EXAM PROJECT FOR PML 2022/2023

REPORT

Christian Dybdahl Troelsen
Department of Computer Science
University of Copenhagen
Universitetsparken 1
DK-2100 Copenhagen Ø
tfp233@alumni.ku.dk

Jens Sørensen
Department of Computer Science
University of Copenhagen
Universitetsparken 1
DK-2100 Copenhagen Ø
qrw992@alumni.ku.dk

Mathias Rasmussen
Department of Computer Science
University of Copenhagen
Universitetsparken 1
DK-2100 Copenhagen Ø
tjc725@alumni.ku.dk

January 20, 2023

1 Density modeling

1.1 Implement a convolutional VAE

Architecture of the convolutional VAE Our implementation of the convolutional VAE (CVAE) is similar in architecture to the original VAE. The primary difference between the two models is that the two linear layers in the encoder and decoder part of the original VAE have been replaced by respectively two convolutional layers and two transposed convolutional layers in the convolutional VAE. The two convolutional layers in the encoder part of the CVAE both use kernels of size 3×3, 1×1 padding and a stride of 2×2. The first convolutional layer has 1 input channel and 16 output channels, while the second convolutional layer has 16 input channels and 32 output channels. The transposed convolutional layers in the decoder part of the CVAE complement the aforementioned convolutional layers. Both transposed convolutional layers use kernels of size 3×3, 1×1 padding, 1×1 output padding and a stride of 2×2. The first transposed convolutional layer has 32 input channels and 16 output channels, while the second transposed convolutional layers has 16 input channels and 1 output channel. Conceptually, the encoder part of the CVAE condenses its input images $\mathbf{x}$ of size $D = H \times W$ into a compact but feature rich representation $\mathbf{z}$ of size $C = 2$, while the corresponding decoder "unfolds" this compact representation in reverse order, ultimately producing output images $\mathbf{y}$ of the same size as $\mathbf{x}$.

Parameter estimation In order to optimize the CVAE model we estimate the parameters $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ that maximize the ELBO, which is a lower bound on the log likelihood of the data $\mathbf{x}$ and is defined as $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}, \mathbf{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\ln p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})] - \mathrm{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \| p_{\boldsymbol{\theta}}(\mathbf{z}))$. This is the same criterion used to optimize the original VAE. In particular, both models use the reparameterization trick to sample latent variables $\mathbf{z}$ from a factorized gaussian approximate posterior $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \mathbf{u}, \mathrm{diag}(\boldsymbol{\sigma}^2))$, so the Kullback-Leibler divergence, which acts as a regularization term in the ELBO, simplifies to $\frac{1}{2}\sum_{d=1}^{D}(\sigma_d^2 + \mu_d^2 - 1 - \ln(\sigma_d^2))$, assuming that the prior over $\mathbf{z}$ is also a diagonal gaussian $p_\theta(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$. Here $\ln \boldsymbol{\sigma}^2$ and $\boldsymbol{\mu}$ are the outputs of the encoder when applied to $\mathbf{x}$. In order to compute the first term in the ELBO, which is the expected likelihood of $\mathbf{x}$ given $\mathbf{z}$, we utilize the output $\mathbf{y}$ of the decoder. We fed $\mathbf{y}$ through a sigmoid activation layer, which produces the mean parameters $\mathbf{p}$ of a factorized multivariate Bernoulli distribution $\prod_j^D \mathrm{Bernoulli}(x_j; p_j)$ acting as the likelihood $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ of the input $\mathbf{x}$ given latent variable $\mathbf{z}$. We then approximate the expected log likelihood by computing the negative of the binary cross entropy of $\mathbf{p}$ and $\mathbf{x}$, where $\mathbf{x}$ is treated as a set of probability values. Given the simplified ELBO, we optimize the parameters of both models using a variant of gradient descent, namely the Adam algorithm with a learning rate of 0.001 and a batch size of 128. In order to ensure convergence we run the Adam algorithm for a total of 25 epochs for each model. We train both models on the complete MNIST training set.

Performance of VAE models We compare the performance of the two VAE models on the MNIST test set using two quantitative measures and three qualitative experiments. The two quantitative measures are the sample-wise

mean ELBO on the MNIST test set and the MSE loss between samples from the MNIST test set and corresponding reconstructions produced by the VAE models. The three qualitative experiments are

1. Clustering MNIST test data in latent space by mapping each observation $\mathbf{x}_i$ to corresponding mean parameters $\boldsymbol{\mu}_i$ using the encoder and subsequently plotting these $\boldsymbol{\mu}_i$ with colors based on the actual labels t_i associated with each $\mathbf{x}_i$.
2. Exploring the latent space by sampling latent variables $\mathbf{z}$ deterministically from a regular grid, then using the decoder to map these $\mathbf{z}$ to corresponding mean parameters $\mathbf{p}$, and finally visualizing these $\mathbf{p}$ as images.
3. Reconstructing selected observations $\mathbf{x}_i$ from the MNIST test set by mapping these to posterior distributions $q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i)$, then randomly sampling $\mathbf{z}_i$ from these posterior distributions, and finally using the decoder to map these $\mathbf{z}_i$ to corresponding mean parameters $\mathbf{p}_i$, which are finally interpreted as reconstruction of the inputs $\mathbf{x}_i$.

A comparison of the sample-wise ELBO and MSE loss for the two VAE models is shown in Table 1. As can be seen, both models seem to have a sample-wise mean ELBO that is on the same scale as well as an MSE loss that is on the same scale. However, it is also apparent that both the sample-wise mean ELBO and the MSE loss is slighter better for the original VAE than for the CVAE. This indicates the VAE has learned a slightly better approximation of the density implicitly defined by the MNIST test set as well as being slightly better at reconstructing images from the MNIST test set. The clustering of the MNIST test data shown in Figure 3a and Figure 3b indicate a similar relationship between the performance of the two models. In particular, we see that both models seem to have learned a representation in latent space that is capable of separating most samples from the MNIST test set based on class labels. That being said, it also clear that the CVAE struggles more than the VAE with separating some samples based on class labels, in particular those representing digits 4, 7, and 9.

The exploration of latent space shown in Figure 2a and Figure 2b corroborate the idea that the VAE is better at separating classes. Note in particular how the mean parameter images produced by the VAE seem to be slightly more well-defined (i.e. less blurry) than those produced by the CVAE. It is also worth noting that the exploration of latent space in the CVAE seem to produce a greater variety of digits than does the corresponding exploration of latent space in the VAE (it seems to mainly produce 6s, 0s, 2s, 1s and 9s). The reconstruction results shown in Figure 1b and Figure 1c also indicate that the VAE is somewhat superior to the CVAE, in this case in terms of accurately reconstructing samples from the MNIST test set. This is most obvious when considering the reconstructions of digits 2 and 4, which the VAE has rendered reasonably clearly, contrary to the CVAE, which has rendered them somewhat blurry. In particular, it seems that the CVAE has trouble distinguishing 4 from 9.

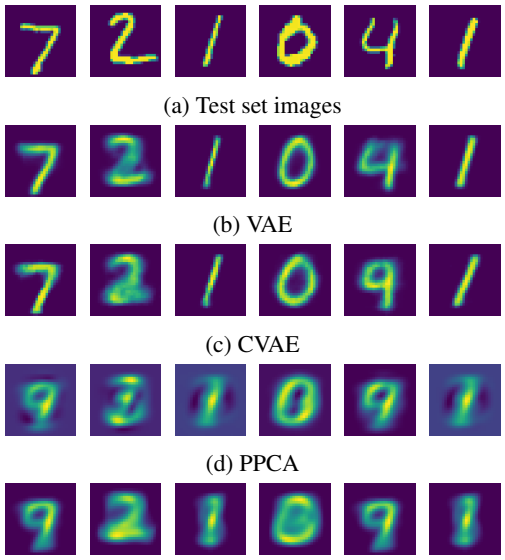
The reason why the CVAE slightly underperforms compared to the VAE is not entirely clear to us. In fact, one might expect the CVAE would perform better than the VAE, given that convolutional layers in general are better than linear layers at capturing complex spatial information in image data. One reason why the CVAE underperforms may be that the MNIST dataset is too simple and hence convolutional layers are not required to extract its most salient features. In particular, since MNIST digits are centered and of similar size, the convolutional layers might not be necessary. Its also possible that the CVAE is not deep enough to effectively learn a useful representation of the MNIST data, that the kernel size (3×3) used by its layers is not large enough to capture spatial dependencies in the MNIST data or that a stride of (2×2) results in too significant a loss of information.

