences $f_x^{(1)} - f_x^{(2)}$ and $f_y^{(1)} - f_y^{(2)}$ are not identically zero. This leads us to the following lemma: The CIMM instance will be identifiable if and only if the non-negativity constraints on $g_x^{(i)}$ and $g_y^{(i)}$ only allow γ and w_1' to be 1 and w_1, respectively.

The non-negativity conditions on the functions $g_x^{(i)}$ and $g_y^{(i)}$ can be written using (A.3) as

$$\frac{1}{\gamma}\sqrt{\frac{w_1'}{w_2'}} \geqslant \sqrt{w_1\, w_2}\ \operatorname{ess\,sup}\left[\frac{f_x^{(2)}(x) - f_x^{(1)}(x)}{\mathcal{P}_x(x)}\right] = \frac{\mu_x^{(2)} - w_2}{\sqrt{w_1\, w_2}}\,, \tag{A.4a}$$

$$\frac{1}{\gamma}\sqrt{\frac{w_2'}{w_1'}} \geqslant \sqrt{w_1\, w_2}\ \operatorname{ess\,sup}\left[\frac{f_x^{(1)}(x) - f_x^{(2)}(x)}{\mathcal{P}_x(x)}\right] = \frac{\mu_x^{(1)} - w_1}{\sqrt{w_1\, w_2}}\,, \tag{A.4b}$$

$$\gamma\sqrt{\frac{w_1'}{w_2'}} \geqslant \sqrt{w_1\, w_2}\ \operatorname{ess\,sup}\left[\frac{f_y^{(2)}(y) - f_y^{(1)}(y)}{\mathcal{P}_y(y)}\right] = \frac{\mu_y^{(2)} - w_2}{\sqrt{w_1\, w_2}}\,, \tag{A.4c}$$

$$\gamma\sqrt{\frac{w_2'}{w_1'}} \geqslant \sqrt{w_1\, w_2}\ \operatorname{ess\,sup}\left[\frac{f_y^{(1)}(y) - f_y^{(2)}(y)}{\mathcal{P}_y(y)}\right] = \frac{\mu_y^{(1)} - w_1}{\sqrt{w_1\, w_2}}\,, \tag{A.4d}$$

where

$$\mu_t^{(i)} = \operatorname{ess\,sup}\left[\frac{w_i f_t^{(i)}(t)}{w_1 f_t^{(1)}(t) + w_2 f_t^{(2)}(t)}\right], \qquad \forall i \in \{1,2\}, \forall t \in \{x,y\}\,. \tag{A.5}$$

It can seen from (A.4) that the $\mu_t^{(i)}$-s satisfy the constraints $w_i \leqslant \mu_t^{(i)} \leqslant 1$, since the essential supremum of the difference between two normalized distributions is non-zero—normalized equations have to cross or be

equal almost surely. Now, multiplying (A.4*a*) with (A.4*c*), and (A.4*b*) with (A.4*d*) we get the following constraint in the ratio w_1'/w_2'

$$\frac{\left(\mu_x^{(2)}-w_2\right)\left(\mu_y^{(2)}-w_2\right)}{w_1\,w_2}\leqslant\frac{w_1'}{w_2'}\leqslant\frac{w_1\,w_2}{\left(\mu_x^{(1)}-w_1\right)\left(\mu_y^{(1)}-w_1\right)}\,. \tag{A.6}$$

Note that all values w_1'/w_2' allowed by this constraint are allowed by (A.4) and vice versa. This implies that the CIMM instance will be identifiable only if the constraint (A.6) only allows $w_1'=w_1$. The upper and lower bounds on the ratio w_1'/w_2' from (A.6) both equal w_1/w_2 iff $\mu_x^{(1)}=\mu_x^{(2)}=\mu_y^{(1)}=\mu_y^{(2)}=1$ (which would also set $\gamma=1$ in (A.4)). This completes the proof of theorems 1 and 2 for the two component case.

A.2. Necessary condition for the $C>2$ case

The necessary condition from theorem 1 for the $C>2$ case can be seen as a corollary of the same theorem 1 for the $C=2$ case, since a nonparametric CIMM instance with more than two components can be identifiable only if for every pair of components, the two component mixture formed by the pair (after appropriately scaling their weights to add up to 1) is identifiable.

A.3. Sufficient condition for the $C>2$ case

Let Ω_x and Ω_y be the sample spaces of x and y, respectively, and let $\mathbb{P}[\cdots]$ represent the probability of an event. Let us consider a bivariate CIMM instance with $C>2$ components which satisfies condition (37), i.e. the sufficient condition for identifiability according to theorem 2 (which is to be proved here)[13]. Let $w_i, f_x^{(i)}$, $f_y^{(i)}$, $\alpha_x^{(i)}$, and $\alpha_y^{(i)}$ have the same meanings as in the rest of the paper.

