Archipelago: Nonparametric Bayesian Semi-Supervised Learning

Ryan Prescott Adams RPA23@CAM.AC.UK
Cavendish Laboratory, University of Cambridge, Cambridge CB3 0HE, UK

Zoubin Ghahramani ZOUBIN@ENG.CAM.AC.UK
Engineering Department, University of Cambridge, Cambridge CB2 1PZ, UK
Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA 15213

Abstract

Semi-supervised learning (SSL), is classification where additional unlabeled data can be used to improve accuracy. Generative approaches are appealing in this situation, as a model of the data's probability density can assist in identifying clusters. Nonparametric Bayesian methods, while ideal in theory due to their principled motivations, have been difficult to apply to SSL in practice. We present a nonparametric Bayesian method that uses Gaussian processes for the generative model, avoiding many of the problems associated with Dirichlet process mixture models. Our model is fully generative and we take advantage of recent advances in Markov chain Monte Carlo algorithms to provide a practical inference method. Our method compares favorably to competing approaches on synthetic and real-world multi-class data.

1. Introduction

Semi-supervised learning (SSL) algorithms solve the problem of classification under the circumstance that only a subset of the training data are labeled. In contrast to the purely-supervised setting, semi-supervised learning assumes that the probability density of the data is important to discovering the decision boundary. Semi-supervised learning is motivated by the situation where copious training data are available, but hand-labeling the data is expensive.

From a Bayesian perspective, the natural way to perform semi-supervised learning is with a generative model of the data. One explicitly models the densities that gave rise to the observations, integrating out the class assignments for unlabeled data. We can then integrate out any parameters of the density model and arrive at predictive classifications of unseen data.

We would like to perform this modeling while making the fewest possible assumptions about the densities in question. Nonparametric Bayesian methods are appealing in this regard, as they incorporate an infinite number of parameters into the model. By using a model with an infinite number of parameters, we do not have to perform difficult computation to determine the appropriate dimensionality.

Unfortunately, many nonparametric Bayesian density models are ill-suited for the semi-supervised setting. Specifically, the main assumption in SSL is that data of the same class will cluster together; the labeled data provide the appropriate class and we use the unlabeled data to determine the cluster boundary. The most common nonparametric Bayesian density model, the Dirichlet Process Mixture Model (DPMM) (Escobar & West, 1995), suffers from the problem that the mixture components are located independently. There is no tendency for mixture models to form contiguous densities. If we take the natural approach of using a DPMM to flexibly model each class density in a semi-supervised learning problem, the mixtures will take unlabeled data away from other clusters.

Due to these difficulties with specifying a flexible density model, discriminative methods are more common than generative models for semi-supervised learning. The Bayesian discriminative model of Lawrence and Jordan (2005) use a Gaussian process (GP) to construct a nonparametric model for binary SSL. Similarly, Chu et al. (2007) and Sindhwani et al. (2007) specify nonparametric Bayesian discriminative models with GPs that exploit graph-based side information.

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009.

In this paper, we present a fully-Bayesian generative approach to semi-supervised learning that uses Gaussian processes to specify the prior on class densities. We call the model *Archipelago* as it performs Bayesian clustering with infinite-dimensional density models, but it prefers contiguous densities. These clusters can form irregular shapes, like islands in a chain.

2. The Archipelago Model

We consider a model on a D-dimensional real space, $\mathbb{R}^D$, with K possible labels. Let $\boldsymbol{x}$ be a point in $\mathbb{R}^D$. We define K scalar functions (one for each class) $\{g_k(\boldsymbol{x})\}_{k=1}^K$, where $g_k(\cdot) : \mathbb{R}^D \to \mathbb{R}$. Conditioned on a point $\boldsymbol{x}$, these functions are used with a softmax function to construct a categorical distribution for $l \in 1, 2, \ldots, K$, the class label of $\boldsymbol{x}$:

$$p(l \,|\, \boldsymbol{x}, \{g_k(\boldsymbol{x})\}_{k=1}^K) = \frac{\exp\{g_l(\boldsymbol{x})\}}{\Lambda(\boldsymbol{x})} \tag{1}$$

where we use $\Lambda(\boldsymbol{x}) = \sum_{k=1}^K \exp\{g_k(\boldsymbol{x})\}$ for notational convenience. We combine this discriminative classifier with a density model for $\boldsymbol{x}$ that is also constructed from the $g_k(\boldsymbol{x})$:

$$p(\boldsymbol{x} \,|\, \{g_k(\boldsymbol{x})\}_{k=1}^K) = \frac{1}{\mathcal{Z}} \frac{\Lambda(\boldsymbol{x})\, \pi(\boldsymbol{x})}{1 + \Lambda(\boldsymbol{x})} \tag{2}$$

where $\pi(\boldsymbol{x})$ is a *base density* that will be explained shortly. The constant $\mathcal{Z}$ is the normalization given by

$$\mathcal{Z} = \int_{\mathbb{R}^D} \mathrm{d}\boldsymbol{x}\, \frac{\Lambda(\boldsymbol{x})\, \pi(\boldsymbol{x})}{1 + \Lambda(\boldsymbol{x})}. \tag{3}$$

