Journal of Machine Learning Research 13 (2012) 2567-2588 Submitted 4/12; Revised 7/12; Published 9/12

Nonparametric Guidance of Autoencoder Representations using Label Information

Jasper Snoek JASPER@CS.TORONTO.EDU
Department of Computer Science
University of Toronto
Toronto, ON, Canada

Ryan P. Adams RPA@SEAS.HARVARD.EDU
School of Engineering and Applied Sciences
Harvard University
Cambridge, MA, USA

Hugo Larochelle HUGO.LAROCHELLE@USHERBROOKE.EDU
Department of Computer Science
University of Sherbrooke
Sherbrooke, QC, Canada

Editor: Neil Lawrence

Abstract

While unsupervised learning has long been useful for density modeling, exploratory data analysis and visualization, it has become increasingly important for discovering features that will later be used for discriminative tasks. Discriminative algorithms often work best with highly-informative features; remarkably, such features can often be learned without the labels. One particularly effective way to perform such unsupervised learning has been to use autoencoder neural networks, which find latent representations that are constrained but nevertheless informative for reconstruction. However, *pure* unsupervised learning with autoencoders can find representations that may or may not be useful for the ultimate discriminative task. It is a continuing challenge to guide the training of an autoencoder so that it finds features which will be useful for predicting labels. Similarly, we often have *a priori* information regarding what statistical variation will be irrelevant to the ultimate discriminative task, and we would like to be able to use this for guidance as well. Although a typical strategy would be to include a parametric discriminative model as part of the autoencoder training, here we propose a nonparametric approach that uses a Gaussian process to guide the representation. By using a nonparametric model, we can ensure that a useful discriminative function exists for a given set of features, without explicitly instantiating it. We demonstrate the superiority of this guidance mechanism on four data sets, including a real-world application to rehabilitation research. We also show how our proposed approach can learn to explicitly ignore statistically significant covariate information that is label-irrelevant, by evaluating on the small NORB image recognition problem in which pose and lighting labels are available.

Keywords: autoencoder, gaussian process, gaussian process latent variable model, representation learning, unsupervised learning

1. Introduction

One of the central tasks of machine learning is the inference of latent representations. Most often these can be interpreted as representing aggregate features that explain various properties of the data.

In probabilistic models, such latent representations typically take the form of unobserved random variables. Often this latent representation is of direct interest and may reflect, for example, cluster identities. It may also be useful as a way to explain statistical variation as part of a density model. In this work, we are interested in the discovery of latent features which can be later used as alternate representations of data for discriminative tasks. That is, we wish to find ways to extract statistical structure that will make it as easy as possible for a classifier or regressor to produce accurate labels.

We are particularly interested in methods for learning latent representations that result in fast feature extraction for out-of-sample data. We can think of these as devices that have been trained to perform rapid approximate inference of hidden values associated with data. Neural networks have proven to be an effective way to perform such processing, and autoencoder neural networks, specifically, have been used to find representatons for a variety of downstream machine learning tasks, for example, image classification (Vincent et al., 2008), speech recognition (Deng et al., 2010), and Bayesian nonparametric modeling (Adams et al., 2010).

The critical insight of the autoencoder neural network is the idea of using a constrained (typically either sparse or low-dimensional) representation within a feedforward neural network. The training objective induces the network to learn to reconstruct its input at its output. The constrained central representation at the bottleneck forces the network to find a compact way to explain the statistical variation in the data. While this often leads to representations that are useful for discriminative tasks, it does require that the salient variations in the data distribution be relevant for the eventual labeling. This assumption does not necessarily always hold; often irrelevant factors can dominate the input distribution and make it poorly-suited for discrimination (Larochelle et al., 2007). In previous work to address this issue, Bengio et al. (2007) introduced weak supervision into the autoencoder training objective by adding label-specific output units in addition to the reconstruction. This approach was also followed by Ranzato and Szummer (2008) for learning document representations.

The difficulty of this approach is that it complicates the task of learning the autoencoder representation. The objective now is to learn not only a hidden representation that is good for reconstruction, but also one that is immediately good for discrimination under the simplified choice of model, for example, logistic regression. This is undesirable because it potentially prevents us from discovering informative representations for the more sophisticated nonlinear classifiers that we might wish to use later. We are forced to solve two problems at once, and the result of one of them (the classifier) will be immediately thrown away.

Here we propose a different take on the issue of introducing supervised guidance into autoencoder representations. We consider Gaussian process priors on the discriminative function that maps the latent codes into labels. The result of this choice is a Gaussian process latent variable model (GPLVM) (Lawrence, 2005) for the labels. This not only allows us to flexibly represent a wide class of classifiers, but also prevents us from having to commit to a particular function at training time. We are then able to combine the efficient parametric feed-forward aspects of the autoencoder with a flexible Bayesian nonparametric model for the labels. This also leads to an interesting interpretation of the back-constrained GPLVM itself as a limiting case of an autoencoder in which the decoder has been marginalized out. In Section 4, we empirically examine our proposed approach on four data sets, including a real-world rehabilitation problem. We also examine a data set that highlights the value of our approach, in which we cannot only use guidance from desired labels, but also introduce guidance away from irrelevant representations.

2. Unsupervised Learning of Latent Representations

The nonparametrically-guided autoencoder presented in this paper is motivated largely by the relationship between two different approaches to latent variable modeling. In this section, we review these two approaches, the GPLVM and autoencoder neural network, and examine precisely how they are related.

2.1 Autoencoder Neural Networks

The autoencoder (Cottrell et al., 1987) is a neural network architecture that is designed to create a latent representation that is informative of the input data. Through training the model to reproduce the input data at its output, a latent embedding must arise within the hidden layer of the model. Its computations can intuitively be separated into two parts:

- An encoder, which maps the input into a latent (often lower-dimensional) representation.
- A decoder, which reconstructs the input through a map from the latent representation.

We will denote the latent space by $\mathcal{X}$ and the visible (data) space by $\mathcal{Y}$ and assume they are real valued with dimensionality J and K respectively, that is, $\mathcal{X} = \mathbb{R}^J$ and $\mathcal{Y} = \mathbb{R}^K$. The encoder, then, is defined as a function $g(y\,;\phi) : \mathcal{Y} \to \mathcal{X}$ and the decoder as $f(x\,;\psi) : \mathcal{X} \to \mathcal{Y}$. Given N data examples $\mathcal{D} = \{y^{(n)}\}_{n=1}^{N}$, $y^{(n)} \in \mathcal{Y}$, we jointly optimize the parameters of the encoder ϕ and decoder ψ over the least-squares reconstruction cost:

$$\phi^\star, \psi^\star = \arg\min_{\phi,\psi} \sum_{n=1}^{N} \sum_{k=1}^{K} (y_k^{(n)} - f_k(g(y^{(n)};\phi);\psi))^2, \tag{1}$$

where $f_k(\cdot)$ is the kth output dimension of $f(\cdot)$. It is easy to demonstrate that this model is equivalent to principal components analysis when f and g are linear projections. However, nonlinear basis functions allow for a more powerful nonlinear mapping. In our empirical analysis we use sigmoidal

$$g(y;\phi) = (1 + \exp(-y_+^T\phi))^{-1}$$

and noisy rectified linear

$$g(y;\phi) = \max\{0, y_+^T\phi + \varepsilon\}, \qquad\qquad \varepsilon \sim \mathcal{N}(0,1)$$

basis functions for the encoder where y_+ denotes y with a 1 appended to account for a bias term. The noisy rectified linear units or NReLU (Nair and Hinton, 2010)) exhibit the property that they are more *equivariant* to the scaling of the inputs (the non-noisy version being perfectly equivariant when the bias term is fixed to 0). This is a useful property for image data, for example, as (in contrast to sigmoidal basis functions) global lighting changes will cause uniform changes in the activations across hidden units.

Recently, autoencoders have regained popularity as they have been shown to be an effective module for "greedy pre-training" of deep neural networks (Bengio et al., 2007). *Denoising* autoencoders (Vincent et al., 2008) are of particular interest, as they are robust to the trivial "identity" solutions that can arise when trying to learn overcomplete representations. Overcomplete representations, which are of higher dimensionality than the input, are considered to be ideal for discriminative tasks. However, these are difficult to learn because a trivial minimum of the autoencoder reconstruction objective is reached when the autoencoder learns the identity transformation. The denoising autoencoder forces the model to learn more interesting structure from the data by providing as input a corrupted training example, while evaluating reconstruction on the noiseless original. The objective of Equation (1) then becomes

$$\phi^\star, \psi^\star = \arg\min_{\phi,\psi} \sum_{n=1}^{N} \sum_{k=1}^{K} (y_k^{(n)} - f_k(g(\widetilde{y}^{(n)};\phi);\psi))^2,$$

where $\widetilde{y}^{(n)}$ is the corrupted version of $y^{(n)}$. Thus, in order to infer missing components of the input or fix the corruptions, the model must extract a richer latent representation.