1.2 Alternative models

1.2.1 Probabilistic PCA

Essential properties and advantages/disadvantages The first model which we implement is the probabilistic PCA (PPCA) model introduced in Chapter 12.2.1 of [Bishop and Nasrabadi, 2006]. The reason this model is attractively is primarily due to the fact that its likelihood is tractable. In particular, given a dataset $\mathbf{X} = \{\mathbf{x}_n\}$ of N samples $\mathbf{x}_n$ with D dimensions each, the log likelihood of the PPCA model is given by $\ln p(\mathbf{X}|\mathbf{W}, \boldsymbol{\mu}, \sigma^2) = -\frac{N}{2}\left(D\ln(2\pi) + \ln|\mathbf{C}| + \mathrm{Tr}\left(\mathbf{C}^{-1}\mathbf{S}\right)\right)$, where $\mathbf{C} = \mathbf{W}\mathbf{W}^{\mathsf{T}} + \sigma^2\mathbf{I}$ and $\mathbf{S}$ is the sample covariance matrix of $\mathbf{X}$. Moreover, there exists a closed form-solution for the parameters $\mathbf{W}, \boldsymbol{\mu}, \sigma^2$, which maximize this log-likelihood.

The existence of a closed-form expression for the likelihood function is useful, as it allows for direct comparison with other density models. The fact that the maximizing parameters of the likelihood function also have a closed-form solution means we can easily optimize the PPCA model. At the same time, the PPCA model also allows for numerical optimization using the Expectation-Maximization (EM) algorithm, which can be more efficient in higher dimensions and when dealing with missing data. From a practical point of view, the PPCA model is useful, as it can be used both for dimensionality-reduction (like the regular PCA model), but also for random sampling due to its probabilistic nature. Finally, a major advantage of the PPCA model is that it allows capturing the most important covariance in its included principal components, while capturing the average variance for all left-out components in σ^2.



(a) Test set images

(b) VAE

(c) CVAE

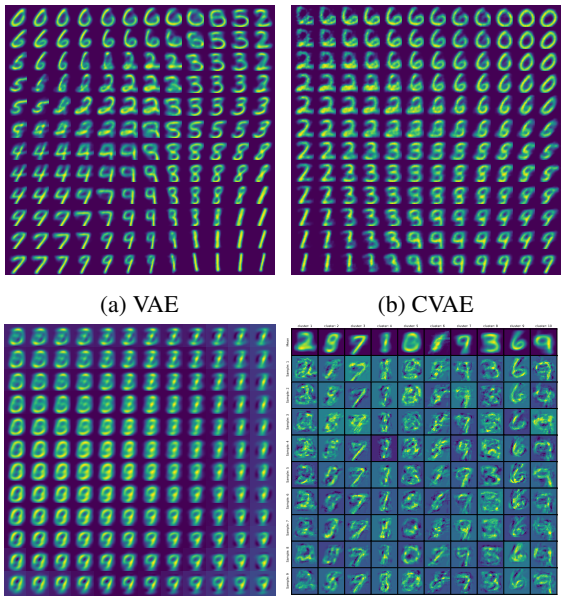
(d) PPCA

(e) GMM

Figure 1: Comparison of MNIST test set images and corresponding mean parameters generated by density models.

	Log-Likelihood/ELBO	MSE
VAE	-1.4281×10^2	0.0375
CVAE	-1.5705×10^2	0.0444
PPCA	-4.3297×10^3	0.0558
GMM	-1.0664×10^7	0.0589

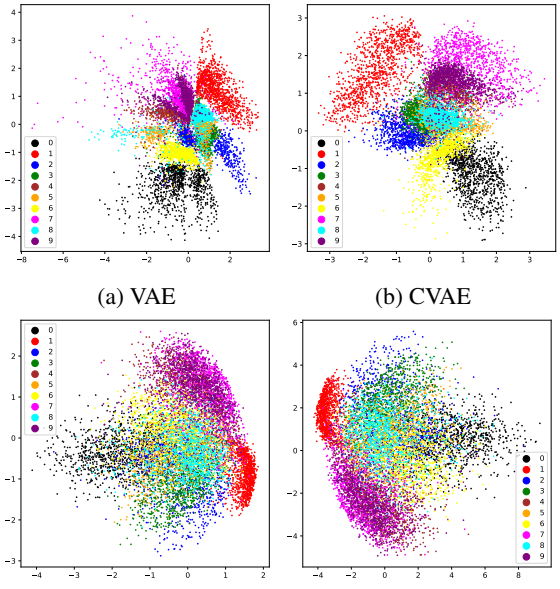
Table 1: Model performance metrics



(a) VAE (b) CVAE

(c) PPCA (d) GMM

Figure 2: Interpolating images from latent space variables using trained density models.



(a) VAE (b) CVAE

(c) PPCA (d) GMM

Figure 3: Clustering on MNIST test data (projection to latent space) using trained density models.

Perhaps the most obvious disadvantage of the PPCA model is that it is based on a linear-gaussian framework and hence, unlike the VAE, may not be able to capture complex non-linear structure in the data it is trained on. On another note, it is worth pointing out that the maximum-likelihood solution to the PPCA model only determines the latent space up to an arbitrary rotation $\mathbf{R}$. We shall for simplicity set $\mathbf{R} = \mathbf{I}$. Moreover, we shall use $M = 2$ principal components, so that we may easily compare the latent space of the PPCA model with that of the original VAE model.

Model performance We assess whether the PPCA model is better or worse than the original VAE by using the same quantitative measures and qualitative experiments used to compare the CVAE with the original VAE. For the MSE loss we compare images $\mathbf{x}_i$ from the MNIST test set with corresponding mean parameter values $\boldsymbol{\eta}_i$ obtained by first mapping each $\mathbf{x}_i$ to a posterior distribution $p(\mathbf{z}_i|\mathbf{x}_i) = \mathcal{N}\left(\mathbf{z}|\text{proj}\left(\mathbf{x}_i\right), \sigma^2\mathbf{M}^{-1}\right)$,where $\text{proj}(\mathbf{x}_i) = \mathbf{M}^{-1}\mathbf{W}^\mathsf{T}\left(\mathbf{x}_i - \boldsymbol{\mu}\right)$ and $\mathbf{M} = \mathbf{W}^\mathsf{T}\mathbf{W} + \sigma^2\mathbf{I}$,then sampling a $\mathbf{z}_i$ from this distribution, and finally computing $\boldsymbol{\eta}(\mathbf{z}_i) = \mathbf{W}\mathbf{z}_i + \boldsymbol{\mu}_i$. This same procedure is used for the reconstruction experiment, though in this case we compare selected $\mathbf{x}_i$ and corresponding $\boldsymbol{\eta}_i$ visually. For the clustering experiment, we reuse $\text{proj}\left(\mathbf{x}_i\right)$ to map each sample $\mathbf{x}_i$ to its corresponding mean parameter in latent space. Symmetrically, for the exploration of latent space we use reuse the back-projection $\boldsymbol{\eta}(\mathbf{z}_i)$ for mapping the deterministically sampled latent variables $\mathbf{z}_i$ to corresponding mean parameters that are subsequently represented as images.

