CD-IMM: The Benefits of Domain-based Mixture Models in Bayesian Continual Learning

Daniele Castellana
University of Florence
daniele.castellana@unifi.it

Antonio Carta
University of Pisa
antonio.carta@unipi.it

Davide Bacciu
University of Pisa
davide.bacciu@unipi.it

Abstract

Real-world streams of data are characterised by the continuous occurrence of new and old classes, possibly on novel domains. Bayesian non-parametric mixture models provide a natural solution for continual learning due to their ability to create new components on the fly when new data are observed. However, popular class-based and time-based mixtures are often tested on simplified streams (e.g. class-incremental), where shortcuts can be exploited to infer drifts. We hypothesise that *domain-based mixtures are more effective on natural streams*. Our proposed method, the CD-IMM, exemplifies this approach by learning an infinite mixture of domains for each class. We experiment on a natural scenario with a mix of class repetitions and novel domains to validate our hypothesis. The experimental results confirm our hypothesis and we find that CD-IMM beats state-of-the-art bayesian continual learning methods.

1 Introduction

Continual learning (CL) is the ability to learn from a non-stationary stream of data. CL is fundamental in many real-life systems that are subject to concept drift, and whenever new data is collected over time [24]. The main challenge of continual learning is the stability-plasticity tradeoff, that is the tradeoff between the stability of the old knowledge and the plasticity necessary to learn from new data [14].

Recent results in the literature suggest that generative models may be more robust than discriminative models [34, 23, 16]. In this paper, we argue that one of the main benefits of generative models is their ability to factorize the data distribution as a mixture of smaller distributions. Mixtures allow to protect old components unrelated to new data from changes, while inferred probability of new data can be used to detect drifts and create new components. However, popular solutions in the literature often exploit class labels [34, 16] or assume strong time coherence [23] to detect drifts (Section 2.4), simple mechanisms that may fail in most realistic settings. For example, in an object detection task, it is hard to assume that an object (or its appearance, such as the color) will never appear again (as in class-incremental or domain-incremental scenario, see Figure 1,2).

In this paper, we hypothesize that existing class-based or time-based generative methods will fail in a simple setting where new domains are discovered over time and classes are revisited (**H1-H3** in Section 3.2).

We propose the Class-Domain Infinite Mixture Model (CD-IMM), a new domain-based generative model which we expect to be more robust to class repetitions and novel domains (**H4**). CD-IMM lies on the Dirichlet Process Mixture Model (DPMM), a non-parametric model that adapts over time to the complexity of the data by adding more clusters when necessary. The method is general and it can work in online [27], task-free [2], or even unsupervised settings [30]. Furthermore, we expect that the clusters found by the CD-IMM will have a better correspondence with the natural clusters in the data distributions (**H4.1**).

We will test our assumptions on three different benchmarks. First, we propose Incremental Moons benchmark, a toy domain-incremental scenario based on the popular Moons dataset that we designed to showcase the advantages of Bayesian non-parametric classifiers and their robustness to domain drifts (**H1**). Then, we will use two different image classification benchmarks: Alphabet-Omniglot, where the class target for each character is its alphabet, and CIFAR100-Superclasses. The CD-IMM and the baseline methods will be tested on a simple scenario with class repetitions (**H1-H3**). Both datasets provide natural clusters (characters for Omniglot, classes for CIFAR100) that will be used to verify whether the clusters learned by the CD-IMM correspond to the natural clusters (**H4.1**).

Proceedings of the 1st ContinualAI Unconference, 2023, Virtual. PMLR Volume 249.

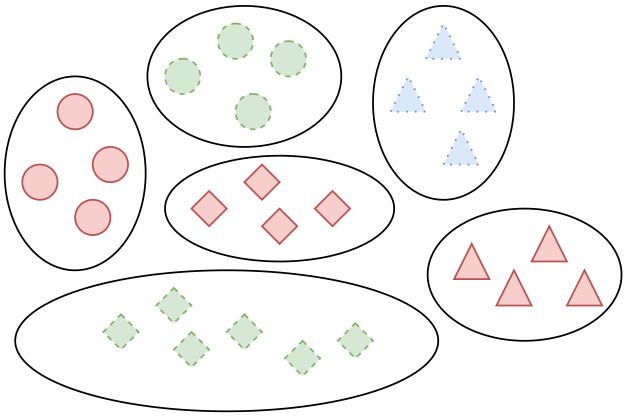


Figure 1: In real-world data, different classes (colors) may be unbalanced and structured into subgroups (shapes). We hypothesize that learning the latent structure of the probability distribution is helpful for continual learning.

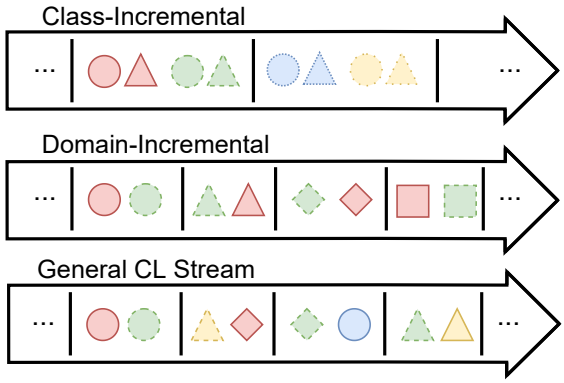


Figure 2: Bayesian CL methods exploit the structure of toy benchmarks (first two arrows), such as the lack of repetitions of classes and domains, to implicitly detect distribution drifts. We hypothesize that these methods will fail in real-world settings (last arrow) where new and old classes (or domains) are continuously encountered.

The experimental results confirm our hypothesis, confirming that the CD-IMM is a better solution for natural continual learning streams with class and domain repetitions. The source code to reproduce the experiments is publicly available[1].

2 Background and Related Works

2.1 Generative Models

Let us consider a learning task where we would like to learn a distribution $f(\boldsymbol{s})$ from a set of samples $\mathcal{D} = \{\boldsymbol{s}_1, \dots, \boldsymbol{s}_N\}$ which are independently drawn from an unknown true distribution $p(\boldsymbol{s})$. The goal of generative models is to learn an approximated distribution $f(\boldsymbol{s}, \boldsymbol{\theta})$ of $p(\boldsymbol{s})$, where $\boldsymbol{\theta}$ are the model parameters which are adapted during the training.

For the purpose of our work, we focus on a specific type of generative model: the mixture model.