2.2 Gaussian Process Latent Variable Models

While the denoising autoencoder learns a latent representation that is distributed over the hidden units of the model, an alternative strategy is to consider that the data intrinsically lie on a lower-dimensional latent *manifold* that reflects their statistical structure. Such a manifold is difficult to define *a priori*, however, and thus the problem is often framed as learning the latent embedding under an assumed smooth functional mapping between the visible and latent spaces. Unfortunately, a major challenge arising from this strategy is the simultaneous optimization of the latent embedding and the functional parameterization. The Gaussian process latent variable model (Lawrence, 2005) addresses this challenge under a Bayesian probabilistic formulation. Using a Gaussian process prior, the GPLVM marginalizes over the infinite possible mappings from the latent to visible spaces and optimizes the latent embedding over a distribution of mappings. The GPLVM results in a powerful nonparametric model that analytically integrates over the infinite number of functional parameterizations from the latent to the visible space.

Similar to the autoencoder, linear kernels in the GPLVM recover principal components analysis. Under a nonlinear basis, however, the GPLVM can represent an arbitrarily complex continuous mapping, depending on the functions supported by the Gaussian process prior. Although GPLVMs were initially introduced for the visualization of high dimensional data, they have been used to obtain state-of-the-art results for a number of tasks, including modeling human motion (Wang et al., 2008), classification (Urtasun and Darrell, 2007) and collaborative filtering (Lawrence and Urtasun, 2009).

The GPLVM assumes that the N data examples $\mathcal{D} = \{y^{(n)}\}_{n=1}^{N}$ are the image of a homologous set $\{x^{(n)}\}_{n=1}^{N}$ arising from a vector-valued "decoder" function $f(x) : \mathcal{X} \to \mathcal{Y}$. Analogously to the squared-loss of the previous section, the GPLVM assumes that the observed data have been corrupted by zero-mean Gaussian noise: $y^{(n)} = f(x^{(n)}) + \varepsilon$ with $\varepsilon \sim \mathcal{N}(0, \sigma^2 \mathbb{I}_K)$. The innovation of the GPLVM is to place a Gaussian process prior on the function $f(x)$ and then optimize the latent representation $\{x^{(n)}\}_{n=1}^{N}$, while marginalizing out the unknown $f(x)$.

2.2.1 GAUSSIAN PROCESS PRIORS

Rather than requiring a specific finite basis, the Gaussian process provides a distribution over random functions of a particular family, the properties of which are specified via a positive definite covariance function. Typically, Gaussian processes are defined in terms of a distribution over scalar functions and in keeping with the convention for the GPLVM, we shall assume that K independent GPs are used to construct the vector-valued function $f(x)$. We denote each of these functions as $f_k(x) : \mathcal{X} \to \mathbb{R}$. The GP requires a covariance kernel function, which we denote as

$C(x,x') : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$. The defining characteristic of the GP is that for any finite set of N data in $\mathcal{X}$ there is a corresponding N-dimensional Gaussian distribution over the function values, which in the GPLVM we take to be the components of $\mathcal{Y}$. The $N\times N$ covariance matrix of this distribution is the matrix arising from the application of the covariance kernel to the N points in $\mathcal{X}$. We denote any additional parameters governing the behavior of the covariance function by θ.

Under the component-wise independence assumptions of the GPLVM, the Gaussian process prior allows one to analytically integrate out the K latent scalar functions from $\mathcal{X}$ to $\mathcal{Y}$. Allowing for each of the K Gaussian processes to have unique hyperparameter θ_k, we write the marginal likelihood, that is, the probability of the observed data given the hyperparameters and the latent representation, as

$$p(\{y^{(n)}\}_{n=1}^{N} \,|\, \{x^{(n)}\}_{n=1}^{N}, \{\theta_k\}_{k=1}^{K}, \sigma^2) = \prod_{k=1}^{K} \mathcal{N}(y_k^{(\cdot)} \,|\, 0, \Sigma_{\theta_k} + \sigma^2 \mathbb{I}_N),$$

where $y_k^{(\cdot)}$ refers to the vector $[y_k^{(1)}, \ldots, y_k^{(N)}]$ and where Σ_{θ_k} is the matrix arising from $\{x_n\}_{n=1}^{N}$ and θ_k. In the basic GPLVM, the optimal x_n are found by maximizing this marginal likelihood.

2.2.2 Covariance Functions

Here we will briefly describe the covariance functions used in this work. For a more thorough treatment, we direct the reader to Rasmussen and Williams (2006, Chapter 4).

A common choice of covariance function for the GP is the automatic relevance determination (ARD) *exponentiated quadratic* (also known as *squared exponential*) kernel

$$K_{\mathsf{EQ}}(\mathrm{x}, \mathrm{x}') = \exp\left\{-\frac{1}{2} r^2(\mathrm{x}, \mathrm{x}')\right\}, \qquad r^2(\mathrm{x}, \mathrm{x}') = (\mathrm{x} - \mathrm{x}')^{\mathsf{T}} \Psi (\mathrm{x} - \mathrm{x}')$$

where the covariance between outputs of the GP depends on the distance between corresponding inputs. Here Ψ is a symmetric positive definite matrix that defines the metric (Vivarelli and Williams, 1999). Typically, Ψ is a diagonal matrix with $\Psi_{d,d} = 1/\ell_d^2$, where the *length-scale* parameters, ℓ_d, scale the contribution of each dimension of the input independently. In the case of the GPLVM, these parameters are made redundant as the inputs themselves are learned. Thus, in this work we assume these kernel hyperparameters are set to a fixed value.

The exponentiated quadratic construction is not appropriate for all functions. Consider a function that is periodic in the inputs. The covariance between outputs should then depend not on the Euclidian distance between inputs but rather on their phase. A solution is to warp the inputs to capture this property and then apply the exponentiated quadratic in this warped space. To model a periodic function, MacKay (1998) suggests applying the exponentiated quadratic covariance to the output of an embedding function $u(x)$, where for a single dimensional input x, $u(x) = [\sin(x), \cos(x)]$ expands from $\mathbb{R}$ to $\mathbb{R}^2$. The resulting *periodic* covariance becomes

$$K_{\mathsf{PER}}(x, x') = \exp\left\{-\frac{2\sin^2\frac{x-x'}{2}}{\ell^2}\right\}.$$

The exponentiated quadratic covariance can be shown (MacKay, 1998) to be the similarity between inputs after they are projected into a feature space by an infinite number of centered radial basis functions. Williams (1998) derived a kernel that similarly, under a specific activation function, reflects a feature projection by a neural network in the limit of infinite units. This results in the *neural network* covariance

$$K_{\mathsf{NN}}(\mathrm{x},\mathrm{x}') = \frac{2}{\pi}\sin^{-1}\left(\frac{2\tilde{\mathrm{x}}^T\Psi\tilde{\mathrm{x}}'}{\sqrt{(1+2\tilde{\mathrm{x}}^T\Psi\tilde{\mathrm{x}}')(1+2\tilde{\mathrm{x}}^T\Psi\tilde{\mathrm{x}}')}}\right), \tag{2}$$

where $\tilde{\mathrm{x}}$ is x with a 1 prepended. An important distinction from the exponentiated quadratic is that the neural net covariance is non-stationary. Unlike the exponentiated quadratic, the neural network covariance is not invariant to translation. We use the neural network covariance primarily to draw a theoretical connection between GPs and autoencoders. However, the non-stationary properties of this covariance in the context of the GPLVM, which can allow the GPLVM to capture more complex structure, warrant further investigation.