As can be seen in Table 1, the sample-wise mean log likelihood of the PPCA model is very low compared to the sample-wise mean ELBO of the original VAE. One explanation why the metric is so low may be that the MNIST test set is rather large. Hence, even with a modest average probability $p(\mathbf{x}_i|\mathbf{W}, \boldsymbol{\mu}, \sigma^2)$ of each sample $\mathbf{x}_i$ in the MNIST test-set, we still may end up with a very small total likelihood $p(\mathbf{X}|\mathbf{W}, \boldsymbol{\mu}, \sigma^2) = \prod_i^N p(\mathbf{x}_i|\mathbf{W}, \boldsymbol{\mu}, \sigma^2)$. On the other hand, considering the fact that the MSE of the PPCA model is also rather low compared to that of the original VAE, it seems likely that the PPCA model has neither learned to model the underlying distribution of the MNIST test set nor to reconstruct its samples accurately. This conjecture is supported by the qualitative experiments. Based on Figure 3c, one may infer that the PPCA model has not learned to separate samples in latent space based on labels very well. In particular, it seems only capable of separating samples with labels that are either 1 or 0. The exploration of latent space shown in Figure 2c seem to further solidity the idea that the PPCA model has not learned a proper representation of digits other than 1, 0 and perhaps 9. Similarly, the reconstructions in Figure 1d are very bad, with only 0 and 1 being somewhat accurately reconstructed.

Why the PPCA model performs so much worse than the original VAE is most likely due to its aforementioned linear components, which are not capable of capturing the non-linear nature of the MNIST dataset. On the other hand, because the VAE utilizes components such as ReLU activation functions, it can more easily model the complexities of the MNIST dataset. The PPCA model might also perform poorly due to the fact that the samples from the MNIST dataset are very high dimensional and may not necessarily lie on a low-dimensional manifold, which a latent space based on 2 principal components assumes. In general, a very low likelihood may be expected when dealing with data that is as high-dimensional as the samples from the MNIST dataset.

1.2.2 Gaussian Mixture Model

Essential properties and advantages/disadvantages The second method that we have chosen to implement is a gaussian mixture model as described in Chapter 9.2 of [Bishop and Nasrabadi, 2006]. The main advantage of this method is that it attempts to separate the data into clusters that should represent the different classes of the dataset.

The method does however have the disadvantage that no closed form solution exists to estimate the parameters of the model and as such the EM method is used. This makes the method dependent on the initial estimates of the parameters and if chosen poorly the method might not converge.

Model implementation The implementation of the method follows the description in Chapter 9.2 of [Bishop and Nasrabadi, 2006] but as the method requires that each x is a vector, each image was flattened to a vector of size $28 * 28$. This causes a problem when using the PDF of the multivariate normal distribution as each input has an extremely small probability and resulted in a value of 0 being returned. Instead, the implementation provided by `scikit-learn` was used to apply the method to the dataset.

Model performance To compare this model with the previous three we use the same quantitative measures and can be seen in Table 1. The mean log-likelihood was calculated by using the built-in method `score` for the test set. The value is extremely negative and suggests that the model is performing very poorly. The MSE was calculated by first obtaining the most likely cluster for each sample x_i in the test set, and then using the mean value for said cluster as the value of $\hat{x}_i$. The MSE of the model is slightly more than PPCA and is still worse than both the VAE and CVAE.

As the implementation of the gaussian mixture model in `scikit-learn` does not provide a method to sample the obtained distribution we cannot perform the same interpolation as with the other model. Instead, we use a multivariate normal distribution, with the mean and covariance to sample for each cluster. The mean for each cluster and the result of sampling, based on each cluster, is displayed in Figure 3d.

From these it can be seen that the method appears to have found a suitable cluster for the digits 2, 1, 3 and 6. The remaining clusters does not appear unique with the digit 9 belonging to three clusters. When sampling around each cluster most becomes unrecognizable, while the remaining mainly keep their form but still contains a lot of noise.

Based on the quantitative and qualitative measures it can be concluded that a gaussian mixture model is not good alternative to a variational autoencoder.

2 Function fitting

2.1 Fitting a GP with Pyro

We have implemented a full Bayesian GP modeling approach with Pyro. We have applied our implementation to the dataset $\mathcal{D}$ described in the assignment text. We use NUTS, with a Gaussian RBF kernel, to sample from the posterior $p(\theta|\mathcal{D})$. The priors used for the kernel parameters σ_l^2 and σ_s^2 are the ones given in the assignment text. For hyperparameters, we use $W = 100$ warmup steps and $C = 4$ chains. In Table 2 we show the ESS and RHat values computed for our trained model. We achieve ESS values of above 400, which indicates that our model approximates the posterior distributions of the actual parameters well. Furthermore, we have RHat values of 1.01 for both parameters, indicating that the sampling has converged. In Figure 5 we show 500 samples from $p(\theta|\mathcal{D})$ from each chain, and in Figure 6 we show $p(f^*|x^*, \mathcal{D})$ along with its confidence interval and mean.

2.2 Bayesian Optimization

In Figure 7 we show the results from running our Bayesian optimization loop. We see that our implementation indeed is capable of finding the global minimum of f, however, from running the optimization loop multiple times, we notice that we are not able to reliably find the global minimum of f. Sometimes the algorithm gets stuck in a local minimum and is never able to escape. This even happens from the first iteration in some cases. Furthermore, we also have intermittent numerical stability problems. In an attempt to address these issues, we have tried a multitude of the different kernels supplied by the Pyro library, while keeping the original prior supplied to us. However, changing the kernel made no difference, and in some cases led to worse results than we observed with the RBF kernel. We refer to the appendix for plots showing the results of running the Bayesian optimization loop with the other kernels.

σ_l^2	
ESS bulk	412.0
ESS tail	725.0
RHat	1.01
σ_p^2	
ESS bulk	534.0
ESS tail	614.0
RHat	1.01

Table 2: Effective sample size (ESS) and RHat values calculated with Arviz

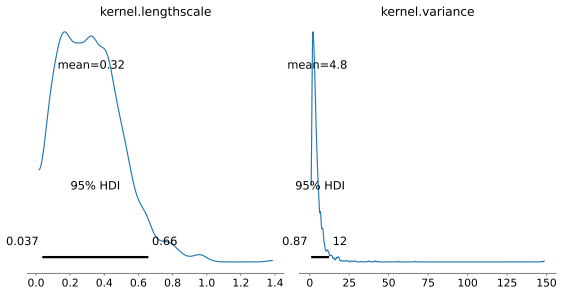


Figure 4: Plot showing the posterior densities of the RBF kernel's learned parameters

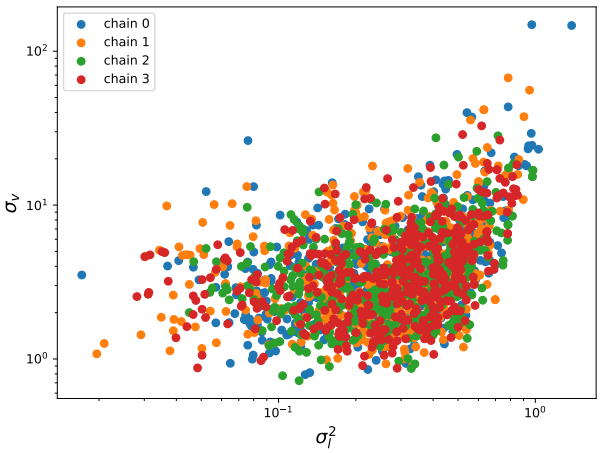


Figure 5: Scatter plot on log-log-scale of $N = 500$ samples from $p(\theta|\mathcal{D})$ from 4 chains