Finite Mixture Models assume that the data distribution can be decomposed into a set of simpler distributions called components [28]:

$$f(\boldsymbol{s}, \boldsymbol{\theta}) = \sum_{z=1}^{K} p(\boldsymbol{s} \mid z, \boldsymbol{\theta}_z) p(z \mid \boldsymbol{\beta}). \tag{1}$$

Each component $p(\boldsymbol{s} \mid z, \boldsymbol{\theta}_z)$ has a set of parameters $\boldsymbol{\theta}_z$, while the prior distribution $p(z)$ is parameterized by the vector $\boldsymbol{\beta}$; both $\boldsymbol{\beta}$ and $\boldsymbol{\theta}_z$ are learned from data by using the Expectation-Maximisation (EM) procedure [10].

[1] https://github.com/AntonioCarta/continual_cd_imm

The generative process for the sample $\boldsymbol{s}_i$ can be sketched as follows:

$$z_i \mid \boldsymbol{\beta} \sim \text{Cat}(\boldsymbol{\beta}), \qquad \boldsymbol{s}_i \mid z_i, (\boldsymbol{\theta}_c)_{c=1}^{K} \sim F(\boldsymbol{\theta}_{z_i}). \tag{2}$$

At first, the random variable z_i is sampled using the categorical distribution $p(z) = \text{Cat}(\boldsymbol{\beta})$. The value z_i indicates which component we should use to obtain the data $\boldsymbol{s}_i$. Hence, $\boldsymbol{s}_i$ is generated by sampling the distribution $p(\boldsymbol{s} \mid z_i, \boldsymbol{\theta}_{z_i}) = F(\boldsymbol{\theta}_{z_i})$; F is the family of the component distributions (e.g., a Gaussian) and $\boldsymbol{\theta}$ are its parameters (e.g., mean and variance for the Gaussian).

A common drawback in mixture models is the choice of the number of components K. In fact, since the true distribution is unknown, it is hard to determine in how many components it can be decomposed

Infinite Mixture Models overcome this limitation by allowing to adapt the number of components directly on the observed data. They are also known as Dirichlet Process Mixture Models (DPMM) [3] since they are based on Dirichlet Process (DP) [12].

For our purpose, it is convenient to define the DPMM generative process using the "stick-breaking" construction [32]:

$$\begin{aligned} \boldsymbol{\beta} \mid \alpha \sim \text{Stick}(\alpha) \quad & z_i \mid \boldsymbol{\beta} \sim \text{Cat}(\boldsymbol{\beta}) \\ \boldsymbol{\theta} \mid \boldsymbol{H} \sim \boldsymbol{H}, \quad & \boldsymbol{s}_i \mid z_i, (\boldsymbol{\theta}_c)_{c=1}^{\infty} \sim F(\boldsymbol{\theta}_{z_i}), \end{aligned} \tag{3}$$

where $\text{Stick}(\alpha)$ and $\boldsymbol{H}$ are the prior of $p(z)$ and $p(\boldsymbol{s} \mid z, \boldsymbol{\theta}_z)$, respectively. Thanks to the prior specification, we can sample new components on the fly during the training, making the model infinite (i.e. $z_i \in [1, \infty]$). The generative process of each sample is equal to the finite mixture

case: at first, a component z_i is selected; then, the selected component is used to sample the data.

Due to the infinite capacity of the model, the posterior becomes intractable and the training cannot be performed with the EM algorithm anymore. Typically, approximated strategies are employed such as sampling [29] and variational inference [5]. It is worth highlighting that during the training the number of components K is always finite.

2.2 Continual Learning

A continual learning stream is a sequence of datasets $\mathcal{D}_1, \dots, \mathcal{D}_T$, where $\mathcal{D}_t = \{\boldsymbol{s}_i \mid \boldsymbol{s}_i \sim p_t(\boldsymbol{s})\}$ and $p_t(\boldsymbol{s})$ may change over time. While our method also works in unsupervised settings, we focus on supervised problems where $\boldsymbol{s} = \langle \boldsymbol{x}, y \rangle$, i.e. both the input $\boldsymbol{x} \in \mathbb{R}^D$, and the class $y \in [1, C]$ are observed. Most methods in deep continual learning assume *virtual drift* [15], which means that the underlying data distribution, which we call the *real distribution* of samples $p(\boldsymbol{x}, y)$, exists and is constant over time. Therefore, at each step, only a subset of this distribution is available for training. For example, in a *class-incremental* setting, we have $p_t(\boldsymbol{x}, y) = p(\boldsymbol{x}, y \mid y \in \mathcal{Y}_t)$, where $\mathcal{Y}_t$ is the set of classes visible at time t. Instead, in a *domain-incremental* setting $p_t(\boldsymbol{x}, y) = p(\boldsymbol{x}, y \mid z \in \mathcal{Z}_t)$, where $\mathcal{Z}_t$ is the set of subdomains visible at time t. Note that while class labels are often visible during training, the domain z is a hidden variable.

2.3 Continual Mixture Models

In the continual learning setting, the objective of generative models is to learn a parametric approximation $f(\boldsymbol{x}, y \mid \boldsymbol{\theta})$ of the joint distribution $p(\boldsymbol{x}, y)$ from a stream of datasets $\mathcal{D}_1, \dots, \mathcal{D}_T$ without storing old data. As stated before, the data $\mathcal{D}_t$ observed at time t is obtained from a portion of the whole joint distribution, i.e. $\mathcal{D}_t \sim p_t(\boldsymbol{x}, y)$.

Time-based Mixture. Since the joint distribution is discovered one portion at a time, it could be reasonable to partition the parametric approximation of the generative model. To this end, mixture models can be leveraged:

$$f(\boldsymbol{x}, y \mid \boldsymbol{\theta}) = \sum_t f_t(\boldsymbol{x}, y \mid \boldsymbol{\theta}_t) p(t), \tag{4}$$

where $f_t(\boldsymbol{x}, y \mid \boldsymbol{\theta}_t)$ is the parametric approximation learned using the data $\mathcal{D}_t$. The variable t represents the partition we are considering: when the distribution shift is known at training time, the variable t is visible; otherwise, the value of t is hidden.

The main advantage of this formulation is that we decompose the learning problem into T independent sub-tasks: once we have learned the approximation f_t for the partition $p_t(\boldsymbol{x}, y)$, we do not have to change it anymore (i.e. f_t is learned only on D_t). However, such a decomposition can be difficult to learn since, in general, each portion $p_t(\boldsymbol{x}, y)$ can be as complex as the whole distribution $p(\boldsymbol{x}, y)$.