2.2.3 THE BACK-CONSTRAINED GPLVM

Although the GPLVM constrains the mapping from the latent space to the data to be smooth, it does not enforce smoothness in the inverse mapping. This can be an undesirable property, as data that are intrinsically close in observed space need not be close in the latent representation. Not only does this introduce arbitrary gaps in the latent manifold, but it also complicates the encoding of novel data points into the latent space as there is no direct mapping. The latent representations of out-of-sample data must thus be optimized, conditioned on the latent embedding of the training examples. Lawrence and Quiñonero-Candela (2006) reformulated the GPLVM to address these issues, with the constraint that the hidden representation be the result of a smooth map from the observed space. They proposed multilayer perceptrons and radial-basis-function networks as possible implementations of this smooth mapping. We will denote this "encoder" function, parameterized by ϕ, as $g(y;\phi):\mathcal{Y}\rightarrow\mathcal{X}$. The marginal likelihood objective of this *back-constrained* GPLVM can now be formulated as finding the optimal ϕ under:

$$\phi^\star = \arg\min_{\phi}\sum_{k=1}^{K}\ln|\Sigma_{\theta_k,\phi}+\sigma^2\mathbb{I}_N| + y_k^{(\cdot)^{\mathsf{T}}}(\Sigma_{\theta_k,\phi}+\sigma^2\mathbb{I}_N)^{-1}y_k^{(\cdot)}, \tag{3}$$

where the kth covariance matrix $\Sigma_{\theta_k,\phi}$ now depends not only on the kernel hyperparameters θ_k, but also on the parameters of $g(y;\phi)$, that is,

$$[\Sigma_{\theta_k,\phi}]_{n,n'} = C(g(y^{(n)};\phi),g(y^{(n')};\phi);\theta_k). \tag{4}$$

Lawrence and Quiñonero-Candela (2006) motivate the back-constrained GPLVM partially through the NeuroScale algorithm of Lowe and Tipping (1997). The NeuroScale algorithm is a radial basis function network that creates a one-way mapping from data to a latent space using a heuristic loss that attempts to preserve pairwise distances between data cases. Thus, the back-constrained GPLVM can be viewed as a combination of NeuroScale and the GPLVM where the pairwise distance loss is removed and rather the loss is backpropagated from the GPLVM.

2.3 GPLVM as an Infinite Autoencoder

The relationship between Gaussian processes and artificial neural networks was established by Neal (1996), who showed that the prior over functions implied by many parametric neural networks becomes a GP in the limit of an infinite number of hidden units. Williams (1998) subsequently derived a GP covariance function corresponding to such an infinite neural network (Equation 2) with a specific activation function.

An interesting and overlooked consequence of this relationship is that it establishes a connection between autoencoders and the back-constrained Gaussian process latent variable model. A GPLVM with the covariance function of Williams (1998), although it does not impose a density over the data, is similar to a density network (MacKay, 1994) with an infinite number of hidden units in the single hidden layer. We can transform this density network into a semiparametric autoencoder by applying a neural network as the backconstraint network of the GPLVM. The encoder of the resulting model is a parametric neural network and the decoder a Gaussian process.

We can alternatively derive this model starting from an autoencoder. With a least-squares reconstruction cost and a linear decoder, one can integrate out the weights of the decoder assuming a zero-mean Gaussian prior over the weights. This results in a Gaussian process for the decoder and learning thus corresponds to the minimization of Equation (3) with a linear kernel for Equation (4). Incorporating any non-degenerate positive definite kernel, which corresponds to a decoder of infinite size, also recovers the general back-constrained GPLVM algorithm.

This infinite autoencoder exhibits some attractive properties. After training, the decoder network of an autoencoder is generally superfluous. Learning a parametric form for this decoder is thus a nuisance that complicates the objective. The infinite decoder network, as realized by the GP, obviates the need to learn a parameterization and instead marginalizes over all possible decoders. The parametric encoder offers a rapid encoding and persists as the training data changes, permitting, for example, stochastic gradient descent. A disadvantage, however, is that the decoder naturally inherits the computational costs of the GP by memorizing the data. Thus, for very high dimensional data, a standard autoencoder may be more desirable.

3. Supervised Guidance of Latent Representations

Unsupervised learning has proven to be effective for learning latent representations that excel in discriminative tasks. However, when the salient statistics of the data are only weakly informative about a desired discriminative task, it can be useful to incorporate label information into unsupervised learning. Bengio et al. (2007) demonstrated, for example, that while a purely supervised signal can lead to overfitting, mild supervised guidance can be beneficial when initializing a discriminative deep neural network. Therefore, Bengio et al. (2007) proposed a hybrid approach under which the unsupervised model's latent representation also be trained to predict the label information, by adding a parametric mapping $c(x;\Lambda) : \mathcal{X} \rightarrow \mathcal{Z}$ from the latent space $\mathcal{X}$ to the labels $\mathcal{Z}$ and backpropagating error gradients from the output. Bengio et al. (2007) used a linear logistic regression classifier for this parametric mapping. This "partial supervision" thus encourages the model to encode statistics within the latent representation that are useful for a specific (but learned) parameterization of such a linear mapping. Ranzato and Szummer (2008) adopted a similar strategy to learn compact representations of documents.

There are disadvantages to this approach. The assumption of a specific parametric form for the mapping $c(x;\Lambda)$ restricts the supervised guidance to classifiers within that family of mappings. Also, the learned representation is committed to one particular setting of the parameters Λ. Consider the learning dynamics of gradient descent optimization for this strategy. At every iteration t of descent (with current state $\phi_t, \psi_t, \Lambda_t$), the gradient from supervised guidance encourages the latent representation (currently parametrized by ϕ_t, ψ_t) to become more predictive of the labels under the current label map $c(x;\Lambda_t)$. Such behavior discourages moves in ϕ,ψ space that make the latent representation more predictive under some other label map $c(x;\Lambda^\star)$ where $\Lambda^\star$ is potentially distant from Λ_t. Hence, while the problem would seem to be alleviated by the fact that Λ is learned jointly, this constant pressure towards representations that are immediately useful increases the difficulty of learning the unsupervised component.

3.1 Nonparametrically Guided Autoencoder

Instead of specifying a particular discriminative regressor for the supervised guidance and jointly optimizing for its parameters and those of an autoencoder, it seems more desirable to enforce only that a mapping to the labels exists while optimizing for the latent representation. That is, rather than learning a latent representation that is tied to a specific parameterized mapping to the labels, we would instead prefer to find a latent representation that is consistent with an entire class of mappings. One way to arrive at such a guidance mechanism is to marginalize out the parameters Λ of a label map $c(x;\Lambda)$ under a distribution that permits a wide family of functions. We have seen previously that this can be done for reconstructions of the input space with a decoder $f(x;\psi)$. We follow the same reasoning and do this instead for $c(x;\Lambda)$. Integrating out the parameters of the label map yields a back-constrained GPLVM acting on the label space $\mathcal{Z}$, where the back constraints are determined by the input space $\mathcal{Y}$. The positive definite kernel specifying the Gaussian process then determines the properties of the distribution over mappings from the latent representation to the labels. The result is a hybrid of the autoencoder and back-constrained GPLVM, where the encoder is shared across models. For notation, we will refer to this approach to guided latent representation as a *nonparametrically guided autoencoder*, or NPGA.

Let the label space $\mathcal{Z}$ be an M-dimensional real space,[1] that is, $\mathcal{Z}=\mathbb{R}^M$, and the nth training example has a label vector $z^{(n)}\in\mathcal{Z}$. The covariance function that relates label vectors in the NPGA is

$$[\Sigma_{\theta_m,\phi,\Gamma}]_{n,n'} = C(\Gamma\cdot g(y^{(n)};\phi),\Gamma\cdot g(y^{(n')};\phi);\theta_m),$$

where $\Gamma\in\mathbb{R}^{H\times J}$ is an H-dimensional linear projection of the encoder output. For $H \ll J$, this projection improves efficiency and reduces overfitting. Learning in the NPGA is then formulated as finding the optimal ϕ,ψ,Γ under the combined objective:

$$\phi^\star,\psi^\star,\Gamma^\star = \arg\min_{\phi,\psi,\Gamma}(1-\alpha)L_{\text{auto}}(\phi,\psi)+\alpha L_{\text{GP}}(\phi,\Gamma)$$

where $\alpha\in[0,1]$ linearly blends the two objectives

$$L_{\text{auto}}(\phi,\psi) = \frac{1}{K}\sum_{n=1}^{N}\sum_{k=1}^{K}(y_k^{(n)} - f_k(g(y^{(n)};\phi);\psi))^2,$$

$$L_{\text{GP}}(\phi,\Gamma) = \frac{1}{M}\sum_{m=1}^{M}\left[\ln|\Sigma_{\theta_m,\phi,\Gamma}+\sigma^2\mathbb{I}_N| + z_m^{(\cdot)\mathsf{T}}(\Sigma_{\theta_m,\phi,\Gamma}+\sigma^2\mathbb{I}_N)^{-1}z_m^{(\cdot)}\right].$$

We use a linear decoder for $f(x;\psi)$, and the encoder $g(y;\phi)$ is a linear transformation followed by a fixed element-wise nonlinearity. As is common for autoencoders and to reduce the number of free parameters in the model, the encoder and decoder weights are tied. As proposed in the

[1] For discrete labels, we use a "one-hot" encoding.

denoising autoencoder variant of Vincent et al. (2008), we always add noise to the encoder inputs in cost $L_{\text{auto}}(\phi, \psi)$, keeping the noise fixed during each iteration of learning. That is, we update the denoising autoencoder noise every three iterations of conjugate gradient descent optimization. For the larger data sets, we divide the training data into mini-batches of 350 training cases and perform three iterations of conjugate gradient descent per mini-batch. The optimization proceeds sequentially over the batches such that model parameters are updated after each mini-batch.