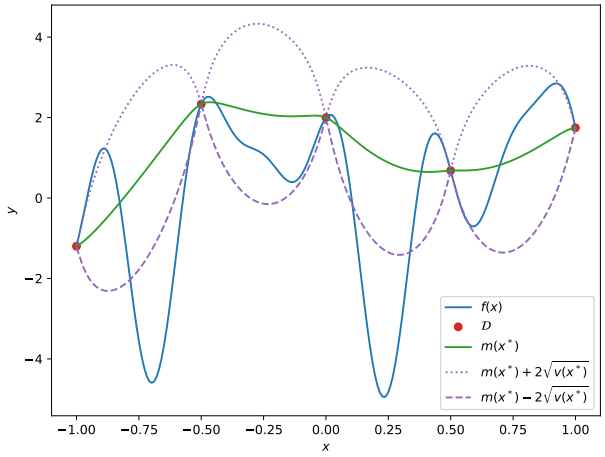


Figure 6: Plot of $p(f^*|x^*, \mathcal{D})$ along with its mean and confidence interval

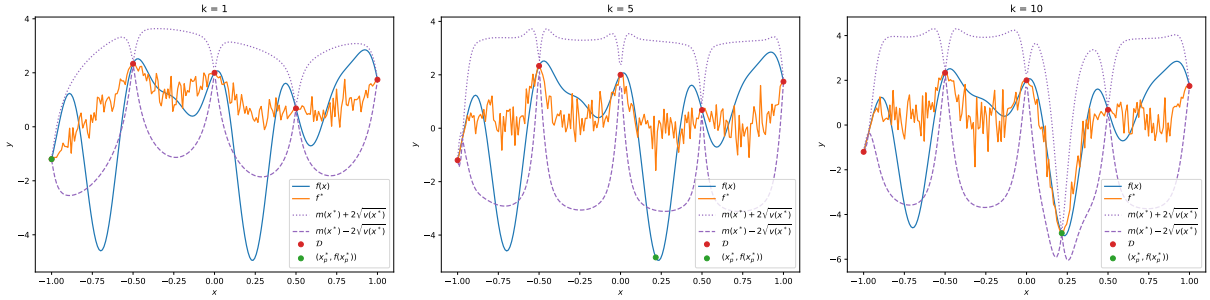


Figure 7: Plots from running our Bayesian optimization loop at iteration k

3 Bibliography

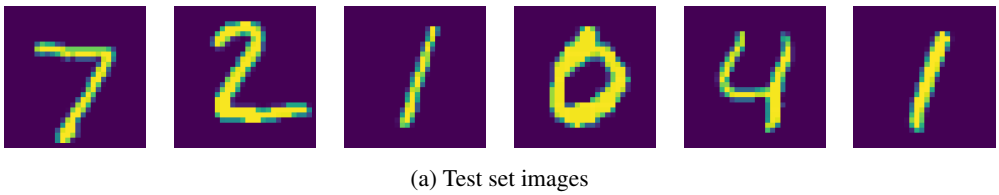
Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

Appendices

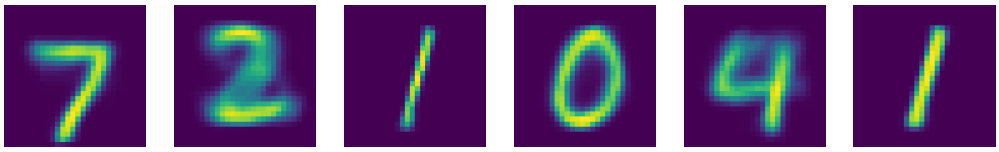
A Code repository

All our code can be found in the Jupyter notebook scripts `a.ipynb` and `b.ipynb` available in the folder `scripts` in the public repository at https://github.com/JackismyShephard/pml-22.

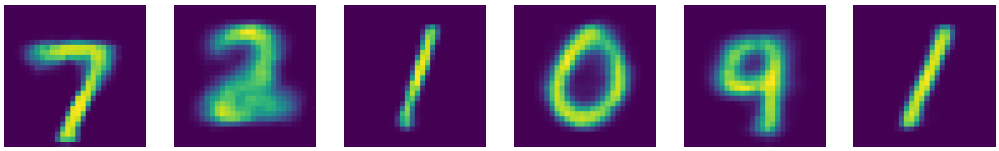
B Extra figures



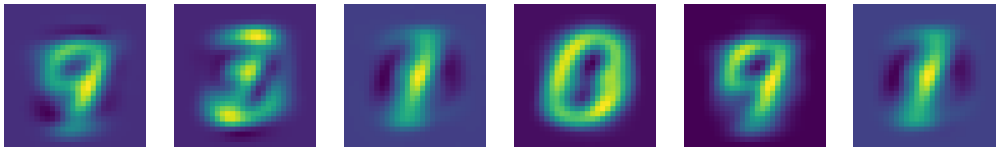
(a) Test set images



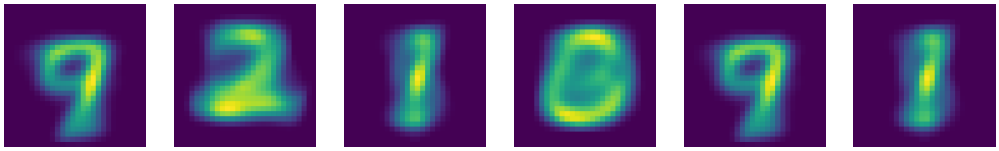
(b) VAE



(c) CVAE

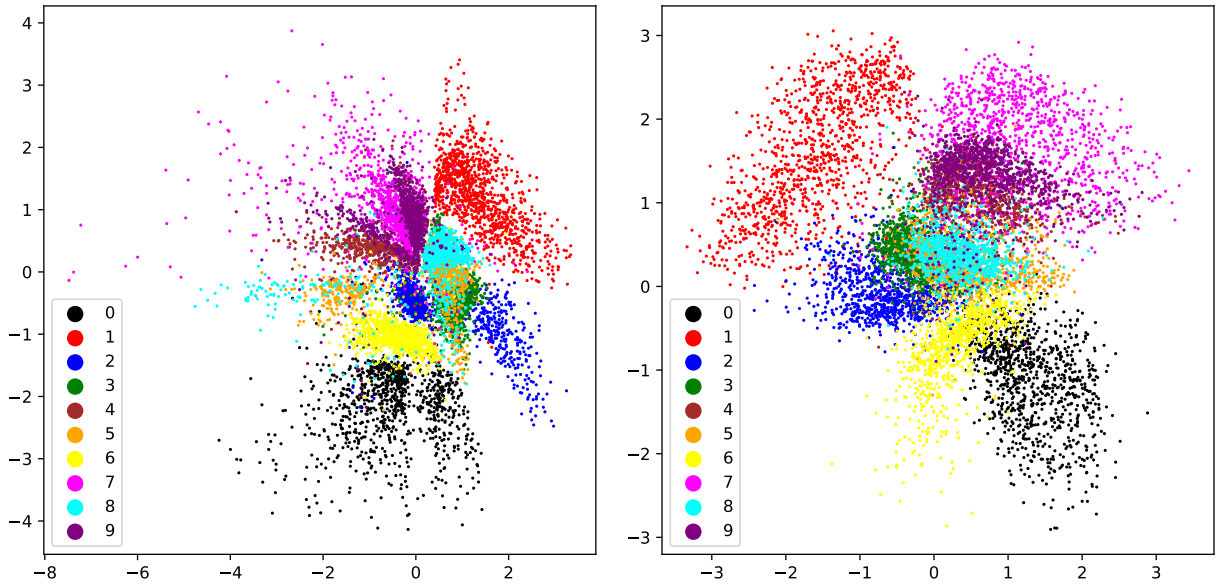


(d) PPCA



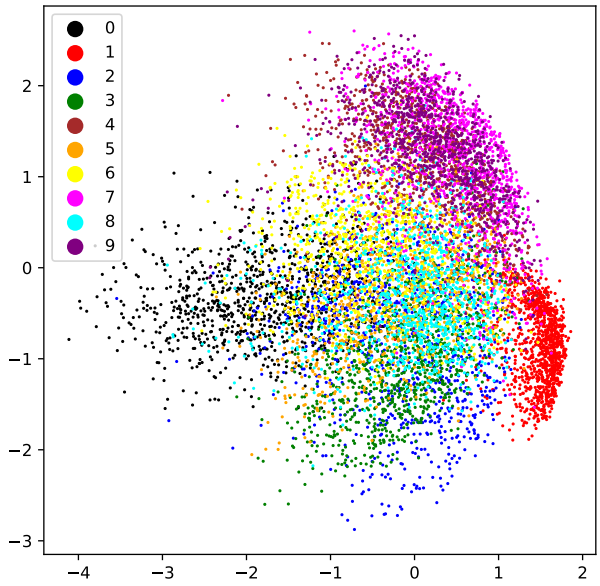
(e) GMM

Figure 8: Comparison of MNIST test set images and corresponding reconstructions sampled from density models