We combine Eqs. 1 and 2 to construct a joint likelihood for the location and label together:

$$p(\boldsymbol{x}, l \,|\, \{g_k(\boldsymbol{x})\}_{k=1}^K) = \frac{1}{\mathcal{Z}} \frac{\exp\{g_l(\boldsymbol{x})\}\, \pi(\boldsymbol{x})}{1 + \Lambda(\boldsymbol{x})}. \tag{4}$$

For the semi-supervised learning problem, this construction is appealing because Eq. 2 can be viewed as marginalizing out the class label in Eq. 4. This is in contrast to typical Bayesian generative methods for SSL which would construct $p(\boldsymbol{x}, l)$ from $p(\boldsymbol{x} \,|\, l)p(l)$, which does not necessarily allow the class label to be easily marginalized out. We have a set of K *coupled densities* which are each intractable individually, but whose sum is tractable.

2.1. Gaussian Process Prior on $g_k(\boldsymbol{x})$

To make the Archipelago model nonparametric, we use K independent Gaussian process priors on the functions $\{g_k(\boldsymbol{x})\}_{k=1}^K$. The GP allows one to specify general beliefs about functions, without choosing a finite set of basis functions. The idea of the GP is that the joint distribution over a discrete set of function values is a multivariate Gaussian determined by the corresponding inputs. The covariance matrix and mean vector that parameterize this Gaussian distribution arise from a positive definite covariance function $C(\cdot, \cdot) : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ and a mean function $m(\cdot) : \mathbb{R}^D \to \mathbb{R}$. Typically, the covariance function is chosen so that points near to each other in the input space have strongly correlating outputs. We will assume that the the kth GP covariance function has hyperparameter θ_k. We will take the mean functions to be zero. See Rasmussen and Williams (2006) for a comprehensive treatment of Gaussian processes.

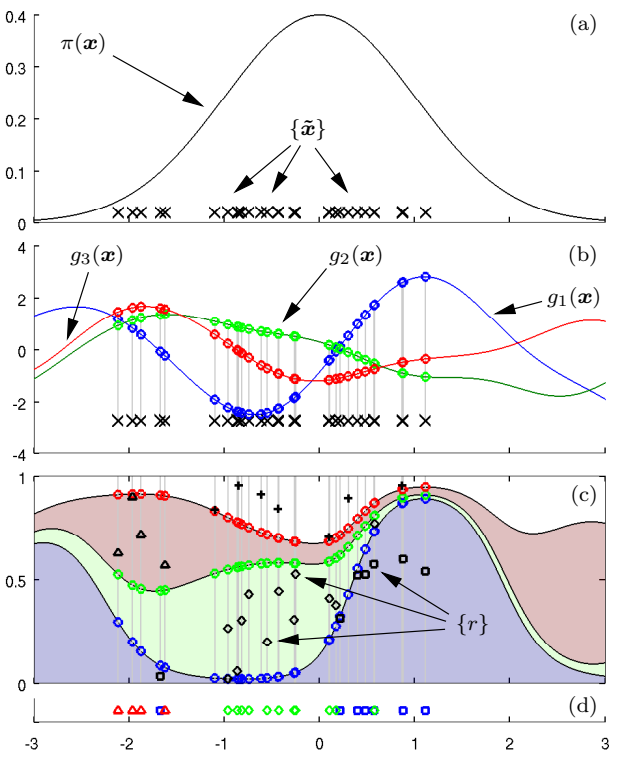


Figure 1. The generative process for Archipelago in one dimension. (a) Proposals are drawn from the base density $\pi(\boldsymbol{x})$. (b) Next, each of the $K = 3$ functions is sampled from the Gaussian process at the proposal locations. (c) The functions are used to partition the unit interval into $K + 1$ slices at each proposal location and uniform variates r are drawn in the vertical coordinate. (d) If r falls in the topmost partition, the proposal is rejected. Otherwise, it is accepted and assigned the class label based on the partition in which r falls.

2.2. The Generative Procedure

Coupling the likelihood of Eq. 4 with GP priors on the $\{g_k(\boldsymbol{x})\}_{k=1}^K$ enables us to define a fully-generative process for arriving at labeled data from a set of K random coupled densities. In other words, we can define a process that is equivalent to drawing a set of

Algorithm 1 Generate Q labeled data from the prior in Section 2

Inputs: Base density $\pi(\boldsymbol{x})$, GP hyperparameters $\{\theta_k\}_{k=1}^K$, number of samples to generate Q
Outputs: Q data/label pairs $\mathcal{Q} = \{\boldsymbol{x}_q, l_q\}_{q=1}^Q$

```
for k ← 1...K do
    X_k ← ∅, G_k ← ∅                          ▷ Initialize conditioning sets.
end for
Q ← ∅                                          ▷ Initialize return value.
while |Q| < Q do                               ▷ Until Q values accepted.
    x̃ ~ π(x)                                   ▷ Draw a proposal from the base density.
    for k ← 1...K do
        g_k(x̃) ~ GP(g | x̃, X_k, G_k, θ_k)     ▷ Draw the function from the GP.
        Λ(x̃) ← Λ(x̃) + exp{g_k(x̃)}
        X_k ← X_k ∪ x̃, G_k ← G_k ∪ g_k(x̃)      ▷ Update conditioning sets.
    end for
    r ~ U(0, 1)                                ▷ Draw a uniform variate on (0, 1).
    for k ← 1...K do
        ρ_k ← ρ_{k−1} + exp{g_k(x̃)} / (1+Λ(x̃))   ▷ Calculate the next partition.
        if r < ρ_k then
            Q ← Q ∪ {x̃, k}                     ▷ Accept this proposal and store.
            break
        end if
    end for
end while
```

data from the density in Eq. 2 with a random $\Lambda(x)$ arising from the GP priors on $\{g_k(\boldsymbol{x})\}_{k=1}^K$, and then labeling that data according to Eq. 1.