Class-based Mixture. Another approach is to decompose the approximation f according to the class label of the samples:

$$f(\boldsymbol{x}, y \mid \boldsymbol{\theta}) = \sum_y f_y(\boldsymbol{x}, y \mid \boldsymbol{\theta}_y) p(y), \tag{5}$$

where $f_c(\boldsymbol{x}, c \mid \theta_c)$ is the parametric approximation of the distribution of samples with class c, i.e. $p(\boldsymbol{x}, y = c)$.

While this decomposition is reasonable from the task point of view, learning each approximation f_c is difficult in the continual setting. In fact, to learn f_c we need all the samples with class c, i.e. the set $\{(\boldsymbol{x}, y) \mid y = c\}$. However, the elements in this set can be scattered into $D_1, \dots, D_T$.

Domain-based Mixture. The last approach decomposes the approximation f according to the domain of the samples:

$$f(\boldsymbol{x}, y \mid \boldsymbol{\theta}) = \sum_z f_z(\boldsymbol{x}, y \mid \boldsymbol{\theta}_z) p(z), \tag{6}$$

where $f_z(\boldsymbol{x}, z \mid \theta_z)$ is the parametric approximation of the distribution of samples with domain z.

While class and task information is usually known (if we focus on supervised tasks with visible task boundaries), the domain of a sample is usually unknown. This worsens the training procedure of domain-based mixture models since they should also estimate the sample domain. We believe that this is the main reason why, as far as we know, such mixture models have been not used in the literature.

2.4 Assumptions and Limitations of Generative Models for CL in the Literature

In a realistic setting, we do not expect to know when or how many times each domain occurs. In general, we expect to see both new domains and classes over time, possibly with repetitions. While this setup seems natural, methods in the literature make some restrictive assumptions about the types of drifts that are allowed.

Simple input distribution: some methods assume that there exists a pre-trained feature extractor that can be frozen and returns linearly separable features. For example, Deep SLDA [16] uses this assumption to model each class as a single Gaussian distribution.

Knowledge about task boundaries : more sophisticated methods either assume that drifts are known or that they are easily predictable. Many class-incremental methods, such as Ven, Li, and Tolias [34], use the presence of new class labels to determine drifts. This

assumes that all the examples of a particular class are observed at the same time.

No repetitions: domains and classes are never repeated during training. More formally, many methods assume $\mathcal{Y}_t \cap \mathcal{Y}_{t'} = \emptyset, \forall t \neq t'$ and $\mathcal{Z}_t \cap \mathcal{Z}_{t'} = \emptyset, \forall t \neq t'$. Many methods that freeze previous components make this assumption [34, 31].

Balancing: data is balanced and uniformly complex. This assumption is used by architectural methods that create separate components for each class or domain [34, 31].

These assumptions are difficult to satisfy with real-world data. Furthermore, methods exploiting these assumptions either fail to learn incrementally or become very inefficient.

We focus on three methods, which we briefly summarize below, to show the general strategies adopted by generative methods for continual learning scenarios.

Simple input distribution: Deep SLDA [16] is a class-based finite mixture model and it assumes that each class can be modelled as a Gaussian distribution learned by a linear discriminant analysis (LDA) classifier. Since the raw input space is not Gaussian in most nontrivial applications, Deep SLDA uses a frozen pre-trained feature extractor. Still, the Gaussianity is a strong assumption if we consider that the feature extractor is frozen and it was never trained on the data from the real distribution $p(\boldsymbol{x}, y)$. The advantage of this approach is that it doesn't suffer from forgetting since the online LDA algorithm is equivalent to offline training.

Class-based Mixture: Class-VAE. Ven, Li, and Tolias [34] proposes an approach, which we call Class-VAE in this paper, based on a class-based finite mixture model where each component is a Variational Auto-Encoder (VAE). In principle, the VAE can model any complex distribution, therefore it does not need to model explicitly each subdomain. A disadvantage is that training VAEs incrementally is still an unsolved problem [25]. Class-VAE avoids this limitation by restricting to a pure class-incremental setting without repetitions. The main limitation of the class-VAE is that it cannot be updated if new data for an old class becomes available.

Time-based Mixture: CN-DPM. Lee et al. [23] proposes a time-based infinite mixture model which trains a VAE and a classifier for each task. The VAE is trained to approximate the $p_t(\boldsymbol{x})$ of the true distribution, while the classifier models $p(y|\boldsymbol{x})$. A new VAE is initialized and trained whenever a new probability drift is automatically detected by computing the probability of the new data given the current model. In practice, CN-DPM has been tested only on class or domain incremental tasks, where drift detection is almost trivial. We hypothesize (**H3**) that CN-DPM will either create too many VAEs or fail to recognize drifts in a scenario with repetitions and multiple domains occurring at the same time.

2.5 Related Work

Continual Learning Methods: Bayesian methods provide a natural solution for continual learning. Popular methods such as EWC can be interpreted as Bayesian methods that exploit an approximation of the posterior to mitigate forgetting [19]. Building on the same intuition, IMM [22] merges the new and old model using the first two moments of the posterior distribution, approximated by a Gaussian. The idea of factorizing the model, possibly freezing old components, is exploited by architectural methods [31], which can also use the same components for task inference [1, 7] to remove the need for task labels. Recently, it was shown that self-supervised objectives are more robust to forgetting than supervised objectives [8, 13]. We expect Bayesian mixture models to behave similarly due to their natural ability to recognize domains, tasks, or classes.

Fair Evaluation and Realistic Benchmarks: Farquhar and Gal [11] show how seemingly minor details in the evaluation can affect the performance of CL models and argue for a fairer evaluation. In this paper, we follow the idea that a fair evaluation should be based on realistic assumptions on the stream distribution, such as dropping the constraints of no class repetitions [9, 17, 6]. It is often argued that class-incremental [33] scenarios are more difficult than domain-incremental ones. In this paper, we argue that class-incremental setting, as popularly used in the literature (no repetitions), trivializes many CL challenges such as drift detection, and allows for simple solutions such as freezing that do not generalize to more complex streams.