From the definition of ess sup, we can see that for all $0<\epsilon<1$, there exist disjoint sets $X_1,\ldots,X_C\subset\Omega_x$ and disjoint sets $Y_1,\ldots,Y_C\subset\Omega_y$ such that[14]

$$\mathbb{P}[x\in X_i]>0\,,\qquad\forall i\in\{1,\ldots,C\}\,, \tag{A.7a}$$

$$\mathbb{P}[y\in Y_i]>0\,,\qquad\forall i\in\{1,\ldots,C\}\,, \tag{A.7b}$$

$$(1-\epsilon)\leqslant\alpha_x^{(i)}(x)\leqslant 1\,,\qquad\forall x\in X_i\,,\forall i\in\{1,\ldots,C\}\,, \tag{A.7c}$$

$$(1-\epsilon)\leqslant\alpha_y^{(i)}(y)\leqslant 1\,,\qquad\forall y\in Y_i\,,\forall i\in\{1,\ldots,C\}\,. \tag{A.7d}$$

From (A.7*c*) and (A.7*d*), we can see that

$$\alpha_x^{(i)}(x)\leqslant\epsilon\,,\qquad\forall x\in X_j\,,\ \forall(i,j)\in\{1,\ldots,C\}^2\ :\ i\neq j\,, \tag{A.8a}$$

$$\alpha_y^{(i)}(y)\leqslant\epsilon\,,\qquad\forall y\in Y_j\,,\ \forall(i,j)\in\{1,\ldots,C\}^2\ :\ i\neq j\,. \tag{A.8b}$$

As ϵ is made arbitrarily small, the region $x\in X_i$ and the region $y\in Y_i$ become arbitrarily close to being populated exclusively by the component i. This induces a block diagonal structure, with the probability $\mathbb{P}_{ij}\equiv\mathbb{P}\left[(x,y)\in X_i\times Y_j\right]$ becoming arbitrarily small if $i\neq j$. More concretely, from (13), we can write

$$\mathbb{P}_{ij}=\mathbb{P}[x\in X_i]\,\mathbb{P}[y\in Y_j]\sum_{k=1}^{C}\frac{E_{x\in X_i}\left[\alpha_x^{(k)}(x)\right]\,E_{y\in Y_j}\left[\alpha_y^{(k)}(y)\right]}{w_k}\,. \tag{A.9}$$

Using (A.7*c*), (A.7*d*), (A.8), and (A.9), we can show that

$$\mathbb{P}_{ij}\leqslant\epsilon\ \mathbb{P}[x\in X_i]\,\mathbb{P}[y\in Y_j]\sum_{k=1}^{C}w_k^{-1}\,,\qquad\forall i\neq j\,. \tag{A.10}$$

[13] The following proof also works for the $C=2$ case.

[14] For notational convenience, the ϵ-dependence of the sets X_i and Y_i is not indicated explicitly.

Similarly, using (A.7*c*), (A.7*d*), and (A.9) we can show that

$$\mathbb{P}_{ii} \geqslant (1-\epsilon)^2 \ \frac{\mathbb{P}[x \in X_i]\ \mathbb{P}[y \in Y_i]}{w_i}, \qquad \forall i. \tag{A.11}$$

Now, let us consider a different CIMM instance with weights w'_i, distributions $f_x^{\prime(i)}$ and $f_y^{\prime(i)}$, and classifiers $\alpha_x^{\prime(i)}$ and $\alpha_y^{\prime(i)}$ which has an observationally equivalent distribution $\mathcal{P}(x,y)$ as the original CIMM instance. We will refer to this as the 'primed CIMM instance'. We will prove that the original CIMM is identifiable by showing that the primed CIMM instance must be equivalent to the original, up to permutations of the component index i.

The key observation is that in the small ϵ limit, no component of primed CIMM instance can have non-vanishing contributions in the region $(x,y) \in X_i \times Y_i$ for more than one i. If some component of the primed instance, say the kth component, has non-vanishing contributions to $\mathbb{P}_{ii}$ and $\mathbb{P}_{jj}$ for $i \neq j$, then the 'off-diagonal probabilities' $\mathbb{P}_{ij}$ and $\mathbb{P}_{ji}$ will also receive non-vanishing contributions (due to conditional independence), which is not allowed by (A.10). To make this argument more carefully, it can be shown, from (A.9), that for all $i,j,k \in \{1,\ldots,C\}$,

$$\begin{aligned}\mathbb{P}_{ij}\mathbb{P}_{ji} \geqslant\ & \mathbb{P}[x \in X_i]\ \mathbb{P}[y \in Y_i]\ \mathbb{P}[x \in X_j]\ \mathbb{P}[y \in Y_j] \\ & \times \frac{E_{x\in X_i}\left[\alpha_x^{\prime(k)}(x)\right] E_{y\in Y_i}\left[\alpha_y^{\prime(k)}(y)\right] E_{x\in X_j}\left[\alpha_x^{\prime(k)}(x)\right] E_{y\in Y_j}\left[\alpha_y^{\prime(k)}(y)\right]}{w_k'^2}.\end{aligned} \tag{A.12}$$

