Learning a Manifold of Fonts
Supplemental Material

Neill D.F. Campbell[1] Jan Kautz[1,2]
[1]University College London [2]NVIDIA Research

A Fonts Used

Andale Mono	Andalus	Angsana New
Aparajita	Arial	Browallia New
Calibri	Cambria	Candara
Consolas	Constantia	Corbel
Cordia New	DaunPenh	David
Ebrima	Euphemia	Franklin Gothic Book
Gautami	Georgia	Gill Sans MT
Kalinga	Lao Sangam MN	Lucida Sans Unicode
MS PMincho	Microsoft Himalaya	Microsoft Sans Serif
Microsoft Tai Le	Miriam	Mongolian Baiti
MoolBoran	Narkisim	Nyala
PMingLiU	Palatino Linotype	Perpetua
Plantagenet Cherokee	Shonar Bangla	SimSun-ExtB
Sylfaen	Tahoma	Times New Roman
Traditional Arabic	Trebuchet MS	Vani
Verdana		

Table 1: *The fonts used in our experiments.*

We took the 'ttf' files from the fonts folder on a *'Windows 7'* installation and removed any non-latin or symbol based fonts. We also removed a few fonts that were topologically inconsistent over many letters, for example handwriting based fonts; if only a few letters are inconsistent, *e.g.*'g', we omitted the inconsistent character and embedded the font with missing data so that the manifold fills in unmatched character [Navaratnam et al. 2007]. Our training set includes examples of many different categories of typefaces (e.g. Humanist, Garalde, Didone/Modern, Slab, Lineale/Grotesque, and Glyphic) for a total of 46 fonts listed in Table 1.

B Details on the GP-LVM

Here we provide a more in-depth introduction to Gaussian Processes, illustrating their use for regression. We then extend this technique to present the Gaussian Process Latent Variable Model (GP-LVM) that we use to construct probabilistic manifolds. Finally, we provide the details about how to learn the true dimensionality of a manifold from the data.

B.1 Gaussian Processes

Gaussian Processes (GPs) are a non-parametric model that consist of distributions over functions. They can be used as a powerful way to model regression, that producing a smooth mapping function from an input space to an output space (both of which maybe multi-dimensional vector spaces); essentially they act as a prior that encourages functions to be smooth. A full treatment is beyond the scope of this work and instead we provide a basic illustrative example; there are many excellent texts on GPs including the definitive book by Rasmussen and Williams [2006]. We begin by considering using GPs for the problem of regression and then show how we may treat the problem of finding a manifold (or probabilistic dimensionality reduction) as a new take on the regression task, forming the GP-LVM.

Notation For the illustration of the regression task we will use the following notation. Suppose that we have a set of training data pairs $\{\mathbf{x}_j, \mathbf{y}_j\}$ that associate an input vector $\mathbf{x}_j \in \mathcal{R}^Q$ with its corresponding output vector $\mathbf{y}_j \in \mathcal{R}^D$ with respective dimensions of Q and D. If we need to consider multiple sets of these vectors we stack them together row-wise to produce a matrix, for example $X = [\cdots \mathbf{x}_j \cdots]^{\mathrm{T}}$ and $Y = [\cdots \mathbf{y}_j \cdots]^{\mathrm{T}}$.

GP Regression In the task of regression, we wish to identify a function $f(\cdot)$ that maps the input to the output space so that we can take a new input $\mathbf{x}^*$ and find the corresponding output as $\mathbf{y}^* = f(\mathbf{x}^*)$. Now, if we take a GP it has the property that any discrete set of samples taken from the GP will be normally distributed. Thus if the GP is evaluated at a series of values $X = [\mathbf{x}_1 \cdots \mathbf{x}_J]^{\mathrm{T}}$ from our input space, the corresponding output values $Y = [\mathbf{y}_1 \cdots \mathbf{y}_J]^{\mathrm{T}}$ will be distributed as

$$P(Y|X) = \mathcal{N}(Y|M(X), C(X,X)) \tag{B-1}$$

where

$$\mathcal{N}(\mathbf{x}|\mu,\Sigma) = \|2\pi\Sigma\|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^{\mathrm{T}}\Sigma^{-1}(\mathbf{x}-\mu)\right) \tag{B-2}$$

is the standard multivariate Gaussian distribution with $M(\cdot)$ as a mean function and $C(\cdot,\cdot)$ as a covariance function. The mean function generates a vector from the input X and the covariance function creates a (positive semi-definite) covariance matrix, these vectors each have the same number of dimensions as there were examples given (in this case J). Thus the GP handles an infinite object (a function defined over a continuous space) by considering only a finite set of input values at a time (J values in our example) and returning the corresponding finite set of output values.