3 Hypotheses and Proposed Method

3.1 Our proposal: Class-Domain Infinite Mixture Model (CD-IMM)

The main novelty of our proposal is to explicitly consider the domains in the data generation process. Usually, different classes have different domains (Figure 1). Thus, we assume that each class is composed of subgroups (i.e. domains), which we model via the discrete latent variable $z_y \in [1, K_y]$. We use the subscript y to emphasise that different classes have different domains; K_y indicates the number of domains in the class y. We model the joint distribution as:

$$p(\boldsymbol{x}, y) = \sum_{z_y} p(\boldsymbol{x} \mid z_y, \boldsymbol{\theta}_{z_y}) p(z_y \mid y, \boldsymbol{\beta}_y) p(y), \quad (7)$$

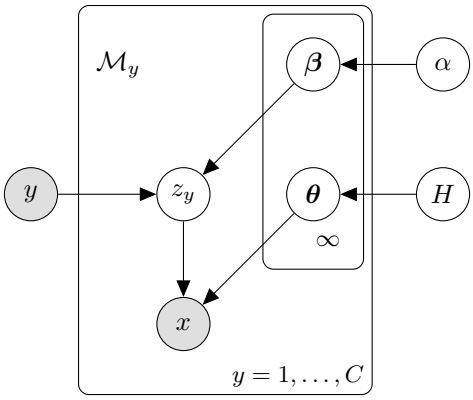


Figure 3: Graphical model of our proposal.

where $\boldsymbol{x} \in \mathbb{R}^D$ can be the input data or features extracted from a frozen backbone.

The above decomposition is obtained by assuming that the data $\boldsymbol{x}$ is independent of the class y given that we know the domain z_y. This conditional independence derives from our assumption that each class has different domains. Thus, if we know the domain z_y we know also the class y. It is worth highlighting that this assumption is not a limitation since the domains are arbitrary: we can always split a shared domain between two classes in two class-specified domains.

The generative model we propose in Equation 7 can be interpreted as a two-level mixture model. The first level is on the class labels: for each class y, we define a model $\mathcal{M}_y$ which is responsible for the generation of the data with label y. The second level is on the domains: each model $\mathcal{M}_y$ is again a mixture model which has a component for each domain $z_y \in [1, K_y]$. While the number of classes C can be assumed to be known, the number of domains K_y is usually unknown. Thus, we define each M_y as a Dirichlet Process Mixture Model: for each class, the number of domains K_y is theoretically infinite. The variables $\{\boldsymbol{\beta}, \boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_\infty\}$ are the parameters of $\mathcal{M}_y$ where we removed the subscript y to ease the notation. In Figure 3 we graphically represent our proposal.

Definition of $\mathcal{M}_y$. Each M_y is modelled as a Gaussian DPMM [3]. Thanks to the DPMM, each M_y can have an infinite number of components (i.e. domains). Each component k is a multivariate Gaussian distribution with parameters $\boldsymbol{\theta_k} = \{\mu_k, \Sigma_k\}$, where $\mu_k \in \mathbb{R}^D$ is the mean and $\Sigma_k \in \mathbb{R}^{D \times D}$ is the covariance matrix. If needed, we can share the same variance across all the components obtaining a *tied* model as done in Deep-SLDA [16]. Also, we can constrain the covariance matrix to be diagonal, obtaining an isotropic multivariate Gaussian distribution. Due to the Bayesian fashion of DPMM, we must always define the prior H of the parameters $\boldsymbol{\theta_k} = \{\mu_k, \Sigma_k\}$. To ease the computation, we define H as a Gaussian-Inverse-Wishart distribution since it is the conjugate prior of $\boldsymbol{\theta_k}$. To be more precise, the prior is factorised as $H(\boldsymbol{\theta_k}) = p(\boldsymbol{\mu}_k \mid \Sigma_k)p(\Sigma_k)$ where: $p(\Sigma_k) = \mathcal{W}^{-1}(\boldsymbol{\Psi}, n_0)$ is the inverse Wishart distribution, and $p(\boldsymbol{\mu}_k \mid \Sigma_k) = \mathcal{N}(\boldsymbol{\mu_0}, \Sigma_k)$ is a multivariate Gaussian distribution with mean $\boldsymbol{\mu}_0$ and covariance Σ_k. Usually, $\boldsymbol{\mu}_0$ is the zero vector; however, this can lead to poor results (especially in the high-dimensional case) since the input data can be far from all the components that have means near zero. To overcome this issue, we can initialise the mean of each component using the kmeans++ algorithm [4] using the first B elements of the stream.

It is worth highlighting that our proposal can reduce to SLDA if M_y is defined as a single Gaussian (i.e. each class has only one domain).

Learning Procedure. Since we observe the class labels, each model $\mathcal{M}_y$ is trained separately. Let us consider a new training sample $\boldsymbol{s}_i = (\boldsymbol{x}_i, y_i)$, the input $\boldsymbol{x}_i$ is considered only to train the model $\mathcal{M}_{y_i}$.

The training of M_y is based on the computation of the posterior $p(\boldsymbol{z}, \boldsymbol{\beta}, \boldsymbol{\theta})$, which is intractable due to the infinite number of clusters. We rely on a variational truncated approximation which defines a maximum number of components. While this might seem the same as a finite model, it is not [5]. The training can also be performed online (i.e. one update for each sample) by applying the Stochastic Variational Inference (SVI) framework [18].

3.2 Our Hypothesis

Our hypothesis is that domain-based mixture models are a better solution for more realistic streams with class and domain repetitions. The following hypothesis state how we expect class-based (**H1-H2**) and time-based method (**H3**) methods to fail, while domain-based mixture models learn the natural domains (**H4**).

H1 - Simple class-based mixture models (Deep SLDA) fail to model complex multi-domain distributions. Simple models such as SLDA do not work when data is not Gaussian.

H2 - Complex class-based mixture models (Class-VAE) fail in settings with class repetitions. Training the Class-VAE on a new domain for some old class will result in catastrophic forgetting.

H3 - Time-based mixture models (CN-DPM) do not account for multiple domains appearing at the same time. CN-DPM is unable to recognize small domain shifts or multiple domains appearing at the same time. (*H3.1*) Since the CN-DPM was tested on simple class-incremental settings, we expect CN-DPM drift detection to heavily rely on different class labels.

H4 - Domain-based mixture models (CD-IMM) are able to learn on general streams with new classes and domains, even with repetitions CD-IMM learns domains, independently from when they occur. (H4.1) The clusters discovered by our proposal match some intrinsic properties of the data.

4 Experimental Protocol

We experiment with three benchmarks in different configurations: Incremental Moons, Omniglot-Alphabets, CIFAR100-Superclasses. The chosen baselines are Deep SLDA, Class VAE, and CN-DPM, which provide representative methods for the class-based and time-based mixture models. We will test the batch setting, which allows for multiple epochs on each batch, and the online setting with a single pass on the data and without task boundaries.