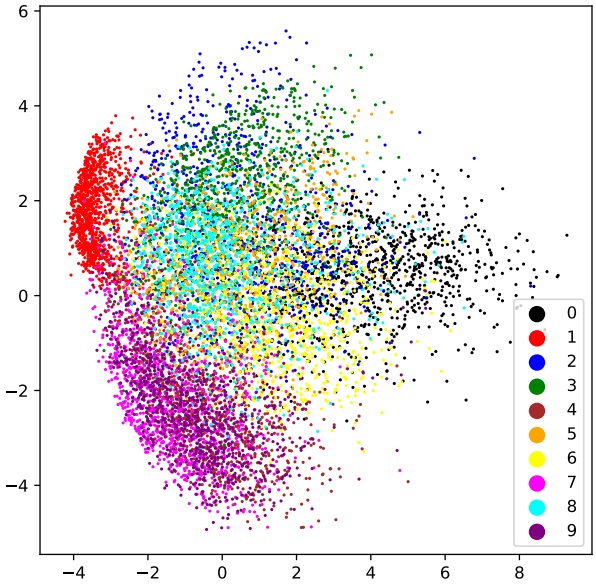


(a) VAE

(b) CVAE

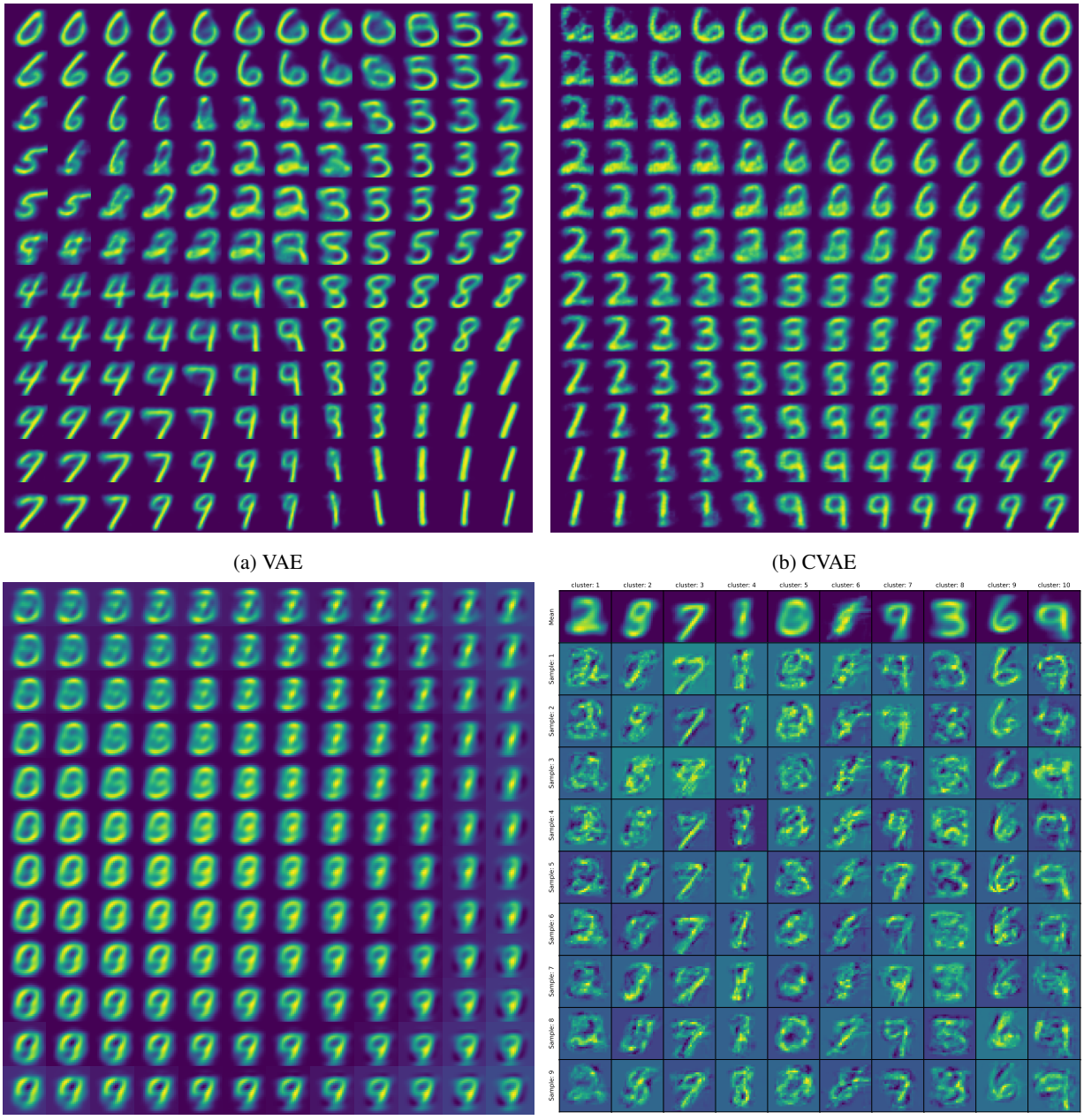


(c) PPCA



(d) GMM

Figure 9: Clustering on MNIST test (projection to latent space) using trained density models.



(a) VAE

(b) CVAE

(c) PPCA

(d) GMM

Figure 10: Interpolating images from latent space variables using trained density models