To generate a labeled datum, we first draw a proposal $\tilde{\boldsymbol{x}}$ from the base density $\pi(\boldsymbol{x})$. We then draw a function value from each of the GPs at the point $\tilde{\boldsymbol{x}}$. We denote these function draws as $\{g_k(\tilde{\boldsymbol{x}})\}_{k=1}^K$. We use the function values to partition the unit interval $[0, 1]$ into $K + 1$ non-overlapping segments with boundaries $0, \rho_1, \rho_2, \ldots, \rho_K, 1$ where

$$\rho_k = \rho_{k-1} + \frac{\exp\{g_1(\tilde{\boldsymbol{x}})\}}{1 + \Lambda(\tilde{\boldsymbol{x}})} \tag{5}$$

and $\rho_0 = 0$. We draw a uniform random variate r from $(0, 1)$ and either assign it label k if $\rho_{k-1} < r < \rho_k$, or reject the point if $r \geq \rho_K$. If we reject the $\tilde{\boldsymbol{x}}$, we make another proposal and continue until we accept. When making these new proposals, we sample conditionally from the GP, incorporating the function draws from previous proposals. We call the set of information already known about function $g_k(\boldsymbol{x})$ the *conditioning set*, with inputs $\boldsymbol{X}_k$ and outputs $\boldsymbol{G}_k$. We can continue this rejection/acceptance process, generating as many data points as we wish, "discovering" the set of functions $\{g_k(\boldsymbol{x})\}_{k=1}^K$ as we iterate. This procedure is given in Algorithm 1 and shown graphically in Figure 1.

This generative procedure is infinitely exchangeable and the data are exact in the sense that they are not biased by the starting state of a finite Markov chain. Infinite exchangeability means that the probability of a set of data is the same under any ordering. It is also not necessary to determine the $\{g_k(\boldsymbol{x})\}_{k=1}^K$ over any more than a finite number of points in $\mathbb{R}^D$, nor is it necessary to determine the normalization constant $\mathcal{Z}$ in order to generate these data.

3. Inference

We have so far described a nonparametric generative model for labeled data. We now use this model to perform semi-supervised learning. The Archipelago inference task to find the predictive distribution over class labels of an unlabeled datum $\boldsymbol{x}_\star$, given N labeled data $\{\boldsymbol{x}_n, l_n\}_{n=1}^N$ and P unlabeled data $\{\boldsymbol{x}_p\}_{p=1}^P$, integrating out the latent function $\{g_k(\boldsymbol{x})\}_{k=1}^K$. Treating the function $g_k(\boldsymbol{x})$ as an infinite vector $\boldsymbol{g}_k$, our objective is to integrate out these K vectors:

$$\begin{aligned} p(l_\star \,|\, \boldsymbol{x}_\star, \{\boldsymbol{x}_n, l_n\}_{n=1}^N, \{\boldsymbol{x}_p\}_{p=1}^P \,|\, \{\theta_k\}_{k=1}^K) = \\ \int \mathrm{d}\boldsymbol{g}_1 \cdots \int \mathrm{d}\boldsymbol{g}_K \; p(l_\star \,|\, \boldsymbol{x}_\star, \{\boldsymbol{g}_k\}_{k=1}^K) \\ \times p(\{\boldsymbol{g}_k\}_{k=1}^K \,|\, \{\boldsymbol{x}_n, l_n\}_{n=1}^N, \{\boldsymbol{x}_p\}_{p=1}^P, \{\theta_k\}_{k=1}^K) \end{aligned} \tag{6}$$

where the posterior distribution on $\{\boldsymbol{g}_k\}_{k=1}^K$ arises from Eq.s 2 and 4 and is proportional to

$$\begin{aligned} p(\{\boldsymbol{g}_k\}_{k=1}^K, \{\boldsymbol{x}_n, l_n\}_{n=1}^N, \{\boldsymbol{x}_p\}_{p=1}^P \,|\, \{\theta_k\}_{k=1}^K) = \\ \mathcal{Z}^{-N-P} \prod_{k=1}^K \mathcal{GP}(\boldsymbol{g}_k \,|\, \{\boldsymbol{x}_n\}_{n=1}^N, \{\boldsymbol{x}_p\}_{p=1}^P, \theta_k) \\ \times \prod_{n=1}^N \frac{\exp\{g_{l_n}(\boldsymbol{x}_n)\}\; \pi(\boldsymbol{x}_n)}{1 + \Lambda(\boldsymbol{x}_n)} \prod_{p=1}^P \frac{\Lambda(\boldsymbol{x}_p)\; \pi(\boldsymbol{x}_p)}{1 + \Lambda(\boldsymbol{x}_p)}. \end{aligned} \tag{7}$$