Domain-Incremental Moons. In order to provide an intuitive visualization of the CD-IMM, we propose Incremental Moons, a toy stream which is an incremental version of the Moons dataset[2].

Each moon is shaped as an arc of a circle plus some small Gaussian noise. Each moon is a different class, and the stream provides different sections of the arc for each moon, which are observed in a domain-incremental fashion. As a result, clusters do not overlap each other, and they are well separated. However, they are close enough to each other that it is not possible to approximate each cluster with a single Gaussian emission.

Class-Incremental with Repetitions. To experiment with more realistic streams, we follow a setup similar to [17]. We split each class by domain and we randomly shuffle all the domains. Then, we group domains together to form a batch of data. As a result, new domains for already known classes can occur over time. To split classes into well-defined natural domains, we use datasets that provide coarse and fine labels. We use CIFAR100 [20] by using the 100 classes as (latent) domains and the 10 superclasses as classes, which we call CIFAR100-Superclasses. Similarly, we use Omniglot [21] with the alphabet as the target class and the character as the domain, which we call Omniglot-Alphabet. We will group class-domains randomly to obtain 10 drifts.

Experimental Setup. Deep SLDA, Class VAE, and CD-IMM need a fixed backbone as a feature extractor. We will use raw features for Incremental Moons, a feature extractor pre-trained on CIFAR10 for CIFAR100 (as done in [34]), and a random MLP on Omniglot. If the random feature extractor for Omniglot is not powerful enough, causmaking all the three methods to fail, we plan to document this failure and use a subset of the data to pre-train a feature extractor. The remaining hyperparameters (α, μ_0, Σ_0 for the CL-IMM) will be found via hyperparameter search evaluated on a separate validation stream for all the methods.

Metrics, Evaluation, Reproducibility. We will use the *accuracy over the whole target distribution*, estimated on a separate test set, as the main evaluation metric. Given A_t, the accuracy of the model at time t on the entire test set, we will show the final accuracy A_T, average accuracy over time $\sum_{i=1}^{T} A_i$, and learning curves (**H1-H4**). We will also show a *confusion matrix* split by subdomain, which will allow us to check if time-based and class-based models are biased in the expected ways (**H2-H3**). For the CD-IMM, we will use a *domain confusion matrix*, where DCM_i^j is the percentage of examples of data-domain i associated with the model-domain j, to check how the natural domains fit within the model's domains (**H4**). Each method will be run 5 times to compute the mean and standard deviations for the metrics. We will release the source code for our experiments using Avalanche [26] to ensure the reproducibility of our results.

5 Changes to Protocol

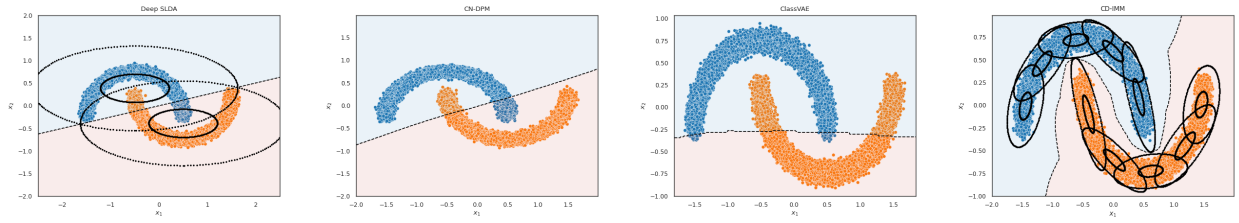
During the experimental evaluation of the methods we decided to do some small modifications of the protocol, which are documented below, along with their motivations.

In the original experimental protocol, we proposed to show learning curves and a *confusion matrix* split by subdomain. In the final version of the paper we decided to show the accuracy of the final model for domains seen at each timestep t separately (Figure 5). We found this figure more informative because it shows a clear picture of the stability-plasticity tradeoff of the models. On the other hand, learning curves are not particularly meaningful for the models we are studying since some of them don't use backpropagation or trained separate components for each class.

As discussed in the original plan, we experimented with a random feature extractor for Omniglot. However, none of the methods were able to solve the task with this configuration. Therefore, we trained the backbone to classify characters. Qualitative results with tSNE showed that, even when training on characters, the resulting embedding space was highly overlapping among classes, which means that this is still a non-trivial task to solve. As a result, Omniglot is the most complex benchmark in this paper, as highlighted by the failure of the baseline models.

Finally, initially we planned to train the ClassVAE in two different ways: fitting one VAE per class or one VAE per domain. However, due to the high computational cost (Figure 6) we were unable to train the second options, which

[2] an implementation of the Moons dataset can be found at `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_moons.html`



(a) Deep SLDA (b) CN-DPM (c) ClassVAE (d) CD-IMM

Figure 4: Decision boundaries on Incremental Moons for all the methods. Plot for Deep SLDA and CD-IMM also show the learned gaussians.

would have been more computationally intensive than the class-based one, which makes it unlikely to be applicable in a realistic continual learning setting. It must be pointed out that Figure 6 only shows the training cost of a single run. However, we found the ClassVAE to be quite sensitive to hyperparameters, and it required the largest grid search among all the methods under study (54 configurations), which made it even more expensive to train for our experiments.

6 Experiments

6.1 Domain-Incremental Moons

In this experiment, we expected the following results:

- **H1** SLDA will fit each moon with a single Gaussian distribution, resulting in a low accuracy.
- **H2** Class-VAE will remember only the final domain (arc section) for each class due to catastrophic forgetting [25].
- **H3** There are two possible failure modes for CN-DPM: (1) it may not recognize new domains, resulting in catastrophic forgetting as the Class-VAE, and (2) it may not be able to learn the VAEs if too many drifts are detected, resulting in underfitted components.

The average accuracy for each method is shown in Table 1. All the results confirm our hypothesis. CD-IMM fits the two classes perfectly, while all the other methods have a lower average accuracy. The issue with the baseline methods can be easily seen in Figure 4, which shows the Gaussians learned by Deep SLDA and the decision boundaries of CN-DPM and ClassVAE. First, we notice that all the methods make all the errors roughly in the same area. However, the mistakes are a result of different failures. Deep SLDA finds the optimal fit of the two gaussians. However, even the optimal fit cannot discriminate between the region where the two arcs are close to each other, which is the area where the errors are located. Instead CN-DPM and ClassVAE could fit arbitrary distributions. However, due to forgetting they are unable to learn the classes distribution incrementally.