3.2 Related Models

An example of the hybridization of an unsupervised connectionist model and Gaussian processes has been explored in previous work. Salakhutdinov and Hinton (2008) used restricted Boltzmann machines (RBMs) to initialize a multilayer neural network mapping into the covariance kernel of a Gaussian process regressor or classifier. They then adjusted the mapping through backpropagating gradients from the Gaussian process through the neural network. In contrast to the NPGA, this model did not use a Gaussian process in the initial learning of the latent representation and relies on a Gaussian process for inference at test time. Unfortunately, this poses significant practical issues for large data sets such as NORB or CIFAR-10, as the computational complexity of GP inference is cubic in the number of data examples. Note also that when the salient variations of the data are not relevant to a given discriminative task, the initial RBM training will not encourage the encoding of the discriminative information in the latent representation. The NPGA circumvents these issues by applying a GP to small mini-batches during the learning of the latent representation and uses the GP to learn a representation that is better even for a linear discriminative model.

Previous work has merged parametric unsupervised learning and nonparametric supervised learning. Salakhutdinov and Hinton (2007) combined autoencoder training with neighborhood component analysis (Goldberger et al., 2004), which encouraged the model to encode similar latent representations for inputs belonging to the same class. Hadsell et al. (2006) employ a similar objective in a fully supervised setting to preserve distances in label space in a latent representation. They used this method to visualize the different latent embeddings that can arise from using additional labels on the NORB data set. Note that within the NPGA, the backconstrained-GPLVM performs an analogous role. In Equation 3, the first term, the log determinant of the kernel, regularizes the latent space. Since the determinant is minimized when the covariance between all pairs is maximized, it pulls all examples together in the latent space. The second term, however, pushes examples that are distant in label space apart in the latent space. For example, when a one-hot coding is used, the labels act as indicator variables reflecting same-class pairs in the concentration matrix. This pushes apart examples that are of different class and pulls together examples of the same class. Thus, the GPLVM enforces that examples close in label space will be closer in the latent representation than examples that are distant in label space.

There are several important differences, however, between the aforementioned approaches and the NPGA. First, the NPGA can be intuitively interpreted as using a marginalization over mappings to labels. Second, the NPGA naturally accommodates continuous labels and enables the use of any covariance function within the wide library from the Gaussian process literature. Incorporating periodic labels, for example, is straightforward through using a periodic covariance. Encoding such periodic signals in a parametric neural network and blending this with unsupervised learning can be challenging (Zemel et al., 1995). Similarly to a subset of the aforementioned work, the NPGA exhibits the property that it not only enables the learning of latent representations that encode information that is relevant for discrimination but as we show in Section 4.3, it can *ignore* salient information that is not relevant to the discriminative representation.

Although it was originally developed as a model for unsupervised dimensionality reduction, a number of approaches have explored the addition of auxiliary signals within the GPLVM. The Discriminative GPLVM (Urtasun and Darrell, 2007), for example, added a discriminant analysis based prior that enforces inter-class separability in the latent space. The DGPLVM is, however, restricted to discrete labels, requires that the latent dimensionality be smaller than the number of classes and uses a GP mapping to the data, which is computationally prohibitive for high dimensional data. A GPLVM formulation in which multiple GPLVMs mapping to different signals share a single latent space, the shared GPLVM (SGPLVM), was introduced by Shon et al. (2005). Wang et al. (2007) showed that using product kernels within the context of the GPLVM results in a generalisation of multilinear models and allows one to separate the encoding of various signals in the latent representation. As discussed above, the reliance on a Gaussian process mapping to the data prohibits the application of these approaches to large and high dimensional data sets. Our model overcomes these limitations through using a natural parametric form of the GPLVM, the autoencoder, to map to the data.

4. Empirical Analyses

We now present experiments with NPGA on four different classification data sets. In all experiments, the discriminative value of the learned representation is evaluated by training a linear (logistic) classifier, a standard practice for evaluating latent representations.

4.1 Oil Flow Data

We begin our emprical analysis by exploring the benefits of using the NPGA on a multi-phase oil flow classification problem (Bishop and James, 1993). The data are twelve-dimensional, real-valued gamma densitometry measurements from a simulation of multi-phase oil flow. The relatively small sample size of these data—1,000 training and 1,000 test examples—makes this problem useful for exploring different models and training procedures. We use these data primarily to explore two questions:

- To what extent does the nonparametric guidance of an unsupervised parametric autoencoder improve the learned feature representation with respect to the classification objective?
- What additional benefit is gained through using nonparametric guidance over simply incorporating a parametric mapping to the labels?

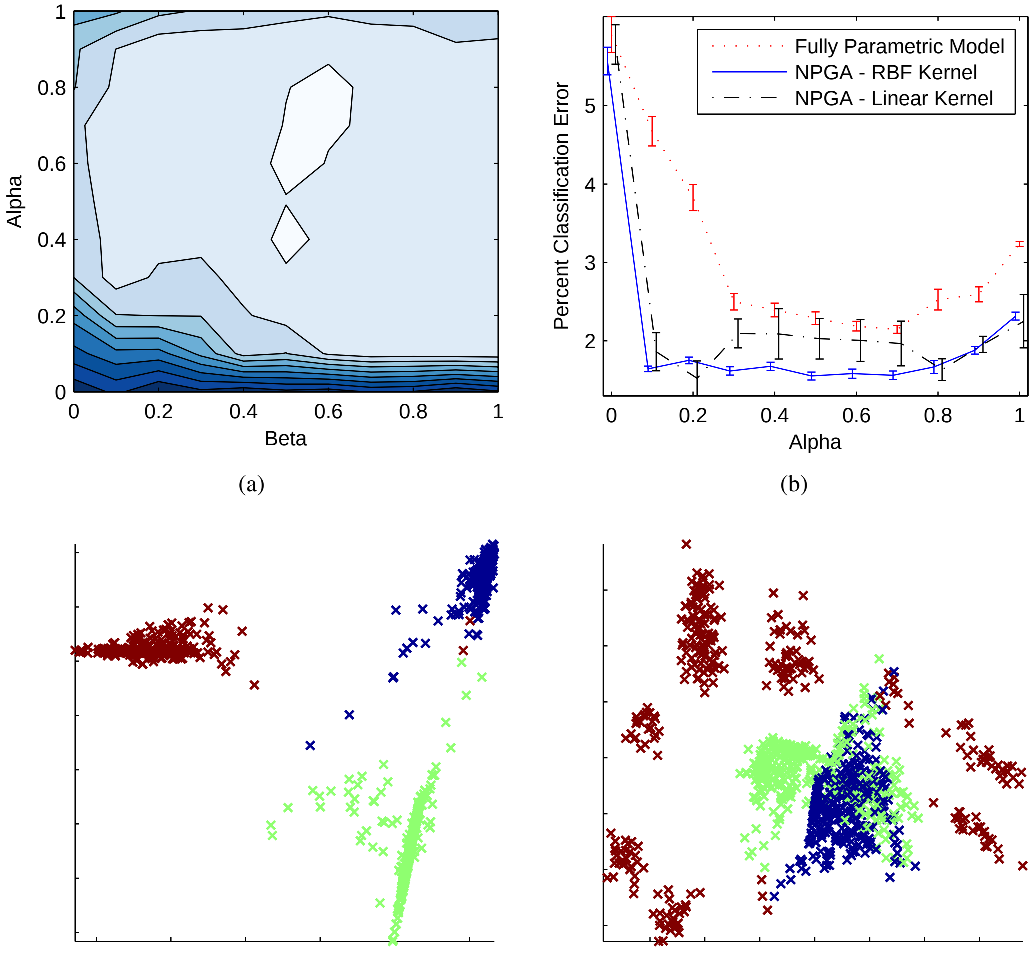
In order to address these concerns, we linearly blend our nonparametric guidance cost $L_{\text{GP}}(\phi,\Gamma)$ with the one Bengio et al. (2007) proposed, referred to as $L_{\text{LR}}(\phi,\Lambda)$:

$$L(\phi,\psi,\Lambda,\Gamma;\alpha,\beta) = (1-\alpha)L_{\text{auto}}(\phi,\psi) + \alpha((1-\beta)L_{\text{LR}}(\phi,\Lambda) + \beta L_{\text{GP}}(\phi,\Gamma)), \tag{5}$$

where $\beta \in [0,1]$ and Λ are the parameters of a multi-class logistic regression mapping to the labels.

