1 Kernel Methods

1.1 Slide 6

This is known as a dual formulation because, by noting that the solution for $\mathbf{a}$ can be expressed as a linear combination of the elements of $\boldsymbol{\phi}(\mathbf{x})$, we recover the original formulation in terms of the parameter vector $\mathbf{w}$.

1.2 Slide 9

Given valid kernels $k_1(\mathbf{x}, \mathbf{x}')$ and $k_2(\mathbf{x}, \mathbf{x}')$, the following new kernels will also be valid:

$$
\begin{aligned}
k(\mathbf{x}, \mathbf{x}') &= c k_1(\mathbf{x}, \mathbf{x}') && (1.1)\\
k(\mathbf{x}, \mathbf{x}') &= f(\mathbf{x}) k_1(\mathbf{x}, \mathbf{x}') f(\mathbf{x}') && (1.2)\\
k(\mathbf{x}, \mathbf{x}') &= q\left(k_1(\mathbf{x}, \mathbf{x}')\right) && (1.3)\\
k(\mathbf{x}, \mathbf{x}') &= \exp\left(k_1(\mathbf{x}, \mathbf{x}')\right) && (1.4)\\
k(\mathbf{x}, \mathbf{x}') &= k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') && (1.5)\\
k(\mathbf{x}, \mathbf{x}') &= k_1(\mathbf{x}, \mathbf{x}') k_2(\mathbf{x}, \mathbf{x}') && (1.6)\\
k(\mathbf{x}, \mathbf{x}') &= k_3\left(\boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}')\right) && (1.7)\\
k(\mathbf{x}, \mathbf{x}') &= \mathbf{x}^{\mathrm{T}} \mathbf{A} \mathbf{x}' && (1.8)\\
k(\mathbf{x}, \mathbf{x}') &= k_a(\mathbf{x}_a, \mathbf{x}'_a) + k_b(\mathbf{x}_b, \mathbf{x}'_b) && (1.9)\\
k(\mathbf{x}, \mathbf{x}') &= k_a(\mathbf{x}_a, \mathbf{x}'_a) k_b(\mathbf{x}_b, \mathbf{x}'_b) && (1.10)
\end{aligned}
$$

where $c > 0$ is a constant, $f(\cdot)$ is any function, $q(\cdot)$ is a polynomial with nonnegative coefficients, $\boldsymbol{\phi}(\mathbf{x})$ is a function from $\mathbf{x}$ to $\mathbb{R}^M$, $k_3(\cdot, \cdot)$ is a valid kernel in $\mathbb{R}^M$, $\mathbf{A}$ is a symmetric positive semidefinite matrix, $\mathbf{x}_a$ and $\mathbf{x}_b$ are variables (not necessarily disjoint) with $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$, and k_a and k_b are valid kernel functions over their respective spaces.

1.3 Slide 11

For kernel PCA, see [Schölkopf et al.(1998)Schölkopf, Smola, and Müller]. For other applications of the kernel trick, see [Mika et al.(1999)Mika, Ratsch, Weston, Scholkopf, and Mullers, Roth and Steinhage(1999), Baudat and Anouar(2000)].

1.4 Slide 13

Schematic illustration of kernel PCA. A data set in the original data space (left-hand plot) is projected by a nonlinear transformation $\boldsymbol{\phi}(\mathbf{x})$ into a feature space (right-hand plot). By performing PCA in the feature space, we obtain the principal components, of which the first is shown in blue and is denoted by the vector $\mathbf{v}_1$. The green lines in feature space indicate the linear projections onto the first principal component, which correspond to nonlinear projections in the original data space. Note that in general it is not possible to represent the nonlinear principal component by a vector in $\mathbf{x}$ space.

1.5 Slide 14

There is a bit more work to be done to account for the fact that $\boldsymbol{\phi}(\mathbf{x}_n)$ need not have zero mean: for details, read Bishop.

1.6 Slide 15

The contours are lines along which the projection onto the corresponding principal component is constant. Note how the first two eigenvectors separate the three clusters, the next three eigenvectors split each of the cluster into halves, and the following three eigenvectors again split the clusters into halves along directions orthogonal to the previous splits.

2 Gaussian Processes

2.1 Slide 6

Squared Exponential. The term b represents a bias that controls the vertical offset of the Gaussian process, while v_0 controls the vertical scale of the process. The squared exponential term captures the idea that vectors that are close in input space should give rise to highly correlated outputs. The a_l parameters allow a different distance measure for each dimension. If a_l is small then the lth input will be downweighted and have little effect on the input: this has a similar effect to Automatic Relevance Determination.

Note that all the parameters in this model must be positive, and so it is convenient to consider the parameter vector in log space:

$$\boldsymbol{\theta} = (\log v_0, \log b, \log a_1, \ldots, \log a_d, \log \sigma_\nu^2). \tag{2.1}$$

Rational Quadratic The parameters v_0, b, and $a_1, \ldots, a_l$ play the same role as in the squared exponential covariance function. The parameter ν controls the rate of decay of the covariance function: larger values of ν give a faster decay and hence a 'rougher' process.

2.2 Slide 8

This result reflects the fact that the two Gaussian sources of randomness, namely that associated with $y(\mathbf{x})$ and that associated with ϵ, are independent and so their covariances simply add.