Given a set of samples of $\{\boldsymbol{g}_k\}_{k=1}^K$, we can es-

timate the distribution in Eq. 6 via a sum, but acquiring samples from the posterior on the $\boldsymbol{g}_k$ is difficult for two reasons: 1) Each $\boldsymbol{g}_k$ is an infinite-dimensional object and we cannot store it naïvely in memory, and 2) the posterior distribution $p(\{\boldsymbol{g}_k\}_{k=1}^K \,|\, \{\boldsymbol{x}_n, l_n\}_{n=1}^N, \{\boldsymbol{x}_p\}_{p=1}^P, \{\theta_k\}_{k=1}^K)$ is *doubly-intractable* due to the presence of $\mathcal{Z}$ in Eq. 7.

In Bayesian inference, it is common to have an unnormalized posterior distribution, where the unknown constant is intractable, but not dependent on the parameters; Markov chain Monte Carlo (MCMC) methods are well-suited for this situation. Doubly-intractable posterior distributions (Murray et al., 2006) are those in which the likelihood function also has an intractable constant that *does* depend on the model parameters. This situation arises most commonly in undirected graphical models, such as Ising models (where $\mathcal{Z}$ is called the *partition function*), but it is also found in density models and doubly-stochastic processes, e.g. the Log Gaussian Cox Process. In such cases, even MCMC – the "sledgehammer" of Bayesian inference – is difficult or impossible to apply.

Fortunately, recent advances in MCMC methods have made it possible to perform inference in doubly-intractable models under special circumstances. If it is possible to generate "fantasy data" exactly from a particular setting of the parameters, then it is possible to perform inference using the method of Møller et al. (2006), Exchange Sampling (Murray et al., 2006), or modeling of the latent history of the generative procedure (Beskos et al., 2006). In general, these methods were developed with the undirected graphical model in mind, where it is sometimes possible to generate exact data via Coupling From The Past. In the case of Archipelago, we follow the approach of Adams et al. (2009) in modeling the latent history of the generative process. This approach not only resolves the problem of double intractability, but it also results in a Markov chain with an infinite-dimensional state that requires only a finite amount of memory.

Given a set of data on which we place the prior of Section 2, we are asserting that these data were generated via the procedure of Section 2.2. If we knew the history of the generative process that resulted in our observed data, then we would know everything necessary to make a prediction at a new location $\boldsymbol{x}_\star$. We perform posterior inference on this *latent history* by introducing a set of latent variables: 1) the number of latent rejections M, 2) the locations of the latent rejections $\{\boldsymbol{x}_m\}_{m=1}^M$, and 3) the values of the functions at the rejections, at the labeled data and at the unlabeled data. We can run the generative algorithm without calculating $\mathcal{Z}$ and without knowing the functions $g_k(\boldsymbol{x})$ everywhere, and via the latent variable model we are able to inherit these properties for inference.

Integrating out the classes of the unlabeled data, as in Eq. 2, the joint distribution over the function values, the number and location of the latent rejections and the fixed data is

$$p(\{\boldsymbol{g}_k\}_{k=1}^K, M, \{\boldsymbol{x}_m\}_{m=1}^M, \{\boldsymbol{x}_p\}_{p=1}^P, \{\boldsymbol{x}_n, l_n\}_{n=1}^N) = \prod_{k=1}^K \mathcal{GP}(\{g_k(\boldsymbol{x}_n)\}_{n=1}^N, \{g_k(\boldsymbol{x}_p)\}_{p=1}^P, \{g_k(\boldsymbol{x}_m)\}_{m=1}^M) \times \prod_{n=1}^N \frac{\exp\{g_{l_n}(\boldsymbol{x}_n)\}\ \pi(\boldsymbol{x}_n)}{1+\Lambda(\boldsymbol{x}_n)} \prod_{p=1}^P \frac{\Lambda(\boldsymbol{x}_p)\ \pi(\boldsymbol{x}_p)}{1+\Lambda(\boldsymbol{x}_p)} \times \prod_{m=1}^M \frac{\pi(\boldsymbol{x}_m)}{1+\Lambda(\boldsymbol{x}_m)}. \quad (8)$$

This joint distribution is implicitly conditioned on the GP hyperparameters $\{\theta_k\}_{k=1}^K$. We have abused notation slightly by not explicitly conditioning the GP on the input locations.

The joint distribution in Eq. 8 is proportional to the posterior distribution over the latent state, and we sample from it with three kinds of Markov transitions: updating the number of latent rejections, updating the rejection locations, and updating the function values.

3.1. Sampling the Number of Rejections

We take advantage of the infinite exchangeability of the generative process to construct a Metropolis–Hastings move that can add or remove rejections from the latent history. Our model for inference is that the data came about as the result of Algorithm 1, stopping when there were $N+P$ data accepted. The unlabeled data were generated by the P labels being discarded, and the joint distribution in Eq. 8 integrates out the unknown label. Due to infinite exchangeability, we can propose a reordering of the latent history at any time and this move will always be accepted. Thus to insert a new latent rejection, we propose moving a rejection that occurred *after* the $N+P$th acceptance to be moved to sometime *before* it. This idea is shown graphically in Figure 2. To delete a rejection, we make the opposite type of transition: select one of the current rejections at random and propose moving it to after the $N+P$th acceptance.