Thus, α allows us to adjust the relative contribution of the unsupervised guidance while β weighs the relative contributions of the parametric and nonparametric supervised guidance.

To assess the benefit of the nonparametric guidance, we perform a grid search over the range of settings for α and β at intervals of 0.1. For each of these intervals, a model was trained for 100



(a)

(b)

(c)

(d)

Figure 1: We explore the benefit of the NPGA on the oil data through adjusting the relative contributions of the autoencoder, logistic regressor and GP costs in the hybrid objective by modifying α and β. (a) Classification error on the test set on a linear scale from 6% (dark) to 1% (light) (b) Cross-sections of (a) at $\beta = 0$ (a fully parametric model) and $\beta = 1$ (NPGA). (c & d) Latent projections of the 1000 test cases within the two dimensional latent space of the GP, Γ, for a NPGA ($\alpha = 0.5$) and a back-constrained GPLVM.

iterations of conjugate gradient descent and classification performance was assessed by applying logistic regression on the hidden units of the encoder. 250 NRenLU units were used in the encoder, and zero-mean Gaussian noise with a standard deviation of 0.05 was added to the inputs of the denoising autoencoder cost. The GP label mapping used an RBF covariance with $H = 2$. To make the problem more challenging, a subset of 100 training samples was used. Each experiment was repeated over 20 different random initializations.

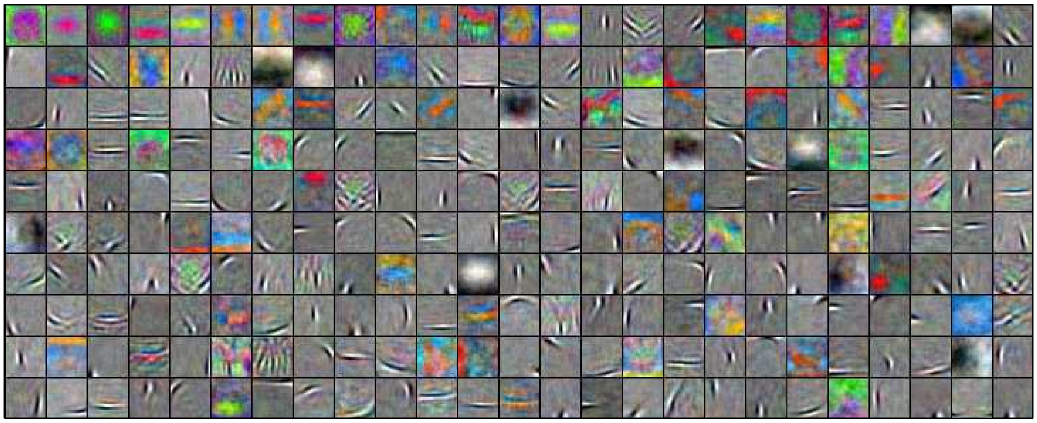


Figure 2: A sample of filters learned on the CIFAR 10 data set. This model achieved a test accuracy of 65.71%. The filters are sorted by norm.

The results of this analysis are visualized in Figure 1. Figure 1b demonstrates that, even when compared with direct optimization under the discriminative family that will be used at test time (logistic regression), performance improves by integrating out the label map. However, in Figure 1a we can see that some parametric guidance can be beneficial, presumably because it is from the same discriminative family as the final classifier. A visualisation of the latent representation learned by an NPGA and a standard back-constrained GPLVM is provided in Figures 1c and 1d. The former clearly embeds much more class-relevant structure than the latter.

We observe also that using a GP with a linear covariance function within the NPGA outperforms the parametric guidance (see Fig. 1b). While the performance of the model does depend on the choice of kernel, this helps to confirm that the benefit of our approach is achieved mainly through integrating out the label mapping, rather than having a more powerful nonlinear mapping to the labels. Another interesting result is that the results of the linear covariance NPGA are significantly noisier than the RBF mapping. Presumably, this is due to the long-range global support of the linear covariance causing noisier batch updates.

4.2 CIFAR 10 Image Data

We also apply the NPGA to a much larger data set that has been widely studied in the connectionist learning literature. The CIFAR 10 object classification data set[2] is a labeled subset of the *80 million tiny images* data (Torralba et al., 2008) with a training set of 50,000 32×32 color images and a test set of an additional 10,000 images. The data are labeled into ten classes. As GPs scale poorly on large data sets we consider it pertinent to explore the following:

> Are the benefits of nonparametric guidance still observed in a larger scale classification problem, when mini-batch training is used?

To answer this question, we evaluate the use of nonparametric guidance on three different combinations of preprocessing, architecture and convolution. For each experiment, an autoencoder is

[2] CIFAR data set at `http://www.cs.utoronto.ca/~kriz/cifar.html`.

Experiment	α	Accuracy
1. Full Images	0.0	46.91%
	0.1	56.75%
	0.5	52.11%
	1.0	45.45%
2. 28x28 Patches	0.0	63.20%
	0.8	65.71%
3. Convolutional	0.0	73.52%
	0.1	75.82%
Sparse Autoencoder (Coates et al., 2011)		73.4%
Sparse RBM (Coates et al., 2011)		72.4%
K-means (Hard) (Coates et al., 2011)		68.6%
K-means (Triangle) (Coates et al., 2011)		77.9%

Table 1: Results on CIFAR 10 for various training strategies, varying the nonparametric guidance α. Recently published convolutional results are shown for comparison.

compared to a NPGA by modifying α. Experiments[3] were performed following three different strategies:

1. Full images: A one-layer autoencoder with 2400 NReLU units was trained on the raw data (which was reduced from $32\times32\times3 = 3072$ to 400 dimensions using PCA). A GP mapping to the labels operated on a $H = 25$ dimensional space.

2. 28×28 patches: An autoencoder with 1500 logistic hidden units was trained on $28\times28\times3$ patches subsampled from the full images, then reduced to 400 dimensions using PCA. All models were fine tuned using backpropagation with softmax outputs and predictions were made by taking the expectation over all patches (i.e., to classify an image, we consider all 28×28 patches obtained from that image and then average the label distributions over all patches). A $H=25$ dimensional latent space was used for the GP.

3. Convolutional: Following Coates et al. (2011), 6×6 patches were subsampled and each patch was normalized for lighting and contrast. This resulted in a $36\times3 = 108$ dimensional feature vector as input to the autoencoder. For classification, features were computed densely over all 6×6 patches. The images were divided into 4×4 blocks and features were pooled through summing the feature activations in each block. 1600 NReLU units were used in the autoencoder but the GP was applied to only 400 of them. The GP used a $H=10$ dimensional space.

[3] When PCA preprocessing was used for autoencoder training, the inputs were corrupted with zero-mean Gaussian noise with standard deviation 0.05. Otherwise, raw pixels were corrupted by deleting (i.e., set to zero) 10% of the pixels. Autoencoder training then corresponds to reconstructing the original input. Each model used a neural net (MLP) covariance with fixed hyperparameters.

After training, a logistic regression classifier was applied to the features resulting from the hidden layer of each autoencoder to evaluate their quality with respect to the classification objective. The results, presented in Table 1, show that supervised guidance helps in all three strategies. The use of different architectures, methodologies and hidden unit activations demonstrates that the nonparametric guidance can be beneficial for a wide variety of formulations. Although, we do not achieve state of the art results on this data set, these results demonstrate that nonparametric guidance is beneficial for a wide variety of model architechtures. We note that the optimal amount of guidance differs for each experiment and setting α too high can often be detrimental to performance. This is to be expected, however, as the amount of discriminative information available in the data differs for each experiment. The small patches in the convolutional strategy, for example, likely encode very weak discriminative information. Figure 2 visualises the encoder weights learned by the NPGA on the CIFAR data.

4.3 Small NORB Image Data

In the following empirical analysis, the use of the NPGA is explored on the small NORB data (LeCun et al., 2004). The data are stereo image pairs of fifty toys belonging to five generic categories. Each toy was imaged under six lighting conditions, nine elevations and eighteen azimuths. The 108×108 images were subsampled to half their size to yield a $48\times48\times2$ dimensional input vector per example. The objects were divided evenly into test and training sets yielding 24,300 examples each. The objective is to classify to which object category each of the test examples belongs.

