PATTERN RECOGNITION AND MACHINE LEARNING

CHAPTER 6: KERNEL METHODS

Dual Representations

Many linear parametric models can be re-cast into an equivalent 'dual representation' in which the predictions are also based on linear combinations of a kernel function evaluated at the training data points.

$$k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}')$$

From this definition, we see that the kernel is a symmetric function of its arguments so that k(x, x') = k(x', x).

The simplest example of a kernel function is the linear kernel

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^{\mathrm{T}} \mathbf{x}' \qquad \boldsymbol{\phi}(\mathbf{x}) = \mathbf{x}$$

Stationary kernels because are invariant to translations in input space

$$k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} - \mathbf{x}')$$

Homogeneous kernels, also known as radial basis functions, depend only on the magnitude of the distance (typically Euclidean) between the arguments

$$k(\mathbf{x}, \mathbf{x}') = k(\|\mathbf{x} - \mathbf{x}'\|)$$

Dual Representations

Many linear models for regression and classification can be reformulated in terms of a dual representation in which the kernel function arises naturally.

$$J(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left\{\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) - t_n\right\}^2 + \frac{\lambda}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}$$

If we set the gradient of J(**w**) with respect to **w** equal to zero

$$\mathbf{w} = -\frac{1}{\lambda}\sum_{n=1}^{N}\left\{\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) - t_n\right\}\boldsymbol{\phi}(\mathbf{x}_n) = \sum_{n=1}^{N} a_n\boldsymbol{\phi}(\mathbf{x}_n) = \boldsymbol{\Phi}^{\mathrm{T}}\mathbf{a}$$

where Φ is the design matrix, whose nth row is given by $\varphi(x_n)^T$

Here the vector $a = (a_1, \ldots, a_N)^T$ is defined as

$$a_n = -\frac{1}{\lambda}\left\{\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) - t_n\right\}$$

If we substitute $w = \Phi^T a$ into J(**w**), we obtain

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{a} - \mathbf{a}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t} + \frac{1}{2}\mathbf{t}^{\mathrm{T}}\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{a}$$

Dual Representations

We now define the Gram matrix $\mathbf{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}$, which is an N × N symmetric matrix with elements

$$K_{nm} = \boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m)$$

In terms of the Gram matrix, the sum-of-squares error function can be written as

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^{\mathrm{T}}\mathbf{K}\mathbf{K}\mathbf{a} - \mathbf{a}^{\mathrm{T}}\mathbf{K}\mathbf{t} + \frac{1}{2}\mathbf{t}^{\mathrm{T}}\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^{\mathrm{T}}\mathbf{K}\mathbf{a}$$

Setting the gradient of J(**a**) with respect to a to zero, we obtain the following solution

$$\mathbf{a} = (\mathbf{K} + \lambda\mathbf{I}_N)^{-1}\mathbf{t}$$

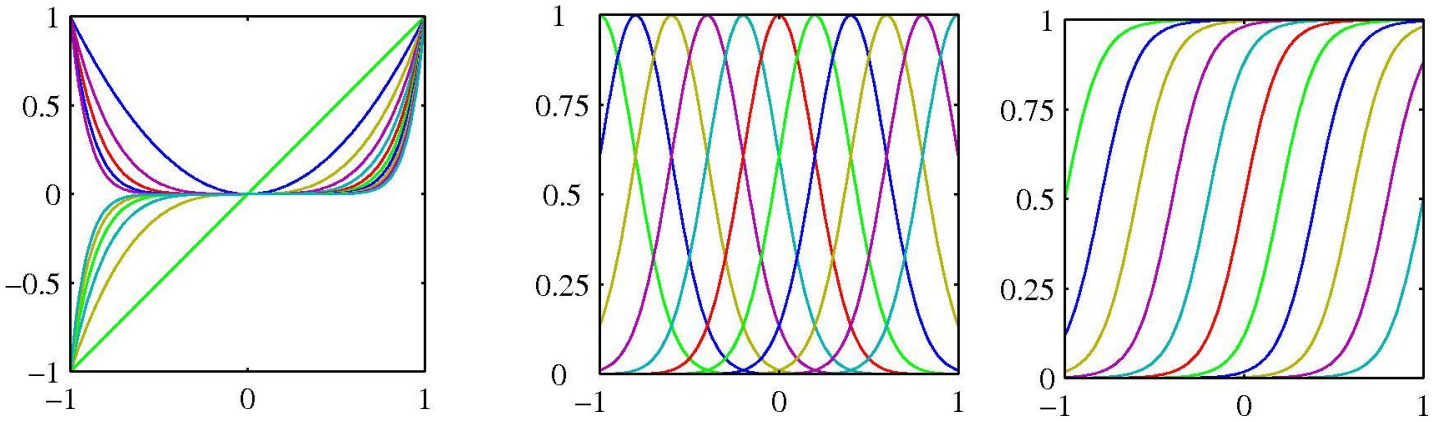
If we substitute this back into the linear regression model, we obtain the following prediction for a new input **x**