Using (A.10) and (A.12), we can show that for all $i,j,k \in \{1,\ldots,C\}$ with $i \neq j$,

$$\frac{E_{x\in X_i}\left[\alpha_x^{\prime(k)}(x)\right] E_{y\in Y_i}\left[\alpha_y^{\prime(k)}(y)\right]}{w'_k} \ \frac{E_{x\in X_j}\left[\alpha_x^{\prime(k)}(x)\right] E_{y\in Y_j}\left[\alpha_y^{\prime(k)}(y)\right]}{w'_k} \leqslant \left[\epsilon \sum_{k=1}^{C} w_k^{-1}\right]^2. \tag{A.13}$$

This equation captures the constraint that no component k of the primed CIMM instance can have non-vanishing contributions in two different regions $(x,y) \in X_i \times Y_i$ and $(x,y) \in X_j \times Y_j$ with $i \neq j$. On the other hand, each of the 'diagonal regions' must receive a non-vanishing contribution from at least of the components of the primed instance. More concretely, from (A.9) and (A.11), one can see that for all $i \in \{1,\ldots,C\}$, there exists a $k \in \{1,\ldots,C\}$ such that

$$\frac{E_{x\in X_i}\left[\alpha_x^{\prime(k)}(x)\right] E_{y\in Y_i}\left[\alpha_y^{\prime(k)}(y)\right]}{w'_k} \geqslant \frac{(1-\epsilon)^2}{C w_i}. \tag{A.14}$$

From (A.13) and (A.14) and the fact that both the original and the primed CIMM instances have the same number of components, one can see that as ϵ is made arbitrarily small, there exists a permutation σ of the component indices such that $f_x^{(i)}$-s are observationally equivalent to the corresponding $f_x^{\prime\sigma(i)}$-s in the region $x \in \bigcup_{i=1}^{C} X_i$, and similarly $f_y^{(i)}$-s are observationally equivalent to $f_y^{\prime\sigma(i)}$-s in the region $y \in \bigcup_{i=1}^{C} Y_i$.

The equality of the weights w_i and $w'_{\sigma(i)}$ and the equivalence of the distributions $f_x^{(i)}$ and $f_x^{\prime\sigma(i)}$ in the entire sample space Ω_x follows from the fact that both the original and primed CIMM instances have observationally equivalent distribution $\mathcal{P}(x,y)$ in the region $(x,y) \in \Omega_x \times Y_i$—note that Y_i has a non-zero measure for all $\epsilon > 0$. A symmetric argument establishes the equivalence of the distributions $f_y^{(i)}$ and $f_y^{\prime\sigma(i)}$ in the entire sample space Ω_y. This completes the proof of theorem 2 for $C \geqslant 2$ components.

Appendix B. Functional gradient of `neg_ctc_cost`

In this section we will discuss a strategy that can speed-up and improve the training of InClass nets using the `neg_ctc_cost` of (24*b*)

$$\texttt{neg\_ctc\_cost} = -E_{\mathcal{P}^*}\left[\log\left\{\frac{\sum_{i=1}^{C}\left[\prod_{v=1}^{V}\beta_v^{(i)}\left(E_{\mathcal{P}^*}\left[\beta_v^{(i)}\right]\right)^{\frac{1-V}{V}}\right]}{\sum_{i=1}^{C}\left[\prod_{v=1}^{V}\left(E_{\mathcal{P}^*}\left[\beta_v^{(i)}\right]\right)^{1/V}\right]}\right\}\right]. \tag{B.1}$$

Note that there are multiple expectations $E_{\mathcal{P}^*}$ in the expression for the cost function. The outermost expectation is similar to the one in a cost function which can be written as an expectation over a

per-datapoint loss function. For such cost functions (which only have an outermost expectation), one can use stochastic or mini-batch gradient descent for faster or more efficient training of the network. However, the presence of the inner expectations $\varphi_v^{(i)} \equiv E_{\mathcal{P}^*}\left[\beta_v^{(i)}\right]$ in our cost function means that the batch size used in the training should be large enough to estimate the pseudo weights $\varphi_v^{(i)}$ well. In particular, the batch size should be large enough to pick up subtle changes in the value of $\varphi_v^{(i)}$ caused by changes to the network weights $\boldsymbol{\theta}$. The need for large batch sizes will only be exacerbated as the number of components increases.

However, we can overcome this difficulty, and facilitate the use of stochastic gradient descent to optimize the `neg_ctc_cost` as shown below. We will begin by deriving the expression for the functional derivative of the cost function with respect to the NN outputs. For convenience, let is define N and D as

$$N \equiv \sum_{i=1}^{C}\left[\prod_{v=1}^{V}\beta_v^{(i)}\left(E_{\mathcal{P}^*}\left[\beta_v^{(i)}\right]\right)^{\frac{1-V}{V}}\right], \tag{B.2a}$$

$$D \equiv \sum_{i=1}^{C}\left[\prod_{v=1}^{V}\left(E_{\mathcal{P}^*}\left[\beta_v^{(i)}\right]\right)^{1/V}\right]. \tag{B.2b}$$

This lets us write

$$\texttt{neg\_ctc\_cost} = -E_{\mathcal{P}^*}\left[\log\left(\frac{N}{D}\right)\right]. \tag{B.3}$$