This is an interesting problem as the variations in the data due to the different imaging conditions are the salient ones and will be the strongest signal learned by the autoencoder. This is nuisance structure that will influence the latent embedding in undesirable ways. For example, neighbors in the latent space may reflect lighting conditions in observed space rather than objects of the same class. Certainly, the squared pixel difference objective of the autoencoder will be affected more by significant lighting changes than object categories. Fortunately, the variations due to the imaging conditions are known *a priori*. In addition to an object category label, there are two real-valued vectors (elevation and azimuth) and one discrete vector (lighting type) associated with each example. In our empirical analysis we examine the following:

> As the autoencoder attempts to coalesce the various sources of structure into its hidden layer, can the NPGA guide the learning in such a way as to separate the class-invariant transformations of the data from the class-relevant information?

An NPGA was constructed with Gaussian processes mapping to each of the four label types to address this question. In order to *separate* the latent embedding of the salient information related to each label, the GPs were applied to disjoint subsets of the hidden units of the autoencoder. The autoencoder's 2400 NReLU units were partitioned such that half were used to encode structure relevant for classification and the other half were evenly divided to encode the remaining three labels. Thus a GP mapping from a four dimensional latent space, $H=4$, to class labels was applied to 1200 hidden units. GPs, with $H=2$, mapping to the three auxiliary labels were applied each to 400 hidden units. As the lighting labels are discrete, we used a one-hot coding, similarly to the class labels. The elevation labels are continuous, so the GP was mapped directly to the labels. Finally, because the azimuth is a periodic signal, a periodic kernel was used for the azimuth GP. This highlights a major advantage of our approach, as the broad library of GP covariance functions facilitate a flexibility to the mapping that would be challenging with a parametric model.

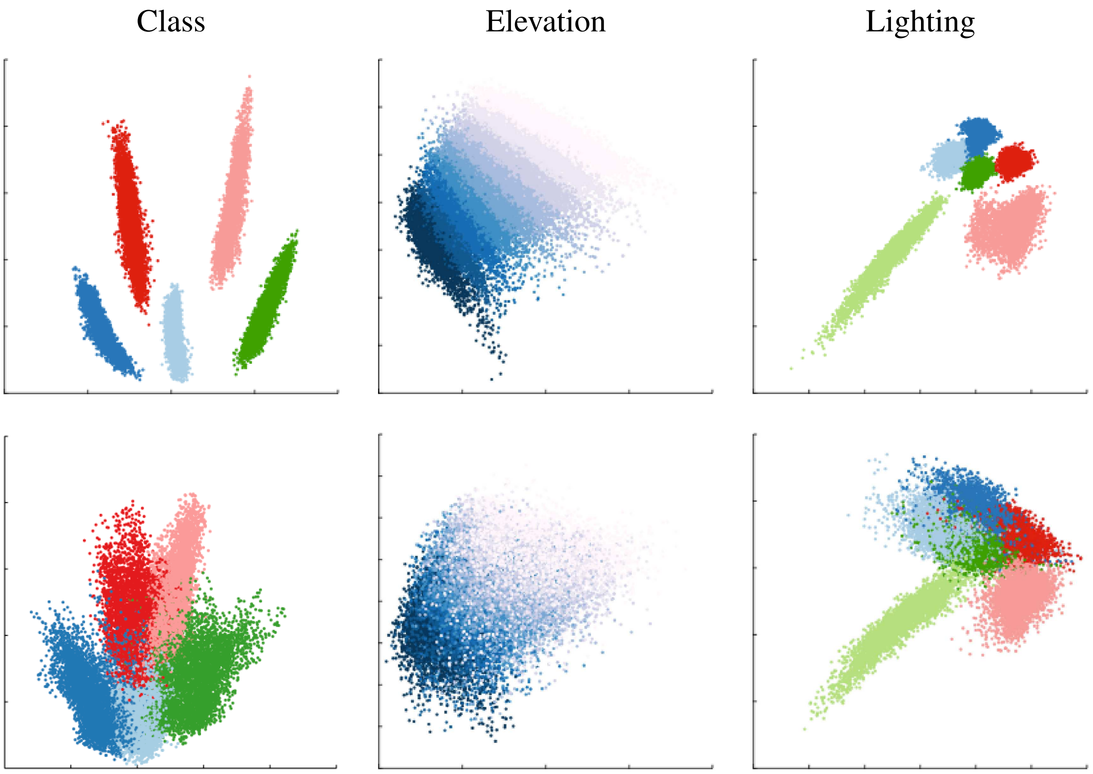


Figure 3: Visualisations of the NORB training (top) and test (bottom) data latent space representations in the NPGA, corresponding to class (first column), elevation (second column), and lighting (third column). The visualizations are in the GP latent space, Γ, of a model with $H = 2$ for each GP. Colors correspond to the respective labels.

To validate this configuration, we empirically compared it to a standard autoencoder (i.e., $\alpha = 0$), an autoencoder with parametric logistic regression guidance and an NPGA with a single GP applied to all hidden units mapping to the class labels. For comparison, we also provide results obtained by a back-constrained GPLVM and SGPLVM.[4] For all models, a validation set of 4300 training cases was withheld for parameter selection and early stopping. Neural net covariances were used for each GP except the one applied to azimuth, which used a periodic RBF kernel. GP hyperparameters were held fixed as their influence on the objective would confound the analysis of the role of α. For denoising autoencoder training, the raw pixels were corrupted by setting 20% of pixels to zero in the inputs. Each image was lighting- and contrast-normalized and the error on the test set was evaluated using logistic regression on the hidden units of each model. A visualisation of the structure learned by the GPs is shown in Figure 3. Results of the empirical comparison are presented in Table 2.

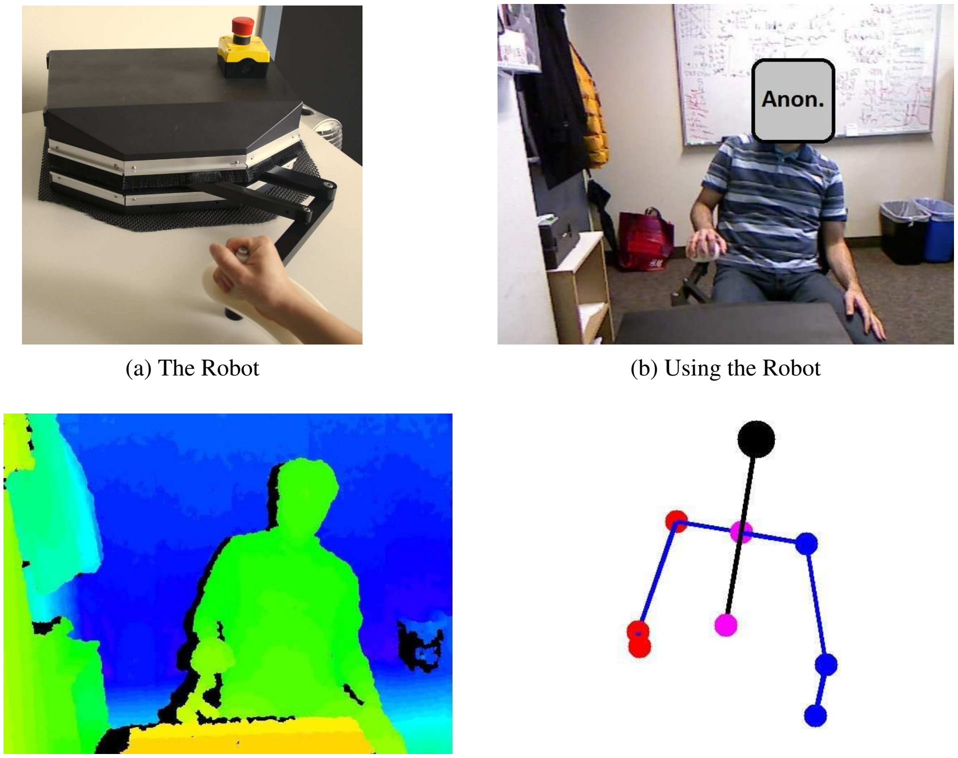
[4] The GPLVM and SGPLVM were applied to a 96 dimensional PCA of the data for computional tractability, used a neural net covariance mapping to the data, and otherwise used the same back-constraints, kernel configuration, and mini-batch training as the NPGA. The SGPLVM consisted of a GPLVM with a latent space that is shared by multiple GPLVM mappings to the data and each of the labels

Model	Accuracy
Autoencoder + 4(Log)reg ($\alpha = 0.5$)	85.97%
GPLVM	88.44%
SGPLVM (4 GPs)	89.02%
NPGA (4 GPs Lin – $\alpha = 0.5$)	92.09%
Autoencoder	92.75%
Autoencoder + Logreg ($\alpha = 0.5$)	92.91%
NPGA (1 GP NN – $\alpha = 0.5$)	93.03%
NPGA (1 GP Lin – $\alpha = 0.5$)	93.12%
NPGA (4 GPs Mix – $\alpha = 0.5$)	94.28%
K-Nearest Neighbors (LeCun et al., 2004)	83.4%
Gaussian SVM (Salakhutdinov and Larochelle, 2010)	88.4%
3 Layer DBN (Salakhutdinov and Larochelle, 2010)	91.69%
DBM: MF-FULL (Salakhutdinov and Larochelle, 2010)	92.77%
Third Order RBM (Nair and Hinton, 2009)	93.5%

Table 2: Experimental results on the small NORB data test set. Relevant published results are shown for comparison. NN, Lin and Mix indicate neural network, linear and a combination of neural network and periodic covariances respectively. Logreg indicates that a parametric logistic regression mapping to labels is blended with the autoencoder.