In practice, it is not necessary to actually maintain an ordering. We define a function $b(\cdot,\cdot) : \mathbb{N} \times \mathbb{N} \to (0,1)$ and we decide randomly whether to propose adding or removing a rejection with probabilities $b(M, N+P)$ and $1-b(M, N+P)$, respectively. If we decide to add

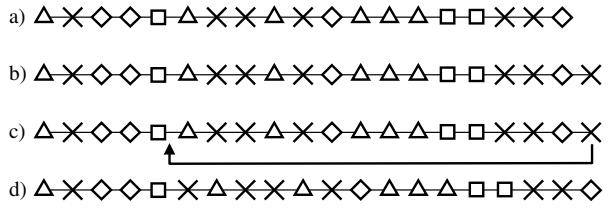


Figure 2. a) An initial state of the latent history, with three possible classes ($K = 3$, shown as ◇, △, and □), and some latent rejections ✕ ($M = 6$). b) Propose a new rejection after the last acceptance by running the procedure forward. c) Propose moving this new rejection into a random location in the history. d) If accepted, we now have a history with an additional latent rejection ($M = 7$).

an additional rejection, we first propose a new location for it, $\hat{\boldsymbol{x}}$, by drawing it from $\pi(\boldsymbol{x})$. We then draw each of the K function values at that location, $\{g_k(\hat{\boldsymbol{x}})\}_{k=1}^K$, conditioning on all of the current function values. Finally, we accept or reject with Metropolis–Hastings acceptance ratio

$$a_{\mathsf{ins}} = \frac{(1 - b(M+1, N+P))\ (M+N+P)}{b(M, N+P)\ (M+1)\ (1+\Lambda(\hat{\boldsymbol{x}}))}. \quad (9)$$

If we are proposing a deletion, we choose an index m uniformly from among the M rejections and remove it with Metropolis–Hastings acceptance ratio

$$a_{\mathsf{del}} = \frac{b(M-1, N+P)\ M\ (1+\Lambda(\boldsymbol{x}_m))}{(1 - b(M, N+P))\ (M+N+P-1)}. \quad (10)$$

In practice we have often found it reasonable to simply set $b(M, N+P) = \frac{1}{2}$. More sophisticated proposals may yield improvements in mixing, but we have not thoroughly explored this topic. Typically we make several of these transitions (≈ 10) for each of the transitions in Sections 3.2 and 3.3.

3.2. Sampling the Rejection Locations

Conditioned on the current function values and the number of rejections M, we also want to update the locations of the rejections. We iterate over each of the M rejections and make a Metropolis–Hastings proposal with two stages. First, we draw a new location $\hat{\boldsymbol{x}}_m$ from a proposal density $q(\hat{\boldsymbol{x}}_m \leftarrow \boldsymbol{x}_m)$. Second, we draw a new set of function values at this location, $\{g_k(\hat{\boldsymbol{x}}_m)\}_{k=1}^K$, from the GP, conditioning on all of the current function values and locations. We then accept or reject the proposal according to the acceptance ratio

$$a_m = \frac{q(\boldsymbol{x}_m \leftarrow \hat{\boldsymbol{x}}_m)\ \pi(\hat{\boldsymbol{x}}_m)\ (1+\Lambda(\boldsymbol{x}_m))}{q(\hat{\boldsymbol{x}}_m \leftarrow \boldsymbol{x})\ \pi(\boldsymbol{x}_m)\ (1+\Lambda(\hat{\boldsymbol{x}}_m))}. \quad (11)$$

3.3. Sampling the Function Values

Sampling the function is the most critical aspect of the model: it is the function values that we depend on to make predictions. We iterate over each of the K functions and use Hamiltonian Monte Carlo (HMC) (Neal, 1997) for efficient sampling. HMC augments the Markov chain with random "momenta" and then simulates Hamiltonian dynamics in fictional time. In this way, Metropolis–Hastings proposals are guided using gradient information and random walk behavior is avoided. We perform gradient calculations in the "whitened" space resulting from transforming the function values with the inverse Cholesky decomposition of the covariance matrix. This is a more natural metric space in which to find gradients. For the kth function, the log conditional posterior distribution is

$$\begin{aligned}\ln p(\boldsymbol{g}_k \mid M, \{\boldsymbol{x}_m\}_{m=1}^M, \{\boldsymbol{x}_n, l_n\}_{n=1}^N, \{\boldsymbol{x}_p\}_{p=1}^P, \theta_k) = \\ -\frac{1}{2}\boldsymbol{g}_k^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{g}_k + \boldsymbol{g}_k^\mathsf{T}\mathbf{1}_{l_n=k} - \sum_{n=1}^N \ln(1+\Lambda(\boldsymbol{x}_n)) \quad (12) \\ + \sum_{p=1}^P \ln\frac{\Lambda(\boldsymbol{x}_p)}{1+\Lambda(\boldsymbol{x}_p)} - \sum_{m=1}^M \ln(1+\Lambda(\boldsymbol{x}_m)) + \text{const}\end{aligned}$$

where $\mathbf{1}_{l_n=k}$ is a vector whose components are 1 where $l_n = k$, and 0 otherwise. $\boldsymbol{\Sigma}$ is the covariance matrix resulting from application of the covariance kernel to the concatenation of labeled, unlabeled and rejected data locations. Eq. 12 comes from: 1) a GP prior term, 2) a labeled data acceptance term, 3) an unlabeled data acceptance term, and 4) a rejection term.