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}) = \mathbf{a}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\phi}(\mathbf{x}) = \mathbf{k}(\mathbf{x})^{\mathrm{T}}(\mathbf{K} + \lambda\mathbf{I}_N)^{-1}\mathbf{t}$$

$$k_n(\mathbf{x}) = k(\mathbf{x}_n, \mathbf{x})$$

Constructing Kernels

$$k(x, x') = \boldsymbol{\phi}(x)^{\mathrm{T}} \boldsymbol{\phi}(x') = \sum_{i=1}^{M} \phi_i(x)\phi_i(x')$$



Constructing Kernels

Techniques for Constructing New Kernels.

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

$$
\begin{aligned}
k(\mathbf{x}, \mathbf{x}') &= c k_1(\mathbf{x}, \mathbf{x}') \\
k(\mathbf{x}, \mathbf{x}') &= f(\mathbf{x}) k_1(\mathbf{x}, \mathbf{x}') f(\mathbf{x}') \\
k(\mathbf{x}, \mathbf{x}') &= q\left(k_1(\mathbf{x}, \mathbf{x}')\right) \\
k(\mathbf{x}, \mathbf{x}') &= \exp\left(k_1(\mathbf{x}, \mathbf{x}')\right) \\
k(\mathbf{x}, \mathbf{x}') &= k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') \\
k(\mathbf{x}, \mathbf{x}') &= k_1(\mathbf{x}, \mathbf{x}') k_2(\mathbf{x}, \mathbf{x}') \\
k(\mathbf{x}, \mathbf{x}') &= k_3\left(\boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}')\right) \\
k(\mathbf{x}, \mathbf{x}') &= \mathbf{x}^{\mathrm{T}} \mathbf{A} \mathbf{x}' \\
k(\mathbf{x}, \mathbf{x}') &= k_a(\mathbf{x}_a, \mathbf{x}'_a) + k_b(\mathbf{x}_b, \mathbf{x}'_b) \\
k(\mathbf{x}, \mathbf{x}') &= k_a(\mathbf{x}_a, \mathbf{x}'_a) k_b(\mathbf{x}_b, \mathbf{x}'_b)
\end{aligned}
$$

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.

Constructing Kernels

A commonly used kernel often called a “Gaussian kernel” is defined as

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\|\mathbf{x} - \mathbf{x}'\|^2 / 2\sigma^2\right)$$

Another powerful approach to the construction of kernels starts from a probabilistic generative model, which allows us to apply generative models in a discriminative setting. Given a generative model p(**x**) we can define a kernel by

$$k(\mathbf{x}, \mathbf{x}') = p(\mathbf{x})p(\mathbf{x}').$$

Constructing Kernels

An alternative technique for using generative models to define kernel functions is known as the Fisher kernel. Consider a parametric generative model p(x|θ) where θ denotes the vector of parameters. The goal is to find a kernel that measures the similarity of two input vectors x and x induced by the generative model.