Condition on Training Data Now we know about the distribution between the input and output spaces of a GP we can make it useful for regression by conditioning the GP on our training data. That is to say that we add the constraint that if any of the training inputs X_{tr} are presented to the GP, the corresponding training output Y_{tr} should be produced. Thus if we consider a new test input $\mathbf{x}^*$, we can find $\mathbf{y}^* = f(\mathbf{x}^*)$ as

$$\begin{bmatrix} Y_{\mathrm{tr}} \\ \mathbf{y}^* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} M(X_{\mathrm{tr}}) \\ M(\mathbf{x}^*) \end{bmatrix}, \begin{bmatrix} C(X_{\mathrm{tr}}, X_{\mathrm{tr}}) & C(X_{\mathrm{tr}}, \mathbf{x}^*) \\ C(\mathbf{x}^*, X_{\mathrm{tr}}) & C(\mathbf{x}^*, \mathbf{x}^*) \end{bmatrix}\right) . \tag{B-3}$$

We simplify the situation by subtracting the mean from our training input and output so that the mean function becomes a zero-vector. Standard manipulation can convert this to an output function as

$$f(\mathbf{x}^*) \sim \mathcal{N}\left(C(\mathbf{x}^*, X_{\mathrm{tr}})C(X_{\mathrm{tr}}, X_{\mathrm{tr}})^{-1}Y_{\mathrm{tr}}, \Sigma^*\right) \tag{B-4}$$

with the output covariance ('error bars') as

$$\Sigma^* = C(\mathbf{x}^*, \mathbf{x}^*) - C(\mathbf{x}^*, X_{\mathrm{tr}})C(X_{\mathrm{tr}}, X_{\mathrm{tr}})^{-1}C(\mathbf{x}^*, X_{\mathrm{tr}}) \ . \tag{B-5}$$

Therefore, the conditioned GP provides the regression function to predict output vectors for new input vectors with the predicted value as the mean of (B-4) and the covariance value Σ^* providing a measure of confidence in the prediction; a low standard deviation would indicate a very precise estimate. Please see [Rasmussen and Williams 2006] for further details.

The Covariance Function The final part of the GP regression model is then to specify the covariance function $C(\cdot,\cdot)$. This function must accept one or more samples from the input space as each argument and return a covariance value between each pairwise combination of samples. The function can depend on a small number of 'hyperparameters' that we shall denote with θ. The elements a particular covariance matrix given by a set of inputs are indexed

using the notation for representing a set of vectors described above. Thus

$$\left[C([\cdots \mathbf{x}_i \cdots]^{\mathrm{T}}, [\cdots \mathbf{x}_j \cdots]^{\mathrm{T}} | \theta)\right]_{i,j} = c\left(\mathbf{x}_i, \mathbf{x}_j | \theta\right) \tag{B-6}$$

where $[\cdot]_{i,j}$ denotes the element at the i^{th} row and j^{th} column of the matrix and $c(\mathbf{x}_i, \mathbf{x}_j|\theta)$ denotes the covariance between two input vectors.

Perhaps the most typical covariance function would be a radial basis function (RBF)

$$c\left(\mathbf{x}_i, \mathbf{x}_j | \theta\right) = \alpha \exp\left(-\frac{1}{2}\psi \left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2\right) \tag{B-7}$$

where α and ψ are the hyperparameters and therefore $\theta = [\alpha, \psi]$. We can see that this covariance function will produce a smooth mapping from the input space to the output space since nearby input vectors will have a high covariance and therefore the output vectors will be correlated. Similarly, if the input vectors are far apart (relative to the length -scale parameter ψ) then there will be a low covariance and the corresponding output vectors will be independent of one another.

The training of a GP consists of learning the values for these hyperparameters by maximizing the log-likelihood (B-1) of the training data $(X_{\text{tr}}, Y_{\text{tr}})$ to give an optimal setting θ^*. Since the covariance function can be a powerful, non-linear function, as in the case of the RBF, this training cannot be performed analytically and is instead performed using gradient based methods such as Conjugate Gradients.

B.2 The GP-LVM

We can think of the task of dimensionality reduction for a set of high dimensional vectors as identifying a set of low dimensional vectors that recreate the corresponding high dimensional vectors under a basic mapping. More powerfully, we could then go further to consider a manifold model that defines a generative mapping function, as well as the low dimensional vectors, such that new locations in the low dimensional space will be mapped to novel but plausible output values.

Manifold Modelling as Regression With this in mind, we return to the GP model that we have established as a powerful, non-linear regression technique. Suppose that no longer have pairs of input and output training examples, but instead only a set of high dimensional vectors; we move from supervised learning to unsupervised learning. If we treat these high dimensional vectors (the font outlines in our case) as the *output* of a GP regression model, then we observe that finding a manifold model is the same as finding a set of corresponding, low dimensional *input* vectors and the hyperparameters for the covariance function (the points on the manifold and the mapping).