Taking the functional derivative with respect to $\beta_u^{(j)}(x'_u)$, one gets

$$\frac{\delta\,\texttt{neg\_ctc\_cost}}{\delta\,\beta_u^{(j)}(x'_u)} = -E_{\mathcal{P}^*}\left[\frac{1}{N}\frac{\delta N}{\delta\,\beta_u^{(j)}(x'_u)}\right] + \frac{1}{D}\frac{\delta D}{\delta\,\beta_u^{(j)}(x'_u)} \tag{B.4}$$

$$\begin{aligned}
&= -\frac{\mathcal{P}_u^*(x'_u)}{\beta_u^{(j)}(x'_u)}\,E_{\mathcal{P}^*}\left[\left.\frac{\prod_{v=1}^{V}\beta_v^{(j)}(x_v)\left(E_{\mathcal{P}^*}\left[\beta_v^{(j)}\right]\right)^{(1-V)/V}}{\sum_{i=1}^{C}\left[\prod_{v=1}^{V}\beta_v^{(i)}(x_v)\left(E_{\mathcal{P}^*}\left[\beta_v^{(i)}\right]\right)^{(1-V)/V}\right]}\right| x_u = x'_u\right] \\
&\quad - \frac{\mathcal{P}_u^*(x'_u)}{E_{\mathcal{P}^*}\left[\beta_u^{(j)}\right]}\,E_{\mathcal{P}^*}\left[\frac{\prod_{v=1}^{V}\beta_v^{(j)}\left(E_{\mathcal{P}^*}\left[\beta_v^{(j)}\right]\right)^{(1-V)/V}}{\sum_{i=1}^{C}\left[\prod_{v=1}^{V}\beta_v^{(i)}\left(E_{\mathcal{P}^*}\left[\beta_v^{(i)}\right]\right)^{(1-V)/V}\right]}\right] \\
&\quad + \frac{\mathcal{P}_u^*(x'_u)}{E_{\mathcal{P}^*}\left[\beta_u^{(j)}\right]}\,\frac{\prod_{v=1}^{V}\left(E_{\mathcal{P}^*}\left[\beta_v^{(j)}\right]\right)^{1/V}}{\sum_{i=1}^{C}\left[\prod_{v=1}^{V}\left(E_{\mathcal{P}^*}\left[\beta_v^{(i)}\right]\right)^{1/V}\right]}.
\end{aligned} \tag{B.5}$$

Using this, we can write the gradient of the cost function with respect to the NN weights $\boldsymbol{\theta}$ as

$$\nabla_{\boldsymbol{\theta}}\,\texttt{neg\_ctc\_cost} = \sum_{j=1}^{C}\sum_{u=1}^{V}\int dx'_u\,\frac{\delta\,\texttt{neg\_ctc\_cost}}{\delta\,\beta_u^{(j)}(x'_u)}\,\nabla_{\boldsymbol{\theta}}\,\beta_u^{(j)}(x'_u) \tag{B.6}$$

$$= -\sum_{j,u}E_{\mathcal{P}^*}\left[\frac{\nabla_{\boldsymbol{\theta}}\,\beta_u^{(j)}}{\beta_u^{(j)}}\,\frac{\prod_{v=1}^{V}\beta_v^{(j)}\left(E_{\mathcal{P}^*}\left[\beta_v^{(j)}\right]\right)^{(1-V)/V}}{\sum_{i=1}^{C}\left[\prod_{v=1}^{V}\beta_v^{(i)}\left(E_{\mathcal{P}^*}\left[\beta_v^{(i)}\right]\right)^{(1-V)/V}\right]}\right]$$

$$-\sum_{j,u}\frac{E_{\mathcal{P}^*}\left[\nabla_{\boldsymbol{\theta}}\,\beta_u^{(j)}\right]}{E_{\mathcal{P}^*}\left[\beta_u^{(j)}\right]}\,E_{\mathcal{P}^*}\left[\frac{\prod_{v=1}^{V}\beta_v^{(j)}\,\left(E_{\mathcal{P}^*}\left[\beta_v^{(j)}\right]\right)^{(1-V)/V}}{\sum_{i=1}^{C}\left[\prod_{v=1}^{V}\beta_v^{(i)}\,\left(E_{\mathcal{P}^*}\left[\beta_v^{(i)}\right]\right)^{(1-V)/V}\right]}\right]$$
$$+\sum_{j,u}\frac{E_{\mathcal{P}^*}\left[\nabla_{\boldsymbol{\theta}}\,\beta_u^{(j)}\right]}{E_{\mathcal{P}^*}\left[\beta_u^{(j)}\right]}\,\frac{\prod_{v=1}^{V}\left(E_{\mathcal{P}^*}\left[\beta_v^{(j)}\right]\right)^{1/V}}{\sum_{i=1}^{C}\left[\prod_{v=1}^{V}\left(E_{\mathcal{P}^*}\left[\beta_v^{(i)}\right]\right)^{1/V}\right]}\,. \tag{B.7}$$