In particular, they consider the Fisher score $\mathbf{g}(\boldsymbol{\theta}, \mathbf{x}) = \nabla_{\boldsymbol{\theta}} \ln p(\mathbf{x}|\boldsymbol{\theta})$ from which the Fischer kernel is defined as

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{g}(\boldsymbol{\theta}, \mathbf{x})^{\mathrm{T}} \mathbf{F}^{-1} \mathbf{g}(\boldsymbol{\theta}, \mathbf{x}').$$

Here F is the Fisher information matrix, given by

$$\mathbf{F} = \mathbb{E}_{\mathbf{x}} \left[\mathbf{g}(\boldsymbol{\theta}, \mathbf{x}) \mathbf{g}(\boldsymbol{\theta}, \mathbf{x})^{\mathrm{T}} \right]$$

In practice, it is often infeasible to evaluate the Fisher information matrix. One approach is simply to replace the expectation in the definition of the Fisher information with the sample average:

$$\mathbf{F} \simeq \frac{1}{N} \sum_{n=1}^{N} \mathbf{g}(\boldsymbol{\theta}, \mathbf{x}_n) \mathbf{g}(\boldsymbol{\theta}, \mathbf{x}_n)^{\mathrm{T}}.$$

Radial Basis Function Networks

Radial basis functions have the property that each basis function depends only on the radial distance (typically Euclidean) from a center μ_j , so that $\varphi_j(x) = h(||x - \mu_j||)$

Given a set of input vectors $\{x_1, \ldots, x_N\}$ along with corresponding target values $\{t_1, \ldots, t_N\}$, the goal is to find a smooth function f(x) that fits every target value exactly, so that $f(x_n) = t_n$ for n = 1, . . . , N.

$$f(\mathbf{x}) = \sum_{n=1}^{N} w_n h(\|\mathbf{x} - \mathbf{x}_n\|)$$

The values of the coefficients $\{w_n\}$ are found by least squares, and because there are the same number of coefficients as there are constraints, the result is a function that fits every target value exactly.

Radial Basis Function Networks

In ML applications, however, the target values are generally noisy, and exact interpolation is undesirable because this corresponds to an over-fitted solution.

If the noise on the input variable x is described by a variable ξ having a distribution v(ξ), then the sum-of-squares error function becomes

$$E = \frac{1}{2}\sum_{n=1}^{N}\int \{y(\mathbf{x}_n + \boldsymbol{\xi}) - t_n\}^2 \nu(\boldsymbol{\xi})\,\mathrm{d}\boldsymbol{\xi}$$

$$y(\mathbf{x}_n) = \sum_{n=1}^{N} t_n h(\mathbf{x} - \mathbf{x}_n)$$

$$h(\mathbf{x} - \mathbf{x}_n) = \frac{\nu(\mathbf{x} - \mathbf{x}_n)}{\sum_{n=1}^{N}\nu(\mathbf{x} - \mathbf{x}_n)}$$

We see that there is one basis function centered on every data point, and it is known as the Nadaraya-Watson model.

Gaussian Processes

Consider a model defined in terms of a linear combination of M fixed basis functions given by the elements of the vector φ(x) so that

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})$$

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})$$

We are therefore interested in the joint distribution of the function values $y(x_1), \ldots, y(x_N)$, which we denote by the vector **y** with elements $y_n = y(x_n)$ for n = 1, . . . , N.

$$\mathbf{y} = \boldsymbol{\Phi}\mathbf{w}$$

$$\begin{aligned}\mathbb{E}[\mathbf{y}] &= \boldsymbol{\Phi}\mathbb{E}[\mathbf{w}] = \mathbf{0} \\ \mathrm{cov}[\mathbf{y}] &= \mathbb{E}\left[\mathbf{y}\mathbf{y}^{\mathrm{T}}\right] = \boldsymbol{\Phi}\mathbb{E}\left[\mathbf{w}\mathbf{w}^{\mathrm{T}}\right]\boldsymbol{\Phi}^{\mathrm{T}} = \frac{1}{\alpha}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} = \mathbf{K}\end{aligned}$$

where **K** is the Gram matrix with elements

$$K_{nm} = k(\mathbf{x}_n, \mathbf{x}_m) = \frac{1}{\alpha}\boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_m)$$

and k(x, x') is the kernel function.