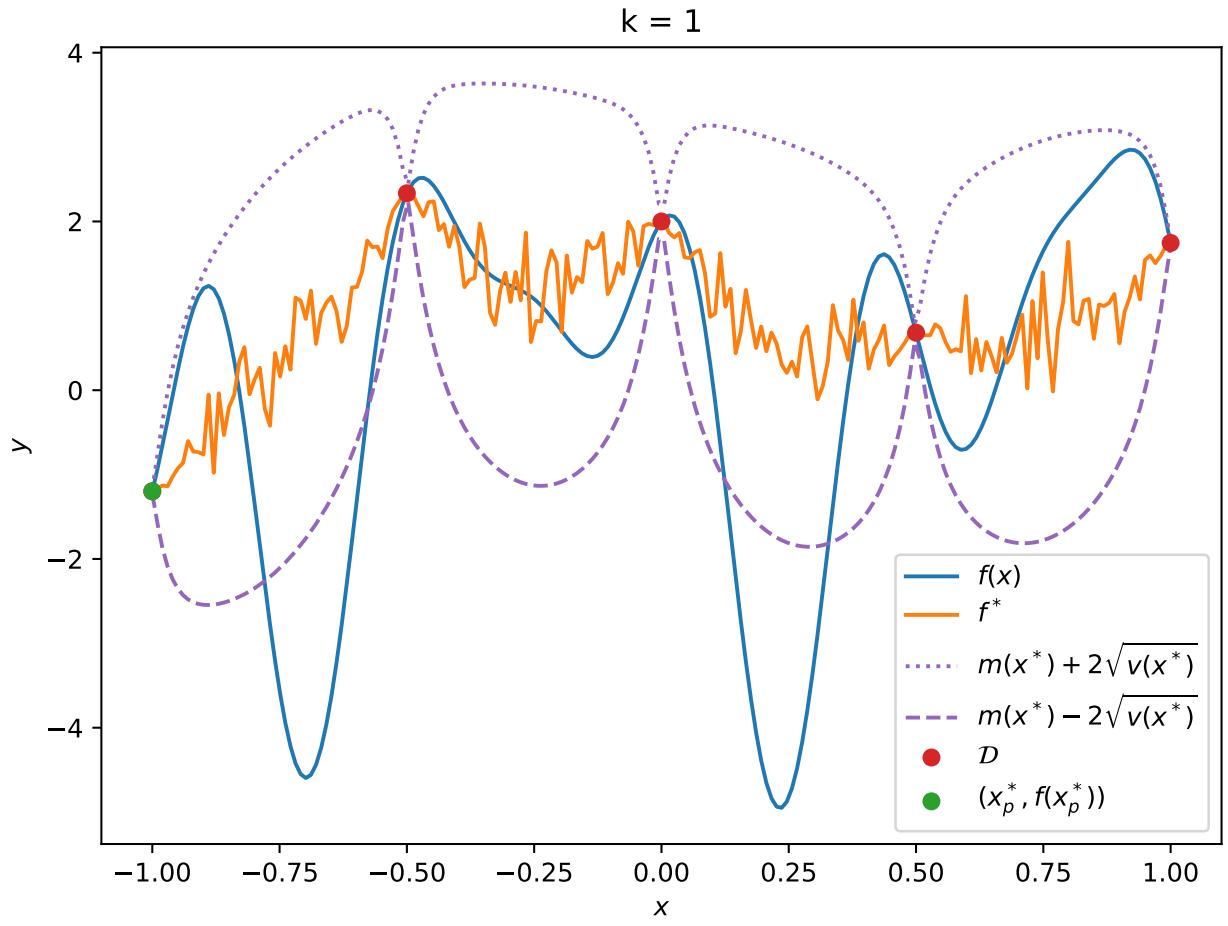


Figure 11: Result of the Bayesian optimization loop with RBF kernel at iteration $k = 1$

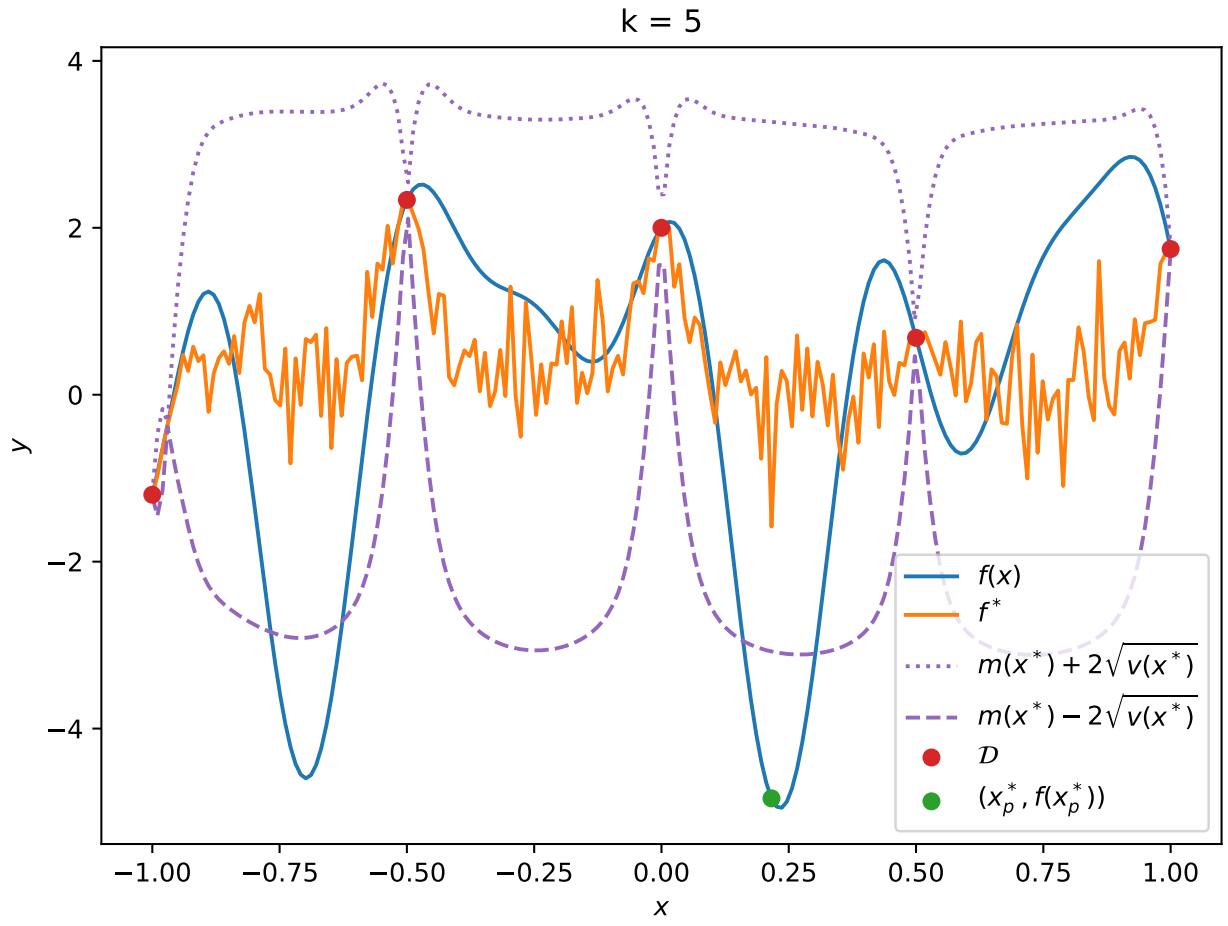


Figure 12: Result of the Bayesian optimization loop with RBF kernel at iteration $k = 5$