Overall, we find that the experiments on Incremental Moons confirm our hypothesis.

6.2 Class-Incremental with Repetition

Expected results: We expect the same failures detailed in Section 6.1.

Results for CIFAR100 and Omniglot are shown in Table 1. CD-IMM is confirmed the best approach on both benchmarks, as expected by our theoretical arguments. More specifically, on CIFAR100 the CD-IMM is better than SLDA by 0.8 (although this difference is smaller than the standard deviation), while on Omniglot the difference is 35.74. The difference between the two benchmarks is partially due to their intrinsic difficulty, and partially due to the power of the pretrained fixed feature extractors.

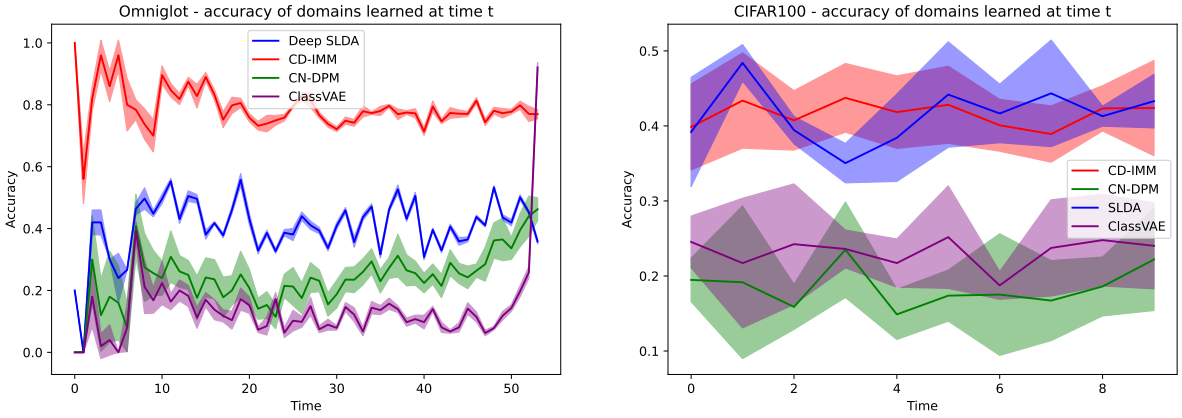
CN-DPM and ClassVAE have worse results than Deep SLDA on both benchmarks. It must be pointed out that Deep SLDA reaches these results with a single hyperparameter (the flag to allow the streaming update of covariances), while CN-DPM and ClassVAE require careful tuning. Overall, While the CD-IMM is more accurate, both the Deep SLDA and CD-IMM are confirmed as very robust choices for incremental training.

Figure 5b and 5a show the accuracy of the final models split by domains seen at time t. On Omniglot (left figure) we find a clear sign of catastrophic forgetting for CN-DPM and Class-VAE, with higher accuracies for the later timesteps and lower accuracies at the beginning. Deep SLDA and CD-IMM have a stable accuracy over time, and the fluctuations can be mostly attributed to the inherent different in complexities between data at different timesteps, which is to be expected in a continual learning stream.

One results that was a bit suriprising and not in perfect agreement with our hypothesis is that Deep SLDA does not show much forgetting. While low amounts of forgetting are present (e.g. first 10 timesteps on Omniglot, time 4 for CIFAR100), Deep SLDA seems more robust than ex-

Table 1: Average accuracy at the end of training. Mean and standard deviation are computed using 5 independent runs. Best in bold.

	MOONS	CIFAR100	Omniglot
ClassVAE	$82.54 \pm (0.50)$	$23.22 \pm (1.17)$	$14.70 \pm (0.26)$
Deep SLDA	$88.31 \pm (0.03)$	$41.52 \pm (0.05)$	$41.74 \pm (0.11)$
CN-DPM	$86.79 \pm (0.56)$	$18.53 \pm (1.20)$	$26.60 \pm (3.91)$
CD-IMM (ours)	$\mathbf{100.00} \pm (0.00)$	$\mathbf{41.60} \pm (0.14)$	$\mathbf{77.48} \pm (0.36)$



(a) (b)

Figure 5: Accuracy of the final model for the domains seen at time t on Omniglot (left) and CIFAR100 (right). Mean (solid line) and standard deviation (shaded area) computed over 5 runs.

pected. It is possible that the pretrained feature extractors tend to learn better features than we expected, resulting in a milder forgetting. Of course, this is highly dependent on the feature extractor and benchmark choice, and having a more powerful model will always be helpful in practice, as clearly exemplified by the results on Omniglot.

Figure 6 shows the training and inference times for all the methods on CIFAR100. We use the original implementation for all the baseline models, which are based on pytorch. Our method is around 3 times more expensive than Deep SLDA, while CN-DPM is more than 5 times more expensive, while ClassVAE is more than 12 times more expensive. Overall, the CD-IMM does not add a large overhead compared to Deep SLDA, while being more accurate in complex settings such as Omniglot.

Overall, the experiments confirm our hypothesis, showing that the CD-IMM is an accurate method robust to forgetting and natural class and domain drifts.

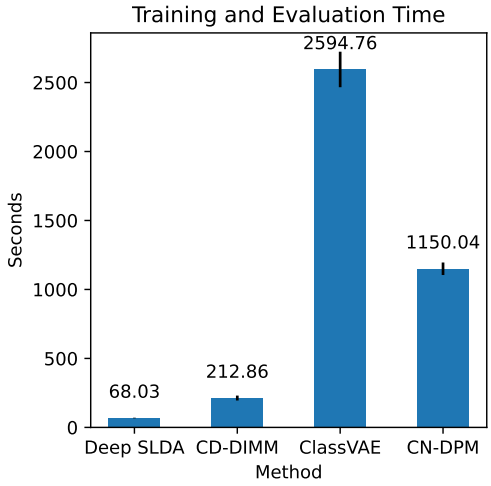
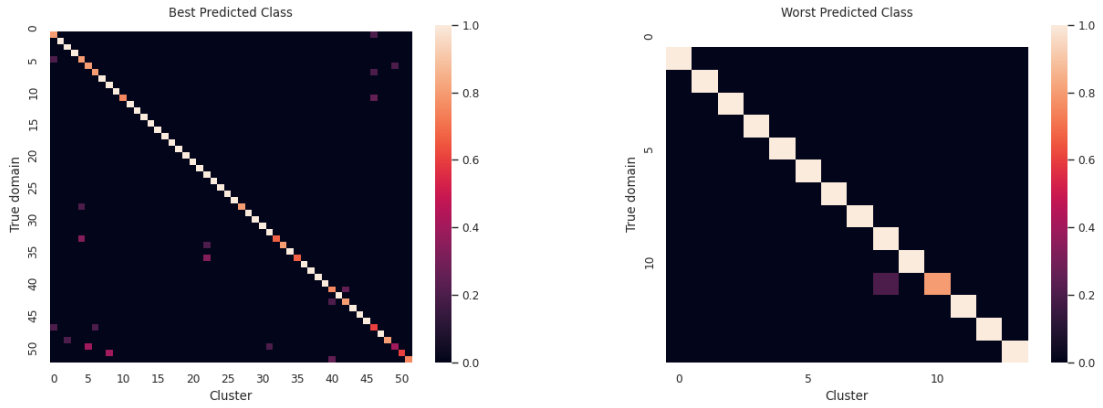