The NPGA model with four nonlinear kernel GPs significantly outperforms all other models, with an accuracy of 94.28%. This is to our knowledge the best (non-convolutional) result for a shallow model on this data set. The model indeed appears to separate the irrelevant transformations of the data from the structure relevant to the classification objective. In fact, a logistic regression classifier applied to only the 1200 hidden units on which the class GP was applied achieves a test error rate of 94.02%. This implies that the half of the latent representation that encodes the information to which the model should be invariant can be discarded with virtually no discriminative penalty. Given the significant difference in accuracy between this formulation and the other models, it appears to be very important to separate the encoding of different sources of variation within the autoencoder hidden layer.

The NPGA with four linear covariance GPs performed more poorly than the NPGA with a single linear covariance GP to class labels (92.09% compared to 93.03%). This interesting observation highlights the importance of using an appropriate mapping to each label. For example, it is unlikely that a linear covariance would be able to appropriately capture the structure of the periodic azimuth signal. An autoencoder with parametric guidance to all four labels, mimicking the configuration of the NPGA, achieved the poorest performance of the models tested, with 86% accuracy. This model incorporated two logistic and two Gaussian outputs applied to separate partitions of the hidden units. These results demonstrate the advantage of the GP formulation for supervised guidance, which gives the flexibility of choosing an appropriate kernel for different label mappings (e.g., a periodic kernel for the rotation label).



(a) The Robot

(b) Using the Robot

(c) Depth Image

(d) Skeletal Joints

Figure 4: The rehabilitation robot setup and sample data captured by the sensor.

4.4 Rehabilitation Data

As a final analysis, we demonstrate the utility of the NPGA on a real-world problem from the domain of assistive technology for rehabilitation. Rehabilitation patients benefit from performing repetitive rehabilitation exercises as frequently as possible but are limited due to a shortage of rehabilitation therapists. Thus, Kan et al. (2011), Huq et al. (2011), Lu et al. (2011) and Taati et al. (2012) developed a system to automate the role of a therapist guiding rehabilitation patients through repetitive upper limb rehabilitation exercises. The system allows users to perform upper limb reaching exercises using a robotic arm (see Figures 4a, 4b) while tailoring the amount of resistance to match the user's ability level. Such a system can alleviate the burden on therapists and significantly expedite rehabilitation for patients.

Critical to the effectiveness of the system is its ability to discriminate between various types of incorrect posture and prompt the user accordingly. The current system (Taati et al., 2012) uses a Microsoft Kinect sensor to observe a patient performing upper limb reaching exercises and records their posture as a temporal sequence of seven estimated upper body skeletal joint angles (see Figures 4c, 4d for an example depth image and corresponding pose skeleton captured by the system). A classifier is then employed to discriminate between five different classes of posture, consisting of good posture and four common forms of improper posture resulting from compensation due to limited agility. Taati et al. (2012) obtained a data set of seven users each performing each class of action at least once, creating a total of 35 sequences (23,782 frames). They compare the use of

Model	Accuracy
SVM^{Multiclass} Taati et al. (2012)	80.0%
Hidden Markov SVM Taati et al. (2012)	85.9%
ℓ_2-Regularized Logistic Regression	86.1%
NPGA ($\alpha = 0.8147, \beta = 0.3227, H = 3$, 242 hidden units)	91.7%

Table 3: Experimental results on the rehabilitation data. Per-frame classification accuracies are provided for different classifiers on the test set. Bayesian optimization was performed on a validation set to select hyperparameters for the ℓ_2-regularized logistic regression and the best performing NPGA algorithm.

a multiclass support vector machine and a hidden Markov support vector machine in a leave-one-subject-out test setting to distinguish these classes and report best per-frame classification accuracy rates of 80.0% and 85.9% respectively.

In our analysis of this problem we use a NPGA to encode a latent embedding of postures that facilitates better discrimination between different posture types. The same formulation as presented in Section 4.1 is applied here. We interpolate between a standard autoencoder ($\alpha = 0$), a classification neural net ($\alpha = 1, \beta = 1$), and a nonparametrically guided autoencoder by linear blending of their objectives according to Equation 5. Rectified linear units were used in the autoencoder. As in Taati et al. (2012), the input to the model is the seven skeletal joint angles, that is, $\mathcal{Y} = \mathbb{R}^7$, and the label space $\mathcal{Z}$ is over the five classes of posture.

In this setting, rather than perform a grid search for parameter selection as in Section 4.1, we optimize validation set error over the hyperparameters of the model using Bayesian optimization (Mockus et al., 1978). Bayesian optimization is a methodology for globally optimizing noisy, black-box functions based on the principles of Bayesian statistics. Particularly, given a prior over functions and a limited number of observations, Bayesian optimization explicitly models uncertainty over functional outputs and uses this to determine where to search for the optimum. For a more in-depth overview of Bayesian optimization see Brochu et al. (2010). We use the Gaussian process expected improvement algorithm with a Matern-$\frac{5}{2}$ covariance, as described in Snoek et al. (2012), to search over $\alpha \in [0,1]$, $\beta \in [0,1]$, $10-1000$ hidden units in the autoencoder and the GP latent dimensionality $H \in \{1..10\}$. The best validation set error observed by the algorithm, on the twelfth of thirty-seven iterations, was at $\alpha = 0.8147$, $\beta = 0.3227$, $H = 3$ and 242 hidden units. These settings correspond to a per-frame classification error rate of 91.70%, which is significantly higher than that reported by Taati et al. (2012). Results obtained using various models are presented in Table 3.

The relatively low number of experiments required by Bayesian optimization to find a state-of-the-art result implies that the validation error is a well behaved function of the various hyperparameters. The relationship between the model hyperparameters and validation error is challenging to visualize, but it is important to assess their relative effect on the performance of the model. Thus, in Figure 5 we explore how the relationship between validation error and the amount of nonparametric guidance α, and parametric guidance β is expected to change as the number of autoencoder hidden units is varied. That is, we show the expected value of the validation error for unobserved points under a Gaussian process regression. Similarly to the results observed in Section 4.1, it seems clear that the best region in hyperparameter space is a combination of all three objectives, the parametric

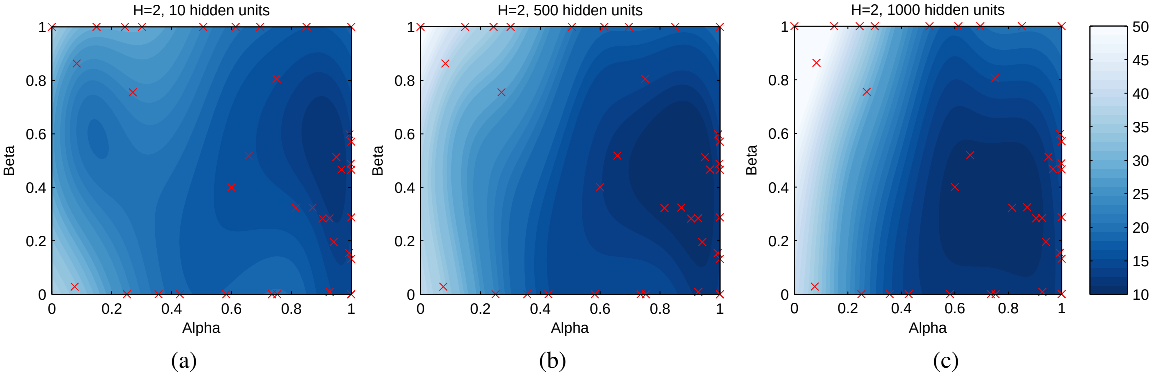


Figure 5: The posterior mean learned by Bayesian optimization over the validation set classification error (in percent) for α and β with H fixed at 2 and three different settings of autoencoder hidden units: (a) 10, (b) 500, and (c) 1000. This shows how the relationship between validation error and the amount of nonparametric guidance, α, and parametric guidance, β, is expected to change as the number of autoencoder hidden units is increased. The red x's indicate points that were explored by the Bayesian optimization routine.

guidance, nonparametric guidance and unsupervised learning. This reinforces the theory that although incorporating a parametric logistic regressor to the labels more directly reflects the ultimate goal of the model, it is more prone to overfit the training data than the GP. Also, as we increase the number of hidden units in the autoencoder, the amount of guidance required appears to decrease. As the capacity of the autoencoder is increased, it is likely that the autoencoder encodes increasingly subtle statistical structure in the data. When there are fewer hidden units, this structure is not encoded unless the autoencoder objective is augmented to reflect a preference for it. Interestingly, the validation error for $H = 2$ was significantly better than $H = 1$ but it did not appear to change significantly for $2 \geq H \leq 10$.