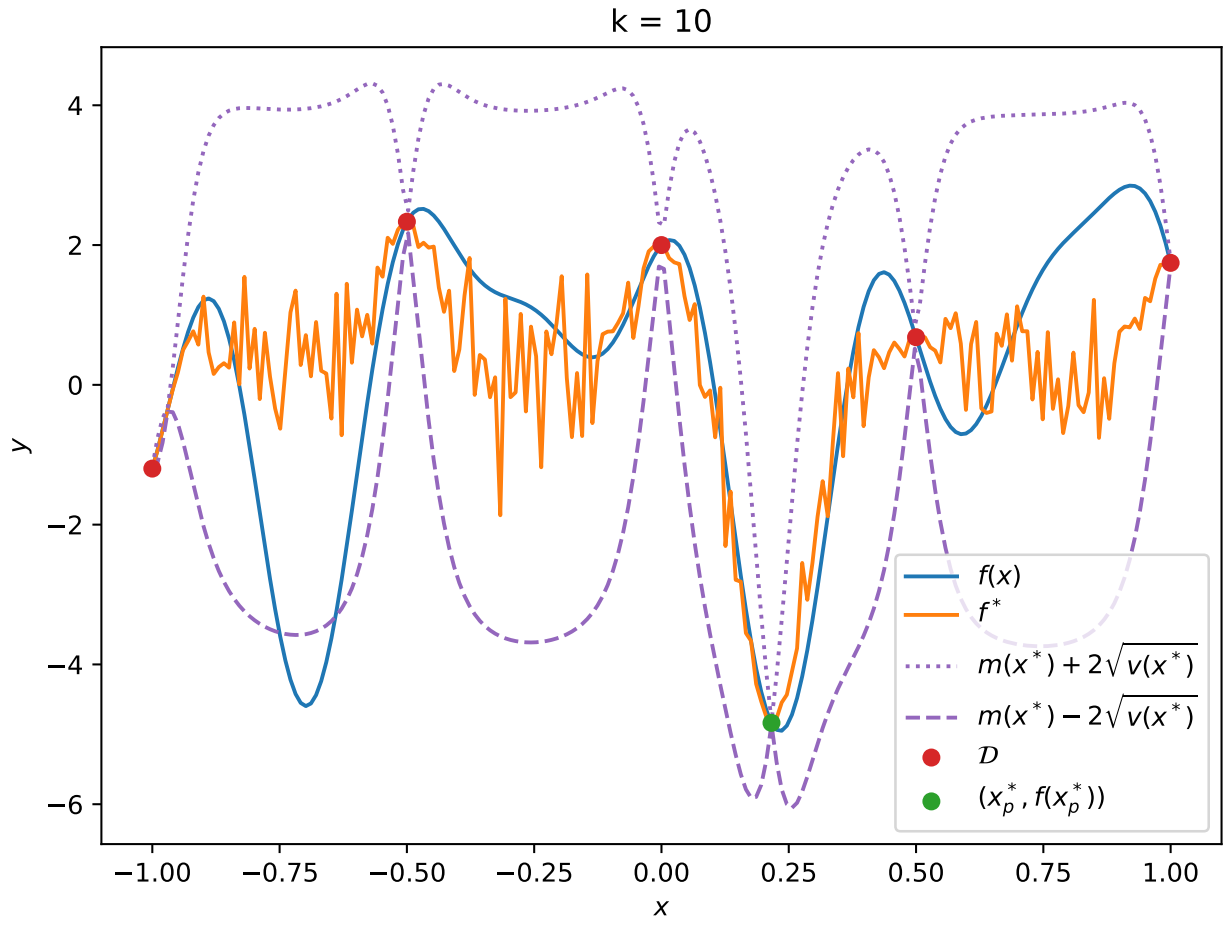


Figure 13: Result of the Bayesian optimization loop with RBF kernel at iteration $k = 10$

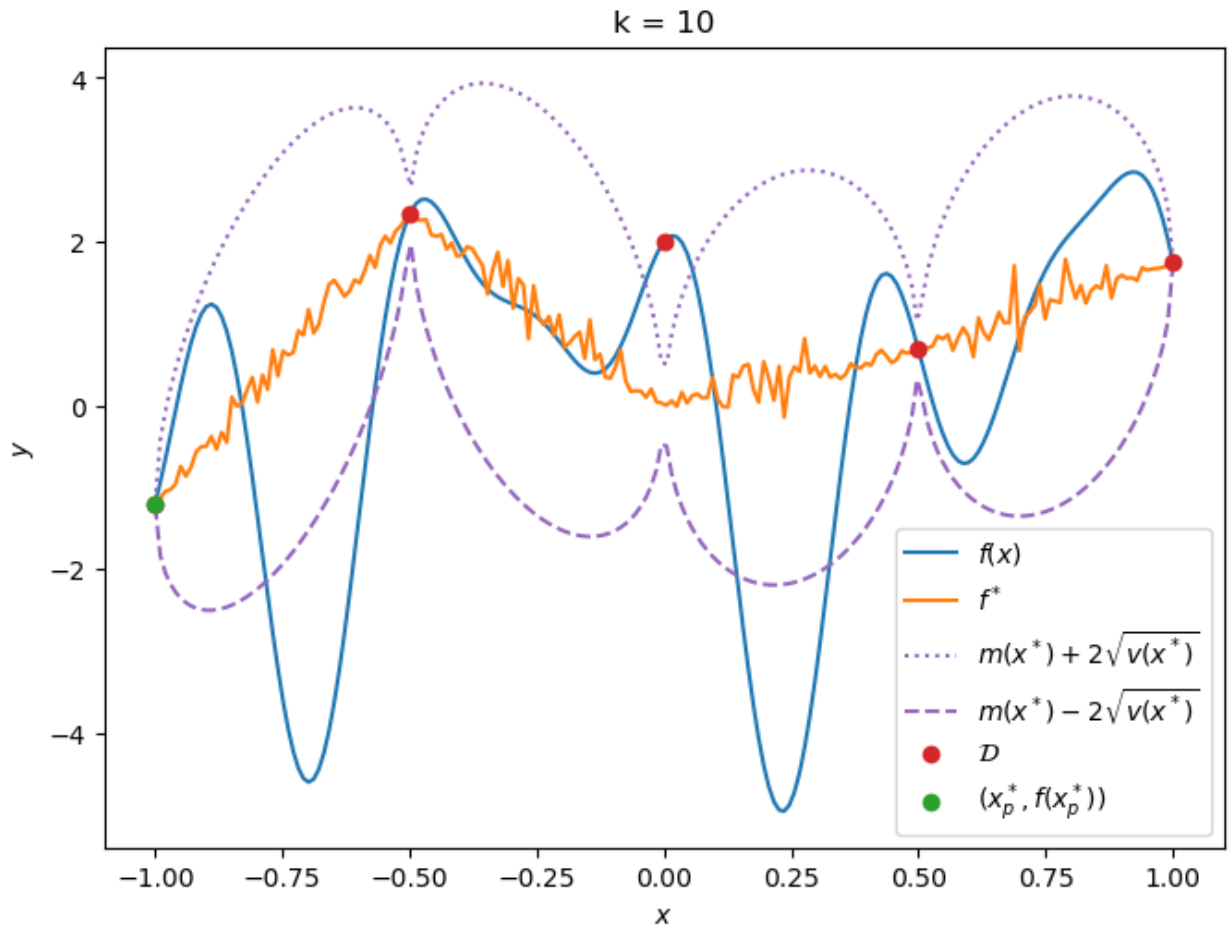


Figure 14: Result of running the Bayesian optimization loop with a Brownian kernel