Figure 6: Training and inference times for all the methods on CIFAR100. Mean (blue bar) and standard deviation (black line) computed on 5 runs.

6.3 Latent Domains Analysis

Expected results: We expect that CD-IMM is able to learn the natural subclasses in Alphabet-Omniglot and CIFAR100-Superclasses (**H4**).

To assess the ability of CD-IMM to learn the natural subclasses in Omniglot, we compute the domain accuracy by assigning to each component the label of the most represented domain. The overall domain accuracy is 68.95%,



(a) (b)

Figure 7: The confusion matrix between true domains and CD-IMM components on the classes with the highest (on the left) and lowest (on the right) accuracy in the Omniglot dataset.

only 9% lower than the class accuracy, showing a good alignment between the label domains and the CD-IMM components even if the former ones are never observed during the training (we recall that only the class labels are observed during the training). To further investigate this behaviour, in Figure 7 we plot the confusion matrix between natural domains and CD-IMM components on two classes: the one with the highest and lowest accuracy score obtained by CD-IMM on the test set. We consider only the correct classified examples to compute the confusion matrix. The plot confirms our hypothesis since the confusion matrices are almost diagonal.

On CIFAR-100, the hypothesis **H4** is not confirmed since CD-IMM learns a single component for each class without capturing the natural partition of the data. We believe that this behaviour could be caused by the features extractor which is not able to map different domains of the same class into different areas of the latent space. We mean to perform more experiments to shed light on this aspect.

7 Conclusion

In this paper, we argued that the current design of Bayesian continual learning models is overfitted to the popular settings such as class-incremental benchmarks. Via a formal model of continual learning based on natural domain and class drifts co-occurring over time, we showed that many methods are unfit to handle even basic domain drifts. To overcome this limitations, we proposed the CD-IMM a bayesian non-parametric model that learn the natural cluster in the data incrementally adapting over time. Experimental results on streams with natural repetitions validate that the CD-IMM is an accurate and robust model. We hope these results will encourage researchers to carefully design their models and experimental settings to deal with more natural continual learning streams.

Acknowledgements

This paper is partially supported by the Italian Ministry of University and Research (MUR) as part of the FSE REACT-EU - PON 2014-2020 “Research and Innovation" resources – Innovation Action - DM MUR 1062/2021.

References

[1] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. “Expert Gate: Lifelong Learning with a Network of Experts”. In: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* 2017-Janua (2017), pp. 7120–7129. ISSN: 9781538604571. DOI: 10.1109/CVPR.2017.753.

[2] Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. “Task-Free Continual Learning”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, USA: IEEE, June 2019, pp. 11246–11255. ISBN: 978-1-72813-293-8. DOI: 10.1109/CVPR.2019.01151.

[3] Charles E Antoniak. “Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems”. In: *The annals of statistics* (1974), pp. 1152–1174.

[4] David Arthur and Sergei Vassilvitskii. “K-means++ the advantages of careful seeding”. In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. 2007, pp. 1027–1035.

[5] David M. Blei and Michael I. Jordan. "Variational inference for Dirichlet process mixtures". In: *Bayesian Analysis* 1.1 (2006), pp. 121–143. DOI: 10.1214/06-BA104.

[6] Guido Borghi, Gabriele Graffieti, and Davide Maltoni. *On the Challenges to Learn from Natural Data Streams*. Jan. 2023. DOI: 10.48550/arXiv.2301.03495. arXiv: 2301.03495 [cs].

[7] A. Cossu, A. Carta, and D. Bacciu. "Continual Learning with Gated Incremental Memories for Sequential Data Processing". In: *2020 International Joint Conference on Neural Networks (IJCNN)*. 2020, pp. 1–8. DOI: 10.1109/IJCNN48605.2020.9207550.

[8] Andrea Cossu et al. *Continual Pre-Training Mitigates Forgetting in Language and Vision*. May 2022. DOI: 10.48550/arXiv.2205.09357. arXiv: 2205.09357 [cs].

[9] Andrea Cossu et al. "Is Class-Incremental Enough for Continual Learning?" In: *Frontiers in Artificial Intelligence* 5 (2022). ISSN: 2624-8212. DOI: 10.3389/frai.2022.829842.

[10] Arthur P Dempster, Nan M Laird, and Donald B Rubin. "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the royal statistical society: series B (methodological)* 39.1 (1977), pp. 1–22.

[11] Sebastian Farquhar and Yarin Gal. "Towards Robust Evaluations of Continual Learning". In: (May 2018).

[12] Thomas S Ferguson. "A Bayesian analysis of some nonparametric problems". In: *The annals of statistics* (1973), pp. 209–230.

[13] Enrico Fini et al. "Self-Supervised Models Are Continual Learners". In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 9611–9620. DOI: 10.1109/CVPR52688.2022.00940.

[14] Robert M. French. "Catastrophic Forgetting in Connectionist Networks". In: *Trends in Cognitive Sciences* 3.4 (Apr. 1999), pp. 128–135. ISSN: 1364-6613. DOI: 10.1016/S1364-6613(99)01294-2.

[15] João Gama et al. "A Survey on Concept Drift Adaptation | ACM Computing Surveys". In: *ACM Computing Surveys* 46.4 (Mar. 2014), pp. 1–37. DOI: 10.1145/2523813.

[16] Tyler L. Hayes and Christopher Kanan. "Lifelong Machine Learning With Deep Streaming Linear Discriminant Analysis". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2020.

[17] Hamed Hemati et al. *Class-Incremental Learning with Repetition*. Jan. 2023. DOI: 10.48550/arXiv.2301.11396. arXiv: 2301.11396 [cs].