An additional interesting result is that, on this problem, the classification performance is worse when the nearest neighbors algorithm is used on the learned representation of the NPGA for discrimination. With the best performing NPGA reported above, a nearest neighbors classifier applied to the hidden units of the autoencoder achieved an accuracy of 85.05%. Adjusting β in this case also did not improve accuracy. This likely reflects the fact that the autoencoder must still encode information that is useful for reconstruction but not discrimination.

In this example, the resulting classfier must operate in real time to be useful for the rehabilitation task. The final product of our system is a simple softmax neural network, which is directly applicable to this problem. It is unlikely that a Gaussian process based classifier would be feasible in this context.

5. Conclusion

In this paper we present an interesting theoretical link between the autoencoder neural network and the back-constrained Gaussian process latent variable model. A particular formulation of the back-constrained GPLVM can be interpreted as an autoencoder in which the decoder has an infinite number of hidden units. This formulation exhibits some attractive properties as it allows one to learn the encoder half of the autoencoder while marginalizing over decoders. We examine the use of this model to guide the latent representation of an autoencoder to encode auxiliary label information without instantiating a parametric mapping to the labels. The resulting nonparametric *guidance* encourages the autoencoder to encode a latent representation that captures salient structure within the input data that is harmonious with the labels. Conceptually, this approach enforces simply that a smooth mapping exists from the latent representation to the labels rather than choosing or learning a specific parameterization. The approach is empirically validated on four data sets, demonstrating that the nonparametrically guided autoencoder encourages latent representations that are better with respect to a discriminative task. Code to run the NPGA is available at `http://hips.seas.harvard.edu/files/npga.tar.gz`. We demonstrate on the NORB data that this model can also be used to *discourage* latent representations that capture statistical structure that is known to be irrelevant through guiding the autoencoder to separate multiple sources of variation. This achieves state-of-the-art performance for a shallow non-convolutional model on NORB. Finally, in Section 4.4, we show that the hyperparameters introduced in this formulation can be optimized automatically and efficiently using Bayesian optimization. With these automatically selected hyperparameters the model achieves state of the art performance on a real-world applied problem in rehabilitation research.

References

Ryan P. Adams, Zoubin Ghahramani, and Michael I. Jordan. Tree-structured stick breaking for hierarchical data. In *Advances in Neural Information Processing Systems*, 2010.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems*, 2007.

Christopher M. Bishop and G. D. James. Analysis of multiphase flows using dual-energy gamma densitometry and neural networks. *Nuclear Instruments and Methods in Physics Research*, 1993.

Eric Brochu, Vlad M. Cora, and Nando de Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *pre-print*, 2010. arXiv:1012.2599.

Adam Coates, Honglak Lee, and Andrew Y. Ng. An analysis of single-layer networks in unsupervised feature learning. In *Conference on Artificial Intelligence and Statistics*, 2011.

Garrison W. Cottrell, Paul Munro, and David Zipser. Learning internal representations from grayscale images: An example of extensional programming. In *Conference of the Cognitive Science Society*, 1987.

Li Deng, Mike Seltzer, Dong Yu, Alex Acero, Abdel-Rahman Mohamed, and Geoffrey E. Hinton. Binary coding of speech spectrograms using a deep autoencoder. In *Interspeech*, 2010.

Jacob Goldberger, Sam Roweis, Geoff Hinton, and Ruslan Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems*, 2004.

Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

Rajibul Huq, Patricia Kan, Robby Goetschalckx, Debbie Hébert, Jesse Hoey, and Alex Mihailidis. A decision-theoretic approach in the design of an adaptive upper-limb stroke rehabilitation robot. In *International Conference of Rehabilitation Robotics (ICORR)*, 2011.

Patricia Kan, Rajibul Huq, Jesse Hoey, Robby Goestschalckx, and Alex Mihailidis. The development of an adaptive upper-limb stroke rehabilitation robotic system. *Neuroengineering and Rehabilitation*, 2011.

Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *International Conference on Machine Learning*, 2007.

Neil D. Lawrence. Probabilistic non-linear principal component analysis with Gaussian process latent variable models. *Journal of Machine Learning Research*, 6:1783–1816, 2005.

Neil D. Lawrence and Joaquin Quiñonero-Candela. Local distance preservation in the GP-LVM through back constraints. In *International Conference on Machine Learning*, 2006.

Neil D. Lawrence and Raquel Urtasun. Non-linear matrix factorization with Gaussian processes. In *International Conference on Machine Learning*, 2009.

Yann LeCun, Fu Jie Huang, and Léon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. *IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

David Lowe and Michael E. Tipping. Neuroscale: Novel topographic feature extraction using RBF networks. In *Advances in Neural Information Processing Systems*, 1997.

Elaine Lu, Rosalie Wang, Rajibul Huq, Don Gardner, Paul Karam, Karl Zabjek, Debbie Hébert, Jennifer Boger, and Alex Mihailidis. Development of a robotic device for upper limb stroke rehabilitation: A user-centered design approach. *Journal of Behavioral Robotics*, 2(4):176–184, 2011.

David J. C. MacKay. Introduction to Gaussian processes. *Neural Networks and Machine Learning*, 1998.

David J.C. MacKay. Bayesian neural networks and density networks. In *Nuclear Instruments and Methods in Physics Research, A*, pages 73–80, 1994.

Jonas Mockus, Vytautas Tiešis, and Antanas Žilinskas. The application of Bayesian methods for seeking the extremum. *Towards Global Optimization*, 2:117–129, 1978.

Vinod Nair and Geoffrey E. Hinton. 3D object recognition with deep belief nets. In *Advances in Neural Information Processing Systems*, 2009.

Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted Boltzmann machines. In *International Conference on Machine Learning*, 2010.

Radford Neal. Bayesian learning for neural networks. *Lecture Notes in Statistics*, 118, 1996.

Marc'Aurelio Ranzato and Martin Szummer. Semi-supervised learning of compact document representations with deep networks. In *International Conference on Machine Learning*, 2008.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006.

Ruslan Salakhutdinov and Geoffrey Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *Conference on Artificial Intelligence and Statistics*, 2007.

Ruslan Salakhutdinov and Geoffrey Hinton. Using deep belief nets to learn covariance kernels for Gaussian processes. In *Advances in Neural Information Processing Systems*, 2008.

Ruslan Salakhutdinov and Hugo Larochelle. Efficient learning of deep Boltzmann machines. In *Conference on Artificial Intelligence and Statistics*, 2010.

Aaron P. Shon, Keith Grochow, Aaron Hertzmann, and Rajesh P. N. Rao. Learning shared latent structure for image synthesis and robotic imitation. In *Advances in Neural Information Processing Systems*, 2005.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, 2012.

Babak Taati, Rosalie Wang, Rajibul Huq, Jasper Snoek, and Alex Mihailidis. Vision-based posture assessment to detect and categorize compensation during robotic rehabilitation therapy. In *International Conference on Biomedical Robotics and Biomechatronics*, 2012.

Antonio Torralba, Rob Fergus, and William T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1958–1970, 2008.

Raquel Urtasun and Trevor Darrell. Discriminative Gaussian process latent variable model for classification. In *International Conference on Machine Learning*, 2007.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine Learning*, 2008.

Francesco Vivarelli and Christopher K. I. Williams. Discovering hidden features with Gaussian process regression. In *Advances in Neural Information Processing Systems*, 1999.

Jack M. Wang, David J. Fleet, and Aaron Hertzmann. Multifactor Gaussian process models for style-content separation. In *International Conference on Machine Learning*, 2007.

Jack M. Wang, David J. Fleet, and Aaron Hertzmann. Gaussian process dynamical models for human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008.

Christopher K. I. Williams. Computation with infinite neural networks. *Neural Computation*, 10 (5):1203–1216, 1998.

Richard S. Zemel, Christopher K. I. Williams, and Michael C. Mozer. Lending direction to neural networks. *Neural Networks*, 8:503–512, 1995.