Gaussian Processes

In order to apply Gaussian process models to the problem of regression, we need to take account of the noise on the observed target values, which are given by $t_n = y_n + \varepsilon_n$ where $y_n = y(x_n)$, and ε_n is a random noise variable whose value is chosen independently for each observation n.

$$p(t_n|y_n) = \mathcal{N}(t_n|y_n, \beta^{-1})$$

Because the noise is independent for each data point, the joint distribution of the target values $\mathbf{t} = (t_1, \ldots, t_N)^T$ conditioned on the values of $\mathbf{y} = (y_1, \ldots, y_N)^T$ is given by an isotropic Gaussian of the form

$$p(\mathbf{t}|\mathbf{y}) = \mathcal{N}(\mathbf{t}|\mathbf{y}, \beta^{-1}\mathbf{I}_N)$$

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K})$$

The kernel function that determines K is typically chosen to express the property that, for points x_n and x_m that are similar, the corresponding values $y(x_n)$ and $y(x_m)$ will be more strongly correlated than for dissimilar points

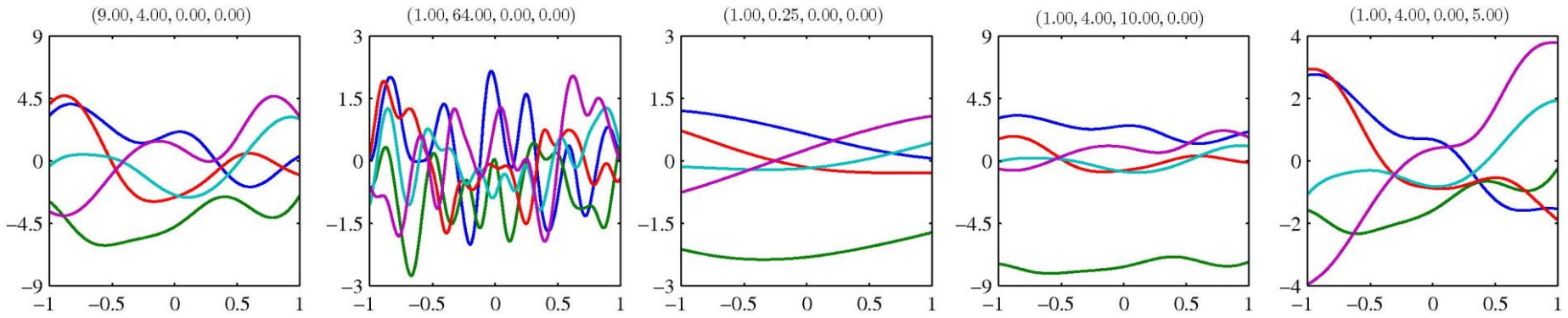
Gaussian Processes

$$p(\mathbf{t}) = \int p(\mathbf{t}|\mathbf{y})p(\mathbf{y})\,\mathrm{d}\mathbf{y} = \mathcal{N}(\mathbf{t}|\mathbf{0}, \mathbf{C})$$

$$C(\mathbf{x}_n, \mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m) + \beta^{-1}\delta_{nm}$$

One widely used kernel function for Gaussian process regression is given by the exponential of a quadratic form, with the addition of constant and linear terms to give

$$k(\mathbf{x}_n, \mathbf{x}_m) = \theta_0 \exp\left\{-\frac{\theta_1}{2}\|\mathbf{x}_n - \mathbf{x}_m\|^2\right\} + \theta_2 + \theta_3 \mathbf{x}_n^{\mathrm{T}}\mathbf{x}_m$$



Gaussian Processes

Our goal in regression, however, is to make predictions of the target variables for new inputs, given a set of training data.
This requires that we evaluate the predictive distribution $p(t_{N+1}|t_N)$.

$$p(\mathbf{t}_{N+1}) = \mathcal{N}(\mathbf{t}_{N+1}|\mathbf{0}, \mathbf{C}_{N+1})$$