This expression allows us to approximate the gradient of the cost function as

$$\begin{aligned}\nabla_{\boldsymbol{\theta}}\,\texttt{neg\_ctc\_cost} \approx -\sum_{j,u}E_{\mathcal{P}^*}&\left[\frac{\nabla_{\boldsymbol{\theta}}\,\beta_u^{(j)}}{\beta_u^{(j)}}\,\frac{\prod_{v=1}^{V}\beta_v^{(j)}\,\left(\hat{\varphi}_v^{(j)}\right)^{(1-V)/V}}{\sum_{i=1}^{C}\left[\prod_{v=1}^{V}\beta_v^{(i)}\,\left(\hat{\varphi}_v^{(i)}\right)^{(1-V)/V}\right]}\right]\\ &-\sum_{j,u}\frac{E_{\mathcal{P}^*}\left[\nabla_{\boldsymbol{\theta}}\,\beta_u^{(j)}\right]}{\hat{\varphi}_u^{(j)}}\,\times\,\texttt{aux}^{(j)}\\ &+\sum_{j,u}\frac{E_{\mathcal{P}^*}\left[\nabla_{\boldsymbol{\theta}}\,\beta_u^{(j)}\right]}{\hat{\varphi}_u^{(j)}}\,\frac{\prod_{v=1}^{V}\left(\hat{\varphi}_v^{(j)}\right)^{1/V}}{\sum_{i=1}^{C}\left[\prod_{v=1}^{V}\left(\hat{\varphi}_v^{(i)}\right)^{1/V}\right]}\,,\end{aligned} \tag{B.8}$$

where $\hat{\varphi}_v^{(i)}$ represents a moving estimate of $E_{\mathcal{P}^*}\left[\beta_v^{(i)}\right]$ maintained throughout the network training process and $\texttt{aux}^{(j)}$ represents a moving estimate of

$$E_{\mathcal{P}^*}\left[\frac{\prod_{v=1}^{V}\beta_v^{(j)}\,\left(E_{\mathcal{P}^*}\left[\beta_v^{(j)}\right]\right)^{(1-V)/V}}{\sum_{i=1}^{C}\left[\prod_{v=1}^{V}\beta_v^{(i)}\,\left(E_{\mathcal{P}^*}\left[\beta_v^{(i)}\right]\right)^{(1-V)/V}\right]}\right]\,. \tag{B.9}$$

Maintaining the moving estimates $\hat{\varphi}_v^{(i)}$ and $\texttt{aux}^{(j)}$ is comparable to maintaining a discriminator in the training of a Generative Adversarial Network (GAN). The discriminator can be used to evaluate (and improve) the generator using mini-batches of data, instead of evaluating the (gradient of the) statistical distance between the training dataset and the GAN-dataset from scratch at every training step. Likewise, $\hat{\varphi}_v^{(i)}$ and $\texttt{aux}^{(j)}$ faciliate the use of stochastic or mini-batch gradient descent for the `neg_ctc_cost` using (B.8)—all the expectations in that expression are amenable to replacement with stochastic or mini-batch estimates. We note that this strategy was not needed for the studies performed in this paper.

The strategy employed in this section to facilitate the use to stochastic gradient descent for optimizing `neg_ctc_cost` is applicable to a number of cost functions which cannot be written as expectations of per-datapoint loss functions. We will expand on this idea in future publications, and may implement it in future versions of `RainDancesVI`.

Appendix C. Surrogate cost functions

For the cost function `neg_ctc_cost` from (24*b*), we can define a *surrogate* cost function `neg_ctc_cost` as

$$\texttt{unnorm\_neg\_ctc\_cost} = -E_{\mathcal{P}^*}\left[\log\left\{\sum_{i=1}^{C}\left[\prod_{v=1}^{V}\beta_v^{(i)}\,\left(E_{\mathcal{P}^*}\left[\beta_v^{(i)}\right]\right)^{(1-V)/V}\right]\right\}\right]\,. \tag{C.1}$$

This surrogate cost function is an alternative cost function whose minimization will also lead to the minimization of the `neg_ctc_cost`. More concretely, if the true distribution $\mathcal{P}^*$ does correspond to a

CIMM, then the surrogate cost function will be minimized only when the network outputs (pseudo classifiers) $\beta_v^{(i)}$ match the classifiers $\alpha_v^{(i)}$ that correspond to a best fitting CIMM instance. This can be proved as follows: From (24*b*) and (C.1), we have

$$\texttt{neg\_ctc\_cost} = \texttt{neg\_ctc\_cost} - E_{\mathcal{P}^*}\left[\log\left\{\sum_{i=1}^{C}\left[\prod_{v=1}^{V}\left(E_{\mathcal{P}^*}\left[\beta_v^{(i)}\right]\right)^{1/V}\right]\right\}\right] \tag{C.2a}$$

$$\geqslant \texttt{neg\_ctc\_cost} - E_{\mathcal{P}^*}\left[\log\left\{\frac{1}{V}\sum_{i=1}^{C}\sum_{v=1}^{V}\left(E_{\mathcal{P}^*}\left[\beta_v^{(i)}\right]\right)\right\}\right] \tag{C.2b}$$