3.4. Sampling the Hyperparameters

An appealing feature of Bayesian inference methods is the ability to perform hierarchical inference. Here, we sample the hyperparameters, $\{\theta_k\}_{k=1}^K$, governing the GP covariance function. Conditioned on the number and locations of the rejections, and the function values, we do hyperparameter inference via Hamiltonian Monte Carlo as described by Neal (1997).

We can also introduce hyperparameters ϕ into the base density $\pi(\boldsymbol{x})$. For example, if $\pi(\boldsymbol{x})$ is a Gaussian distribution, ϕ might be the mean and covariance parameters. Conditioning on the number and locations of the rejections, inference of ϕ is a standard MCMC parametric density estimation problem that can be solved with Gibbs sampling or Metropolis–Hastings.

3.5. Making Predictions

With samples from the posterior distribution over the function values, we can now approximate the integral in Eq. 6. At each MCMC step after burn-in, we sample

the function values $\{g_k(\boldsymbol{x}_\star)\}_{k=1}^K$ from the GP, conditioned on the current state. From these we can approximate the predictive distribution via a mixture of softmax functions. We might also be interested in the class-conditional predictive distributions. These K distributions are the ones that arise on data space, conditioning on membership in class k, but integrating out the latent function and hyperparameters. While not available in closed form, it is straightforward to generate predictive fantasies. At each MCMC step after burn-in, run the generative process forward from the current state until a datum is accepted that is a member of class k.

4. Empirical Results

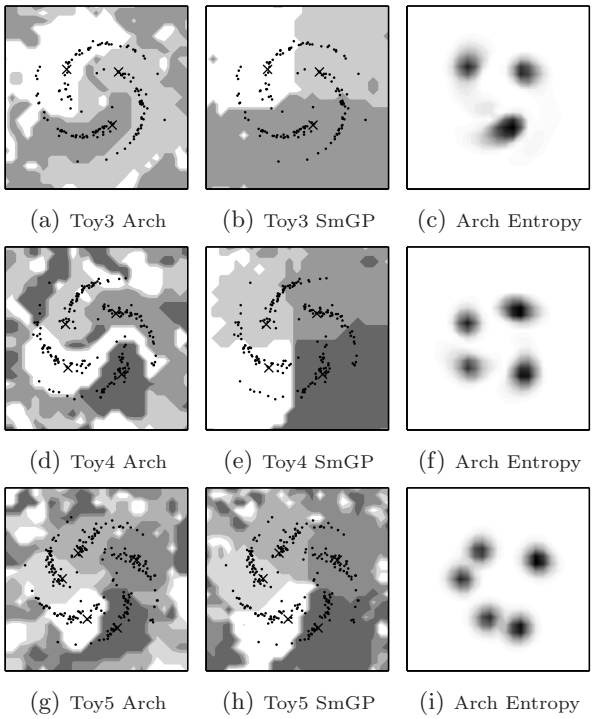
We compared the Archipelago model and inference method to three other multi-class Gaussian process classification approaches: the standard softmax GP classifier (Williams & Barber, 1998), the probit classifier of Girolami and Rogers (2006), and the multiclass extension (Rogers & Girolami, 2007) of the Null Category Noise Model of Lawrence and Jordan (2005). We compared these models on three two-dimensional toy problems and two real-world datasets. Unlabeled data were ignored in the softmax and probit models.

4.1. Toy Pinwheel Data

The toy problems were noisy pinwheels with three, four and five classes, shown in Figures 3(a), 3(d), and 3(g), respectively. There were 50 data points in each set and the algorithms were run with 1, 2, 4, and 8 labeled data in each class. 300 held-out data for each class were used to evaluate the predictive distributions by error rate and perplexity. Inference for each method was performed using MCMC, and each used an isotropic squared-exponential covariance function. For each method we also sampled from the length-scale and amplitude hyperparameters using Hamiltonian Monte Carlo (HMC) (Neal, 1997). The Archipelago base density was a bivariate Gaussian. Figure 3 shows the predictive modes for the Archipelago and softmax models, as well as the entropy of the Archipelago predictive distribution as a function of space. Numerical results are in Table 1.

4.2. Wine Data

The wine data are thirteen-dimensional, with three classes from Aeberhard et al. (1992) via Asuncion and Newman (2007). As in Rogers and Girolami (2007), we translated and scaled the inputs to have zero mean and unit variance. We used an ARD covariance function (Rasmussen & Williams, 2006) and HMC for inference of length scales and amplitude. We divided the data into two sets of 89 for training and testing, with the classes divided approximately evenly. We ran four sets of experiments with each method, as with the toy data, with 1, 2, 4, and 8 labeled data in each class. We used a multivariate Gaussian for the Archipelago base density. The results are given in Table 1.