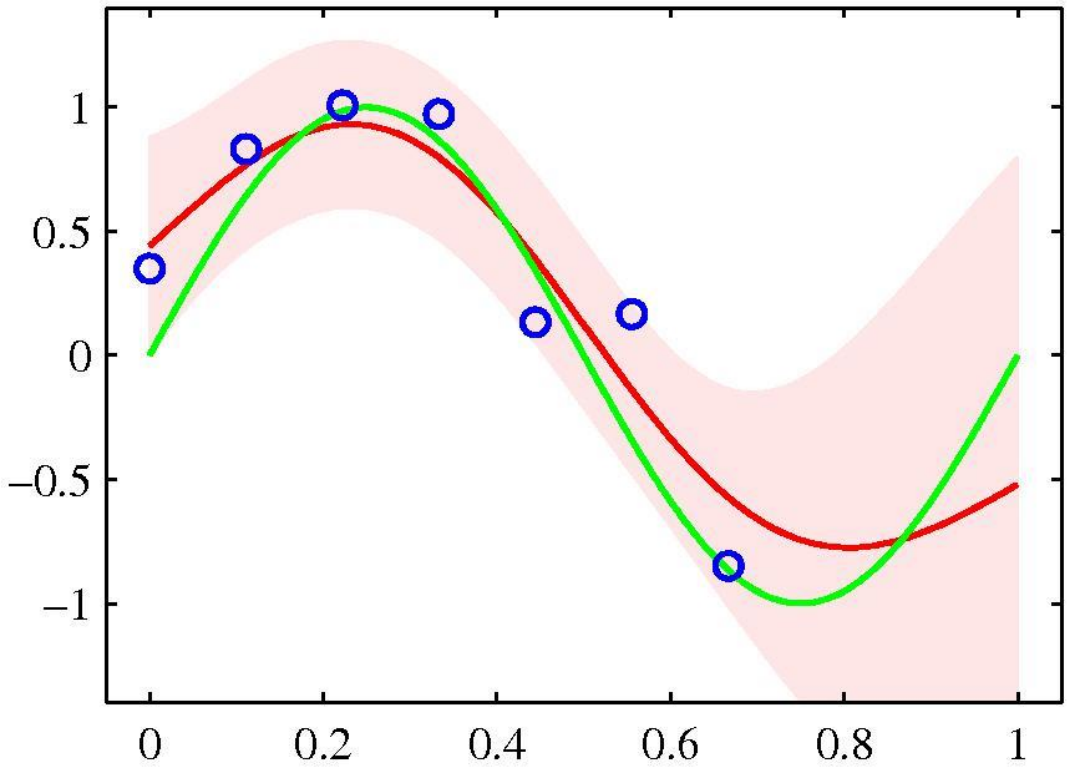
$$\mathbf{C}_{N+1} = \begin{pmatrix} \mathbf{C}_N & \mathbf{k} \\ \mathbf{k}^{\mathrm{T}} & c \end{pmatrix}$$

where $c = k(x_{N+1}, x_{N+1}) + \beta^{-1}$

The predictive distribution $p(t_{N+1}|t_N)$ is a Gaussian distribution with mean and covariance given by

$$\begin{aligned} m(\mathbf{x}_{N+1}) &= \mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{t} \\ \sigma^2(\mathbf{x}_{N+1}) &= c - \mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{k}. \end{aligned}$$

Gaussian Processes



Gaussian Processes

The predictions of a Gaussian process model will depend, in part, on the choice of covariance function.

Techniques for learning the hyperparameters are based on the evaluation of the likelihood function p(t|θ) where θ denotes the hyperparameters of the Gaussian process model.

$$\ln p(\mathbf{t}|\boldsymbol{\theta}) = -\frac{1}{2}\ln|\mathbf{C}_N| - \frac{1}{2}\mathbf{t}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{t} - \frac{N}{2}\ln(2\pi)$$

For nonlinear optimization, we also need the gradient of the log likelihood function with respect to the parameter vector θ.

$$\frac{\partial}{\partial\theta_i}\ln p(\mathbf{t}|\boldsymbol{\theta}) = -\frac{1}{2}\mathrm{Tr}\left(\mathbf{C}_N^{-1}\frac{\partial\mathbf{C}_N}{\partial\theta_i}\right) + \frac{1}{2}\mathbf{t}^{\mathrm{T}}\mathbf{C}_N^{-1}\frac{\partial\mathbf{C}_N}{\partial\theta_i}\mathbf{C}_N^{-1}\mathbf{t}$$

Gaussian Processes

In a probabilistic approach to classification, our goal is to model the posterior probabilities of the target variable for a new input vector, given a set of training data. These probabilities must lie in the interval (0, 1), whereas a Gaussian process model makes predictions that lie on the entire real axis.

Consider first the two-class problem with a target variable $t \in \{0, 1\}$. If we define a Gaussian process over a function $a(x)$ and then transform the function using a logistic sigmoid $y = \sigma(a)$, then we will obtain a non-Gaussian stochastic process over functions $y(x)$ where $y \in (0, 1)$.