$$= \texttt{neg\_ctc\_cost}\,. \tag{C.2c}$$

In (C.2*b*), we have used the inequality of arithmetic and geometric means and in step (C.2*c*), we have used the constraint $\sum_{i=1}^{C}\beta_v^{(i)}(x_v) = 1$ satisfied by the NN outputs. Note that setting the pseudo classifiers $\beta_v^{(i)}$ to be equal to the classifiers $\alpha^{(i)}$ corresponding to a best fitting CIMM instance both a) minimizes `neg_ctc_cost`, and b) satisfies the condition for equality in (C.2*b*). This completes the proof that the `unnorm_neg_ctc_cost` is a suggogate cost function for the `neg_ctc_cost` when the data does correspond to some CIMM. The bivariate special case `unnorm_neg_cmi_cost` which is a surrogate cost function for the `neg_cmi_cost` can be explictly written as

$$\texttt{unnorm\_neg\_cmi\_cost} = -E_{\mathcal{P}^*}\left[\log\left\{\sum_{i=1}^{C}\frac{\beta_x^{(i)}\,\beta_y^{(i)}}{\sqrt{E_{\mathcal{P}^*}\left[\beta_x^{(i)}\right]E_{\mathcal{P}^*}\left[\beta_y^{(i)}\right]}}\right\}\right]. \tag{C.3}$$

These surrogate cost functions are also implemented in the `RainDancesVI` package.

ORCID iDs

Konstantin T Matchev https://orcid.org/0000-0003-4182-9096
Prasanth Shyamsundar https://orcid.org/0000-0002-2748-9091

References

[1] McLachlan G J, Lee S X and Rathnayake S I 2019 Finite mixture models *Annu. Rev. Stat. Appl.* **6** 355–78
[2] Teicher H 08 1967 Identifiability of mixtures of product measures *Ann. Math. Stat.* **38** 1300–2
[3] Chauveau D, Hunter D R and Levine M 2015 Semi-parametric estimation for conditional independence multivariate finite mixture models *Stat. Surv.* **9** 1–31
[4] Zhu X and Hunter D R 2016 Theoretical grounding for estimation in conditional independence multivariate finite mixture models *J. Nonparametr. Stat.* **28** 683–701
[5] Hall P, Neeman A, Pakyari R and Elmore R 09 2005 Nonparametric inference in multivariate mixtures *Biometrika* **92** 667–78
[6] Chauveau D and Hoang V T L 2016 Nonparametric mixture models with conditionally independent multivariate component densities *Comput. Stat. Data Anal.* **103** 1–16
[7] Lazarsfeld P F and Henry N W 1968 *Latent Structure Analysis* (Boston: Houghton Mifflin Company)
[8] Andersen E B 1982 Latent Structure Analysis: a survey *Scand. J. Stat.* **9** 1–12
[9] Clogg C C 1995 Latent class models *Handbook of Statistical Modeling for the Social and Behavioral Sciences* ed G Arminger, C C Clogg and M E Sobel (Boston, MA: Springer) ch 6, pp 311–59
[10] Compiani G and Kitamura Y 2016 Using mixtures in econometric models: a brief review and some new results *Econometrics J.* **19** C95–C127
[11] Hinde J, Ingrassia S, Lin T-I and McNicholas P D 2017 Special issue on mixture models *Econometrics Stat.* **3** 89–90
[12] Vermunt J K 2003 Applications of latent class analysis in social science research *Symbolic and Quantitative Approaches to Reasoning with Uncertainty* (*Lecture Notes in Computer Science*) vol 2711, ed T D Nielsen and N L Zhang (Berlin: Springer) pp 22–36
[13] Vermunt J and Magidson J 2004 Latent class analysis *The SAGE Encyclopedia of Social Science Research Methods* ed M S Lewis-Beck, A Bryman and T F Liao (Thousand Oaks, CA: SAGE Publishing) pp 549–53
[14] Porcu M and Giambona F 2017 Introduction to latent class analysis with applications *J. Early Adolesc.* **37** 129–58
[15] Petersen K J, Qualter P and Humphrey N 2019 The application of latent class analysis for investigating population child mental health: a systematic review *Front. Psychol.* **10** 1214
[16] Yu T 2010 An exploratory data analysis method to reveal modular latent structures in high-throughput data *BMC Bioinform.* **11** 440
[17] Nemec J and Nemec A F L 1991 Mixture models for studying stellar populations. I. Univariate mixture models, parameter estimation and the number of discrete population components *Publ. Astron. Soc. Pac.* **103** 95