(a) Toy3 Arch (b) Toy3 SmGP (c) Arch Entropy

(d) Toy4 Arch (e) Toy4 SmGP (f) Arch Entropy

(g) Toy5 Arch (h) Toy5 SmGP (i) Arch Entropy

Figure 3. The results of applying Archipelago to "pinwheel" data with one labeled datum per class (marked with an 'x'). The top row has three classes, the middle row has four classes and the bottom has five. The figures on the left show the modes of the class assignments from Archipelago. The middle column shows the class assignment modes using a softmax GP. The right column shows the entropies of the predictive distribution produced by Archipelago.

4.3. Oil Pipe Data

The oil pipe data are included with software for Williams and Barber (1998). The task is to classify the type of flow in a pipe (laminar, hydrogenous, or amorphous), using a set of fourteen gamma ray readings. We split the 400 data into 134 training and 266 testing. We used an ARD covariance function and followed the same procedures as in Section 4.2. The results are given in Table 1.

5. Discussion

Archipelago combines ideas both from Gaussian process density modeling and GP classification. When using a kernel such as the squared-exponential, it ex-

presses the idea that similar data should have similar probabilities *and* similar class assignments. In contrast, an SSL approach with a mixture model for each class must maintain both a class and mixture component assignment for each datum, and the notion of smoothness does not extend beyond the parametric form of the components.

The most relevant Bayesian model to Archipelago is the discriminative Null Category Noise Model (NCNM) (Lawrence & Jordan, 2005) and the multi-class extension of Rogers and Girolami (2007). The NCNM enforces the idea of "margin" in a GP classifier by requiring a null-class transition between any two "real" classes. The latent rejections play a similar role in Archipelago by allowing regions with mass under $\pi(\boldsymbol{x})$ to have no data. However, Archipelago models data in the null class explicitly as part of inference, to let Archipelago model arbitrarily complex densities. In contrast to Archipelago, NCNM requires *a priori* setting of the null-category width and labeling probability. In almost all of our tests, Archipelago had lower test classification error than the NCNM. In the regime with few labels, Archipelago outperformed the other methods on test error.

In supervised classification and regression with GPs (and in the NCNM), it is sufficient to know only the values of the functions at the observed data. In density modeling, however, the areas where there is little density are as important as those with significant density. Unfortunately, data are only likely to be observed in areas of notable density, so we must have some way to "peg down" the functions in the low-density areas. The explicitly-modeled latent rejections, although a part of the generative model, fulfill this need in a way that would be difficult to motivate in a purely discriminative setting.

Table 1. Results from four methods applied to test data: Archipelago, the softmax GP (SmGP), the probit GP (PrGP) and the Null Category Noise Model (NCNM).

			Arch	SmGP	PrGP	NCNM
Toy3	1	err	**0.167**	0.347	0.290	0.612
		perp	**1.461**	2.588	1.631	14.120
	2	err	**0.111**	0.312	0.279	0.462
		perp	1.565	2.769	**1.350**	4.801
	4	err	**0.042**	0.118	0.142	0.114
		perp	1.092	1.310	1.041	**1.002**
	8	err	**0.037**	0.056	0.137	0.070
		perp	1.070	1.065	1.043	**1.000**
Toy4	1	err	**0.120**	0.305	0.416	0.554
		perp	2.478	3.436	**1.239**	16.160
	2	err	**0.092**	0.188	0.253	0.429
		perp	**2.471**	2.861	7.896	8.257
	4	err	**0.032**	0.119	0.077	0.056
		perp	1.469	1.505	1.343	**1.113**
	8	err	**0.025**	0.061	0.088	0.0533
		perp	1.135	1.076	1.086	**1.009**
Toy5	1	err	**0.183**	0.192	0.515	0.482
		perp	3.766	4.519	**3.533**	294.758
	2	err	**0.124**	0.181	0.178	0.365
		perp	2.811	2.008	**1.476**	11.669
	4	err	**0.081**	0.159	0.113	0.093
		perp	2.345	1.730	1.375	**1.210**
	8	err	**0.051**	0.095	0.073	**0.051**
		perp	2.105	1.290	1.201	**1.093**
Wine	1	err	**0.141**	0.260	0.303	0.393
		perp	1.928	2.790	**1.769**	31.832
	2	err	**0.135**	0.292	0.213	0.393
		perp	**2.654**	2.760	2.851	2.799
	4	err	**0.090**	0.146	0.101	0.101
		perp	1.198	2.440	1.387	**1.153**
	8	err	0.0899	**0.067**	**0.067**	**0.067**
		perp	1.084	1.250	1.224	**1.045**
Oil Pipe	1	err	0.538	0.568	**0.432**	0.650
		perp	**2.997**	3.024	5.631	3.56K
	2	err	**0.297**	0.331	0.308	0.365
		perp	2.451	2.576	3.499	**1.024**
	4	err	**0.211**	0.215	0.271	0.176
		perp	1.658	1.864	2.558	**1.301**
	8	err	0.068	0.053	0.068	**0.038**
		perp	1.241	1.137	1.232	**1.002**

5.1. Computational Considerations

Gaussian processes have relatively high computational costs, and Archipelago inherits them. With a naïve implementation, the time complexity per MCMC step is $O(K(N+P+M))^3$ and the space complexity is $O(K(N+P+M)^2)$. There is a large literature on sparse approximations to GPs, e.g. Quiñonero-Candela and Rasmussen (2005), but we have not explored this topic. Mixing of Markov chains is an additional concern, but we have not found this to be a problem in practice due to our use of Hamiltonian Monte Carlo. Figure 4 shows an autocorrelation plot of the number of latent rejections during an MCMC run on the Toy5 data.