The GP-LVM, therefore, may be thought of as a GP regression model with the regression input and output operating the other way around. The input to the GP-LVM model is a set of high dimensional data $\{\mathbf{y}_j\} \in \mathcal{R}^D$ (the regression output) and we find corresponding low dimensional data $\{\mathbf{x}_j\} \in \mathcal{R}^Q$ (the regression input), with $Q \ll D$, and the hyperparameters θ. The low dimensional space is also termed the 'latent' space since it is unobserved and must be inferred. Once we have performed this learning task, the manifold model then becomes a GP regression process, as in (B-4) where any location in the low dimensional space (a point on the font manifold) will be mapped into a new high dimensional vector (a new font outline) with the variance value indicating the likelihood of the generated high dimensional vector in terms of the training fonts provided.

Noise Model Representing a set of very high dimensional vectors by a low dimensional manifold is obviously a challenging process and we require a trade-off between producing a perfect reconstruction of the data and a smooth and useful manifold. With this in mind, a noise model is usually added to the covariance matrix during training to account for any slight noise in the high dimensional vectors. We therefore replace the basic covariance function of (B-7) with

$$c'\left(\mathbf{x}_i, \mathbf{x}_j | \theta\right) = \alpha \exp\left(-\frac{1}{2}\psi \left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2\right) + \sigma^2 \tag{B-8}$$

where σ is the standard deviation of the noise. The noise term is only present during training and is not used afterwards when generating high dimensional vectors using (B-4).

In our experiments, the noise variation was found to be very small, suggesting that the font outlines do indeed lie on a low dimensional manifold.

Training The training process for the GP-LVM considers the likelihood of the high dimensional data Y as

$$P(Y|X,\theta) = \prod_{m=1}^{M} \mathcal{N}\left(\mathbf{y}_m \mid \mathbf{0}, C(X,X|\theta) + \sigma^2 I\right) \tag{B-9}$$

where there are M training examples (one for each font) and I is the identity matrix. Whereas, for the GP model we simply maximized the log-likelihood over the hyperparameters θ, for the GP-LVM training we must maximize jointly over the latent vectors $X = [\cdots \mathbf{x}_j \cdots]^{\mathrm{T}}$ as well as θ such that

$$X^*, \theta^* = \arg\max_{X,\theta} \log\left[P(Y|X,\theta)\right] \ . \tag{B-10}$$

Again, the covariance function is a complex, non-linear function, both in terms of its relationship to X and to θ; however, gradients may be found against both so training may be performed using Conjugate Gradients [Lawrence 2005]. Such methods require initial values for both X and θ. As is standard [Lawrence 2005], we initialize the latent values with a linear PCA reduction of the high dimensional vectors Y and set the hyperparameters to conservative values with a wide uninformative prior.

B.3 Manifold Dimensionality

For the purpose of visualization, we have limited the manifold dimensionally in the figures (and the interactive javascript viewer) to $Q = 2$. In reality the dimensionality of the manifold may be higher. In order to identify the true dimensionality of the manifold, we used a more advanced derivate of the GP-LVM, the Bayesian GP-LVM of Titsias and Lawrence [2010]. Their approach uses the 'Automatic Relevance Determination' (ARD) covariance function

$$c_{\text{ard}}\left(\mathbf{x}_i, \mathbf{x}_j | \theta\right) = \alpha \exp\left(-\frac{1}{2}\sum_{q=1}^{Q} \psi_q (x_{i,q} - x_{j,q})^2\right) \tag{B-11}$$

instead of the standard RBF kernel of (B-7). Comparing the RBF and ARD kernels we see that the ARD kernel has a different length-scale parameters for each dimension, $\{\psi_q\}$. The Bayesian GP-LVM marginalizes out, rather than optimize, the latent space X. This allows us to set Q to a large value, run the learning process, and then look at the values of each ψ_q. If a dimension is unnecessary, we will have the corresponding value of $\psi_q \to 0$. Thus, we count the number of non-zero length-scales to determine how many dimensions are useful. For the joint manifold, shown in Figure 1, the best setting is obtained with $Q = 4$ manifold; this explains the separation into islands when constrained to $Q = 2$. Manifolds for individual characters may differ in intrinsic dimensionality, for example the character 'g' has $Q = 3$.

References

LAWRENCE, N. 2005. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *The Journal of Machine Learning Research 6*, 1783–1816.

NAVARATNAM, R., FITZGIBBON, A., AND CIPOLLA, R. 2007. The joint manifold model for semi-supervised multi-valued regression. In *IEEE 11th International Conference on Computer Vision*.

RASMUSSEN, C. E., AND WILLIAMS, C. 2006. *Gaussian Processes for Machine Learning*. MIT Press.

TITSIAS, M., AND LAWRENCE, N. 2010. Bayesian gaussian process latent variable model. In *13^{th} International Workshop on AI and Stats.*