[18] Bovy J, Myers A D, Hennawi J F, Hogg D W, McMahon R G, Schiminovich D, Sheldon E S, Brinkmann J, Schneider D P and Weaver B A 2012 Photometric redshifts and quasar probabilities from a single, data-driven generative model *Astrophys. J.* **749** 41
[19] Lee K J, Guillemot L, Yue Y L, Kramer M and Champion D J 2012 Application of the Gaussian mixture model in pulsar astronomy—pulsar classification and candidates ranking for the Fermi 2FGL catalogue *Mon. Not. R. Astron. Soc.* **424** 2832–40
[20] Melchior P and Goulding A D 2018 Filling the gaps: Gaussian mixture models from noisy, truncated or incomplete samples *Astron. Comput.* **25** 183–94
[21] Kuhn M and Feigelson E 2017 Mixture models in astronomy *Handbook of Mixture Analysis* ed S Fruhwirth-Schnatter, G Celeux and C P Robert (Boca Raton, FL: CRC Press) ch 19, pp 463–83
[22] Necib L, Lisanti M and Belokurov V 2018 Inferred evidence for dark matter kinematic substructure with SDSS-Gaia (arXiv:1807.02519)
[23] Jones D M and Heavens A F 2019 Gaussian mixture models for blended photometric redshifts *Mon. Not. R. Astron. Soc.* **490** 3966–86
[24] Stepanek M, Franc J and Kus V 2015 Modification of Gaussian mixture models for data classification in high energy physics *J. Phys.: Conf. Ser.* **574** 012150
[25] Cranmer K, Pavez J and Louppe G 2015 Approximating likelihood ratios with calibrated discriminative classifiers (arXiv:1506.02169)
[26] Cranmer K, Pavez J, Louppe G and Brooks W 2016 Experiments using machine learning to approximate likelihood ratios for mixture models *J. Phys.: Conf. Ser.* **762** 012034
[27] Rossi R J 2018 *Mathematical Statistics: An Introduction to Likelihood Based Inference* (New York: Wiley)
[28] Dempster A P, Laird N M and Rubin D B 1977 Maximum likelihood from incomplete data via the EM algorithm *J. R. Stat. Soc.* B **39** 1–38
[29] Berrettini M, Galimberti G, Ranciati S and Murphy T B 2021 Flexible Bayesian modelling of concomitant covariate effects in mixture models (arXiv:2105.12852)
[30] Zhang B 2002 An EM algorithm for a semiparametric finite mixture model *J. Stat. Comput. Simul.* **72** 791–802
[31] Benaglia T, Chauveau D and Hunter D R 2009 An EM-like algorithm for semi- and nonparametric estimation in multivariate mixtures *J. Comput. Graph. Stat.* **18** 505–26
[32] Xiang S, Yao W and Yang G 08 2019 An overview of semiparametric extensions of finite mixture models *Stat. Sci.* **34** 391–404
[33] Hall P and Zhou X-H 2003 Nonparametric estimation of component distributions in a multivariate mixture *Ann. Stat.* **31** 201–24
[34] Benaglia T, Chauveau D, Hunter D and Young D 2009 mixtools: an R package for analyzing mixture models *J. Stat. Softw.* **32** 1–29
[35] Sgouritsa E, Janzing D, Peters J and Schölkopf B 2013 Identifying finite mixtures of nonparametric product distributions and causal inference of confounders *Proc. 29th Conf. on Uncertainty in Artificial Intelligence, UAI'13* (Arlington, VA: AUAI Press) pp 556–75
[36] Levine M, Hunter D R and Chauveau D 2011 Maximum smoothed likelihood for multivariate mixtures *Biometrika* **98** 403–16
[37] Kasahara H and Shimotsu K 2014 Nonparametric identification and estimation of the number of components in multivariate mixtures *J. R. Stat. Soc.* B **76** 97–111
[38] Zheng C and Wu Y 2019 Nonparametric estimation of multivariate mixtures *J. Am. Stat. Assoc.* **115** 1456–71
[39] Yakowitz S J and Spragins J D 02 1968 On the identifiability of finite mixtures *Ann. Math. Stat.* **39** 209–14
[40] Gyllenberg M, Koski T, Reilink E and Verlaan M 1994 Non-uniqueness in probabilistic numerical identification of bacteria *J. Appl. Probab.* **31** 542–8
[41] Elmore R, Hall P and Neeman A 2005 An application of classical invariant theory to identifiability in nonparametric mixtures *Ann. Inst. Fourier* **55** 1–28
[42] Allman E S, Matias C and Rhodes J A 2009 Identifiability of parameters in latent structure models with many observed variables *Ann. Stat.* **37** 3099–132
[43] Kasahara H and Shimotsu K 2009 Nonparametric identification of finite mixture models of dynamic discrete choices *Econometrica* **77** 135–75
[44] Kovtun M, Akushevich I and Yashin A 2014 On identifiability of mixtures of independent distribution laws *ESAIM Probab. Stat.* **18** 207–32
[45] Tahmasebi B, Motahari A and Maddah-Ali M 2018 On the identifiability of finite mixtures of finite product measures (arXiv:1807.05444)
[46] Lloyd S 1982 Least squares quantization in PCM *IEEE Trans. Inf. Theory* **28** 129–37
[47] Ester M, Kriegel H-P, Sander J and Xu X 1996 A density-based algorithm for discovering clusters in large spatial databases with noise *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining, KDD'96* (AAAI Press) pp 226–31
[48] Dhillon I S, Mallela S and Modha D S 2003 Information-theoretic co-clustering *Proc. 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, KDD '03* (New York: Association for Computing Machinery) pp 89–98
[49] Friedman N, Mosenzon O, Slonim N and Tishby N 2001 Multivariate information bottleneck *Proc. 17th Conf. in Uncertainty in Artificial Intelligence, UAI '01* (San Francisco, CA: Morgan Kaufmann Publishers Inc.) pp 152–61
[50] Ji X, Henriques J F and Vedaldi A 2019 Invariant information clustering for unsupervised image classification and segmentation *Proc. IEEE Int. Conf. on Computer Vision* pp 9865–74
[51] Quadrianto N, Smola A J, Caetano T S and Le Q V 2008 Estimating labels from label proportions *Proc. 25th Int. Conf. on Machine Learning, ICML '08* (New York: Association for Computing Machinery) pp 776–83
[52] Patrini G, Nock R, Rivera P and Caetano T 2014 (Almost) no label no cry *Advances in Neural Information Processing Systems* vol 27, ed Z Ghahramani, M Welling, C Cortes, N D Lawrence and K Q Weinberger (Curran Associates, Inc.) pp 190–8
[53] Yu F, Kumar S, Jebara T and Chang S 2014 On learning with label proportions (arXiv:1402.5902)
[54] Metodiev E M, Nachman B and Thaler J 2017 Classification without labels: learning from mixed samples in high energy physics *J. High Energy Phys.* **10** 174
[55] Hyvärinen A, Karhunen J and Oja E 2001 *Independent Component Analysis* (*Adaptive and Learning Systems for Signal Processing, Communications and Control Series*) (New York: Wiley)
[56] Gershman S J and Blei D M 2012 A tutorial on Bayesian nonparametric models *J. Math. Psychol.* **56** 1–12
[57] Pivk M and Le Diberder F R 2005 SPlot: a statistical tool to unfold data distributions *Nucl. Instrum. Methods Phys. Res.* A **555** 356–69
[58] Goodfellow I, Bengio Y and Courville A 2016 *Deep Learning* (Cambridge, MA: MIT Press)
[59] Watanabe S 1960 Information theoretical analysis of multivariate correlation *IBM J. Res. Dev.* **4** 66–82
[60] Garner W R 1962 *Uncertainty and Structure as Psychological Concepts* (New York: Wiley)