As the computational complexity grows rapidly with the number of latent rejections, minimizing these rejections is paramount. The number of latent rejections relates directly to the mismatch between the base density $\pi(\boldsymbol{x})$ and the true data density. Hyperparameter inference as described in Section 3.4 can lower the computational burden by adapting $\pi(\boldsymbol{x})$ to the data, but it is important to choose an appropriate parametric family. In high dimensions, this choice is difficult to make, and in the absence of significant domain knowledge we expect that Archipelago will not work well in high dimensions.

5.2. Future Directions and Model Variations

One advantage of Gaussian processes for density modeling is that GPs can be applied to alternate domains.

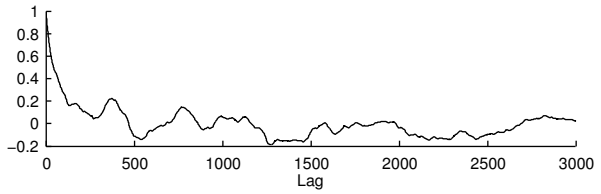


Figure 4. Autocorrelation plot of the number of rejections in Toy5 with one labeled datum during MCMC simulation.

Positive definite kernels exist for spaces such as graphs and permutation groups, and Archipelago can be applied directly to SSL problems in these domains.

In this paper we have assumed that the K GPs are independent. It may be useful to introduce dependency between the latent functions using, for example, the method of Teh et al. (2005). We have also assumed that the number of classes K is known and that an example from each class has been observed. It would be interesting to consider a model where the number of true classes can exceed the number of observed classes. In this case, it may be useful to allow for infinite K.

5.3. Summary of Contributions

We presented a nonparametric Bayesian generative approach to the semi-supervised learning problem. It works for any number of classes and does not require any finite-dimensional approximation for inference. It improves over mixture-based Bayesian approaches to SSL while still modeling complex density functions. In empirical tests, our model compares favorably with Bayesian discriminative approaches, particularly when little labeled data are available.

Acknowledgements

Ryan Adams thanks the Gates Cambridge Trust and the Canadian Institute for Advanced Research.

References

Adams, R. P., Murray, I., & MacKay, D. J. C. (2009). The Gaussian process density sampler. *Advances in Neural Information Processing Systems 21* (pp. 9–16).

Aeberhard, S., Coomans, D., & de Vel, O. (1992). *Comparison of classifiers in high dimensional settings* (Technical Report 92-02). James Cook University.

Asuncion, A., & Newman, D. (2007). UCI machine learning repository. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Beskos, A., Papaspiliopoulos, O., Roberts, G. O., & Fearnhead, P. (2006). Exact and computationally efficient likelihood-based estimation for discretely observed diffusion processes. *Journal of the Royal Statistical Society, Series B*, *68*, 333–382.

Chu, W., Sindhwani, V., Ghahramani, Z., & Keerthi, S. S. (2007). Relational learning with Gaussian processes. *Advances in Neural Information Processing Systems 19* (pp. 289–296).

Escobar, M. D., & West, M. (1995). Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, *90*, 577–588.

Girolami, M., & Rogers, S. (2006). Variational Bayesian multinomial probit regression with Gaussian process priors. *Neural Computation*, *18*, 1790–1817.

Lawrence, N. D., & Jordan, M. I. (2005). Semi-supervised learning via Gaussian processes. *Advances in Neural Information Processing Systems 17* (pp. 753–760).

Møller, J., Pettitt, A. N., Reeves, R., & Berthelsen, K. K. (2006). An efficient Markov chain Monte Carlo method for distributions with intractable normalising constants. *Biometrika*, *93*, 451–458.

Murray, I., Ghahramani, Z., & MacKay, D. (2006). MCMC for doubly-intractable distributions. *Uncertainty in Artificial Intelligence 22* (pp. 359–366).

Neal, R. M. (1997). *Monte Carlo implementation of Gaussian process models for Bayesian regression and classification* (Technical Report 9702). Department of Statistics, University of Toronto.

Quiñonero-Candela, J., & Rasmussen, C. E. (2005). A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, *6*, 1935–1959.

Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian processes for machine learning*. Cambridge, MA: MIT Press.

Rogers, S., & Girolami, M. (2007). Multi-class semi-supervised learning with the ϵ-truncated multinomial probit Gaussian process. *Journal of Machine Learning Research: Gaussian Processes in Practice* (pp. 17–32).

Sindhwani, V., Chu, W., & Keerthi, S. S. (2007). Semi-supervised Gaussian process classifiers. *International Joint Conference on Artificial Intelligence* (pp. 1059–1064).

Teh, Y. W., Seeger, M., & Jordan, M. I. (2005). Semiparametric latent factor models. *International Conference on Artificial Intelligence and Statistics 10* (pp. 333–340).

Williams, C. K. I., & Barber, D. (1998). Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*, 1342–1351.