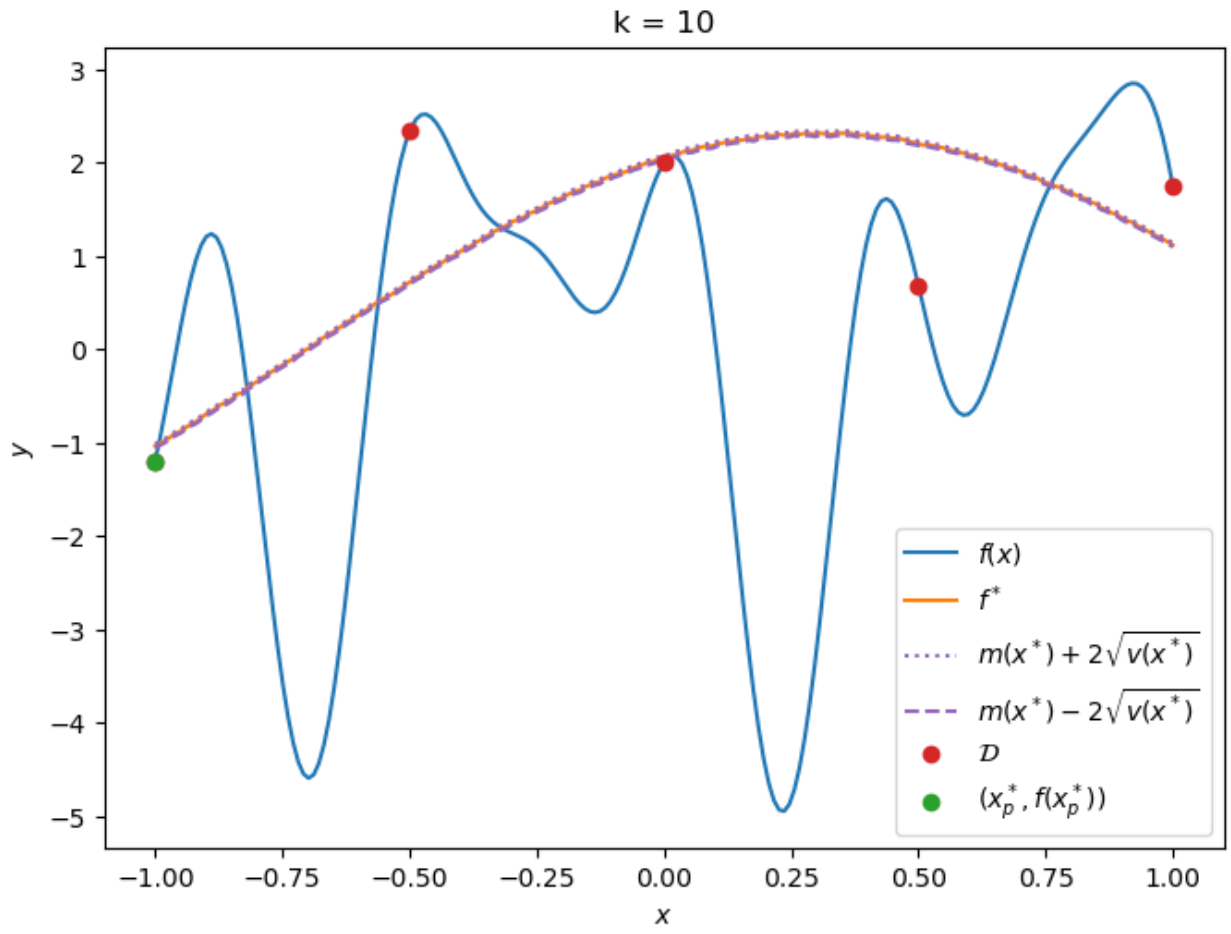


Figure 15: Result of running the Bayesian optimization loop with a Cosine kernel

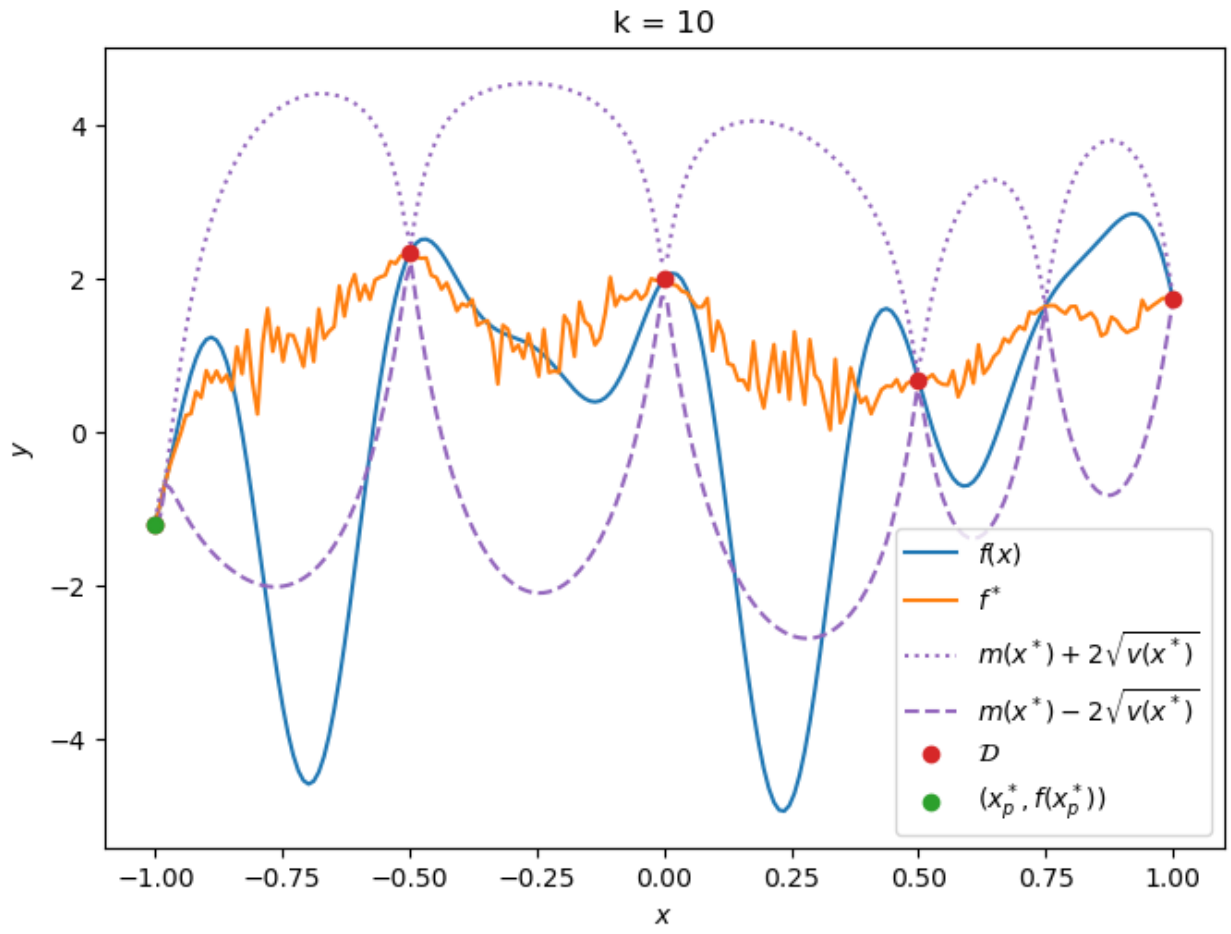


Figure 16: Result of running the Bayesian optimization loop with a Matern32 kernel

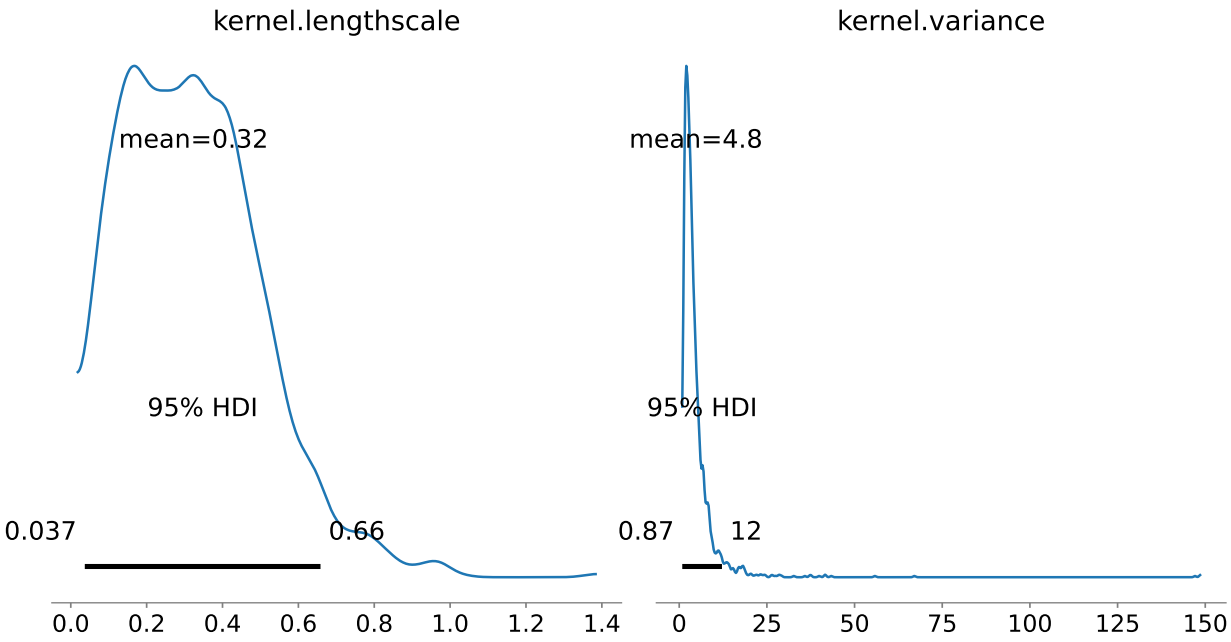


Figure 17: Highest density intervals for our RBF kernel parameters, produced with Arviz