[61] Abadi M *et al* 2016 TensorFlow: a system for large-scale machine learning *12th USENIX Symp. on Operating Systems Design and Implementation (OSDI 16)* pp 265–83
[62] The `RainDancesVI` package is (available at https://prasanthcakewalk.gitlab.io/raindancesvi)
[63] Shyamsundar P and Matchev K T 2020 InClass nets code and data repository (directory: arXiv_2009.00131) gitlab (available at: https://gitlab.com/prasanthcakewalk/code-and-data-availability/)
[64] Nair V and Hinton G E 2010 Rectified linear units improve restricted boltzmann machines *Proc. 27th Int. Conf. on Machine Learning, ICML'10* (Madison, WI: Omnipress) pp 807–14
[65] Kingma D and Ba J 2014 Adam: a method for stochastic optimization *Int. Conf. on Learning Representations* (arXiv:1412.6980)
[66] Rosenblatt M 1956 Remarks on some nonparametric estimates of a density function *Ann. Math. Stat.* **27** 832–7
[67] Parzen E 1962 On estimation of a probability density function and mode *Ann. Math. Stat.* **33** 1065–76
[68] LeCun Y, Cortes C and Burges C 2010 MNIST handwritten digit database *ATT Labs* vol 2 (available at: http://yann.lecun.com/exdb/mnist)
[69] Matchev K T and Shyamsundar P 2020 OASIS: optimal analysis-specific importance sampling for event generation *SciPost Phys.* **10** 34
[70] Rasmussen C E and Williams C K I 2005 *Gaussian Processes for Machine Learning* (*Adaptive Computation and Machine Learning*) (Cambridge, MA: MIT Press)
[71] Neal R M 1996 Priors for infinite networks *Bayesian Learning for Neural Networks* (*Lecture Notes in Statistics*) ed R M Neal (New York: Springer) pp 29–53
[72] Lee J, Bahri Y, Novak R, Schoenholz S, Pennington J and Sohl-Dickstein J 2018 Deep neural networks as Gaussian processes *Int. Conf. on Learning Representations* (arXiv:1711.00165)
[73] Matthews A G D G, Hron J, Rowland M, Turner R E and Ghahramani Z 2018 Gaussian process behaviour in wide deep neural networks *Int. Conf. on Learning Representations*
[74] Kwon C and Mbakop E 2021 Estimation of the number of components of nonparametric multivariate finite mixture models *Ann. Stat.* **49** 2178–205
[75] Liu Q, Yu L, Luo L, Dou Q, Heng P A and Heng P A 2020 Semi-supervised medical image classification with relation-driven self-ensembling model *IEEE Trans. Med. Imaging* **39** 3429–40