[18] Matthew D Hoffman et al. "Stochastic variational inference". In: *Journal of Machine Learning Research* (2013).

[19] James Kirkpatrick et al. "Overcoming Catastrophic Forgetting in Neural Networks". In: *Proceedings of the National Academy of Sciences* (2017), p. 201611835.

[20] Alex Krizhevsky. *Learning Multiple Layers of Features from Tiny Images*. Tech. rep. University of Toronto, Apr. 2009, p. 60.

[21] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. "Human-Level Concept Learning through Probabilistic Program Induction". In: *Science* 350.6266 (Dec. 2015), pp. 1332–1338. DOI: 10.1126/science.aab3050.

[22] Sang-Woo Lee et al. *Overcoming Catastrophic Forgetting by Incremental Moment Matching*. Jan. 2018. DOI: 10.48550/arXiv.1703.08475. arXiv: 1703.08475 [cs].

[23] Soochan Lee et al. "A Neural Dirichlet Process Mixture Model for Task-Free Continual Learning". In: *arXiv:2001.00689 [cs, stat]* (Jan. 2020). arXiv: 2001.00689 [cs, stat].

[24] Timothée Lesort et al. "Continual Learning for Robotics: Definition, Framework, Learning Strategies, Opportunities and Challenges". In: *Information Fusion* 58 (June 2020), pp. 52–68. ISSN: 1566-2535. DOI: 10.1016/j.inffus.2019.12.004. arXiv: 1907.00182.

[25] Timothée Lesort et al. "Generative Models from the Perspective of Continual Learning". In: July (2018), pp. 14–19. ISSN: 9781728120096.

[26] Vincenzo Lomonaco et al. "Avalanche: An End-to-End Library for Continual Learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 3600–3610.

[27] Zheda Mai et al. *Online Continual Learning in Image Classification: An Empirical Survey*. Oct. 2021. arXiv: 2101.10423 [cs].

[28] G. McLachlan and D. Peel. *Finite Mixture Models*. Wiley series in probability and statistics: Applied probability and statistics. Wiley, 2004. ISBN: 9780471654063.

[29] Radford M Neal. "Markov chain sampling methods for Dirichlet process mixture models". In: *Journal of computational and graphical statistics* 9.2 (2000), pp. 249–265.

[30] Dushyant Rao et al. “Continual Unsupervised Representation Learning”. In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019.

[31] Andrei A. Rusu et al. “Progressive Neural Networks”. In: (June 2016).

[32] Jayaram Sethuraman. “A constructive definition of Dirichlet priors”. In: *Statistica sinica* (1994), pp. 639–650.

[33] Gido M. van de Ven and Andreas S. Tolias. “Three Scenarios for Continual Learning”. In: *Arxiv preprint* (Apr. 2019). DOI: 10.48550/arXiv.1904.07734. arXiv: 1904.07734 [cs, stat].

[34] G. M. van de Ven, Z. Li, and A. S. Tolias. “Class-Incremental Learning with Generative Classifiers”. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Los Alamitos, CA, USA: IEEE Computer Society, June 2021, pp. 3606–3615. DOI: 10.1109/CVPRW53098.2021.00400.

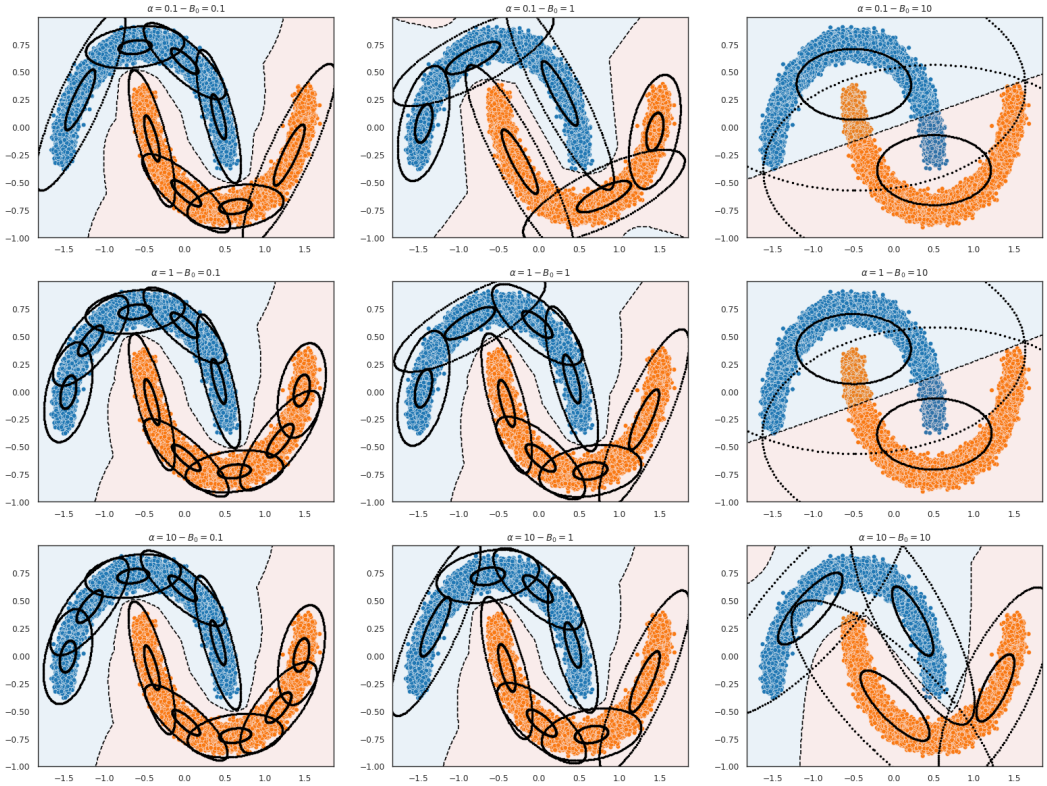


Figure 8: Decision boundary of CD-IMM on Moons dataset with different hyperparameters configurations.

A Other Plots

In Figure 8, we plot the decision boundary learning by CD-IMM on Moons dataset with different hyperparamters configurations. In particular, we consider the concentration α of the Dirichlet Process and the hyperparamter $\mathbf{\Psi}$ of the covariance matrix prior since they most affect the number of components created. When the prior does not penalise components with high variances, the CD-IMM uses a small number of components (right column of the figure). Viceversa, if the prior prefers small variances, we obtain more components (left columns of the figure).