2.3 Slide 10

The $p(t_{N+1}|\boldsymbol{t}_N)$ distribution is conditioned also on the variables $\mathbf{x}_1, \ldots, \mathbf{x}_N$ and $\mathbf{x}_{N+1}$. However, to keep the notation simple we will not show these conditioning variables explicitly.

2.4 Slide 14

Samples from the ARD prior for Gaussian processes, in which the kernel function is squared exponential with a separate length-scale parameter for each input. The left plot corresponds to $\eta_1 = \eta_2 = 1$, and the right plot corresponds to $\eta_1 = 1$, $\eta_2 = 0.01$.

2.5 Slide 15

In this synthetic dataset, x_1 is sampled uniformly from the range $(0, 1)$ and has a low level of added Gaussian noise, x_2 is a copy of x_1 with a higher level of added noise, and x_3 is sampled randomly from a Gaussian distribution.

The single target variable is given by $t = \sin(2\pi x_1)$ with additive Gaussian noise. Thus x_1 is very relevant for determining the target value, x_2 is of some relevance, while x_3 should in principle be irrelevant.

3 Gaussian Process Latent Variable Model

Python software can be found at

- Neil Lawrence (originator of the model). His group's code `https://github.com/SheffieldML/GPyGPy`.
- `https://pyro.ai/examples/gplvm.html`
- `https://gpflow.readthedocs.io/en/master/notebooks/basics/GPLVM.html`

Please note that I haven't tested any of the packages myself.

3.1 Slide 7

The GP-LVM uses a squared exponential kernel function. The kernel was initialised using PCA to set $\mathbf{X}$. The likelihood was optimised using scaled conjugate gradients.

The greyscales in the GP-LVM plot indicate the precision with which the manifold was expressed in data space for that latent point. The use of a Gaussian process to perform our 'mapping' means that we can express uncertainty about the positions of the points in the data space. For our formulation of the GP-LVM the level of uncertainty is shared across all D dimensions and thus may be visualised in the latent-space.

The errors made by different methods when using the latent space for nearest neighbour classification are shown in Table 3.1.

Table 3.1: Errors made by the different methods when using the latent space for classification.

Method	PCA	GP-LVM	GTM	kernel PCA
Errors	20	4	7	13

3.2 Slide 8

Require: a size for the active set d. A number of iterations, T. Initialise $\mathbf{X}$ through PCA;
for *T iterations* **do**
  Select a new active set using the IVM algorithm.;
  Optimise with respect to the parameters of $\mathbf{K}$ using scaled conjugate gradients.;
  Select a new active set.;
  for *each point not in active set, j* **do**
    Optimise with respect to $\mathbf{x}$ using scaled conjugate gradients.;
  end
end

Algorithm 1: An algorithm for modelling with a GPLVM.

3.3 Slide 9

The MLP kernel is derived by considering a multi-layer perceptron with an infinite number of hidden units.

$$k_{mlp}(\mathbf{x}_i, \mathbf{x}_j) = \theta_{mlp} \sin^{-1}\left(\frac{w\mathbf{x}_i^T\mathbf{x}_j + b}{\sqrt{(w\mathbf{x}_i^T\mathbf{x}_i + b + 1)(w\mathbf{x}_j^T\mathbf{x}_j + b + 1)}}\right) \tag{3.1}$$

This covariance function also leads to smooth functions, but they have an important characteristic that differentiates them from the squared exponential (RBF) kernel: outside regions where the data lies, functions will not fall to zero, but tend to remain at the same value.

The results of the nearest-neighbour classifier in different projects on the full oil dataset are shown in Table 3.2.

Table 3.2: Errors made by the different methods when using the latent space for classification on the full oil dataset (1000 points).

Method	PCA	Sparse GP-LVM (RBF)	GP-LVM (RBF)	Sparse GP-LVM (MLP)	GTM
Errors	162	24	1	14	11

3.4 Slide 10

A 2-D visualisation of a sub-set of 3000 of the digits 0-4 (600 of each digit) from a 16×16 greyscale version of the USPS digit data set. Again we made use of the RBF and the MLP kernel and used the sparse GP-LVM. As well as visualising with the GP-LVM we present visualisations from a GTM and PCA. Comparision with the full GP-LVM model for this data set is not currently practical.

The digit images visualised in the 2-D latent-space. '0' is represented by red crosses; '1': green circles; '2' blue pluses; '3' cyan stars; and '4' magenta squares.

Table 3.3: Errors made by the different methods when using the latent space for classification.

Method	PCA	Sparse GP-LVM (RBF)	Sparse GP-LVM (MLP)	GTM
Errors	780	208	202	158

References

[Baudat and Anouar, 2000] Baudat, G. and F. Anouar 2000. Generalized discriminant analysis using a kernel approach. *Neural computation* **12** (10), 2385–2404.

[Mika *et al.*, 1999] Mika, S., G. Ratsch, J. Weston, B. Scholkopf, and K.-R. Mullers 1999. Fisher discriminant analysis with kernels. In *Neural networks for signal processing IX: Proceedings of the 1999 IEEE signal processing society workshop (cat. no. 98th8468)*, pp. 41–48. Ieee.

[Roth and Steinhage, 1999] Roth, V. and V. Steinhage 1999. Nonlinear discriminant analysis using kernel functions. In S. Solla, T. Leen, and K. Müller (eds), *Advances in Neural Information Processing Systems*, Volume 12. MIT Press.

[Schölkopf *et al.*, 1998] Schölkopf, B., A. Smola, and K.-R. Müller 1998. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation* **10** (5), 1299–1319.