PATTERN RECOGNITION AND MACHINE LEARNING

CHAPTER 6: KERNEL METHODS

Previous Chapters

- Presented linear models for regression and classification
- Focused to learn y(x, w)
  - Training data used to learn the adaptive parameters w either as a point estimate or a posterior distribution
  - Training data is then discarded and the predictions for new data is done based on the learned parameter vector w
- Same approach is used in nonlinear models such as NN

Previous Chapters

- Other approach: keep the training data or part of it and use it for deciding on the new data:
  - Example: nearest neighbor (NN), k-NN, etc.
  - **Memory-based approaches** → need a metric to compute similarity between two data points in the input space
  - Generally, they are faster to train, slower to make predictions for new data

Remember Kernels?

- Linear parametric models can be re-cast into an equivalent ‘dual representation’
- The predictions are also based on linear combinations of a *kernel function* evaluated at the training data points
- Given a non-linear *feature space* mapping ϕ(x), the kernel function is given by:

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

Kernel Functions

- Are symmetric: $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$.
- Introduced in the 1960s, neglected for many years, re-introduced in ML in 1990s by inventing Support Vector Machines (SVMs)
- Simplest example of kernel: identity mapping of the feature space $\phi(\mathbf{x}) = \mathbf{x}$
- It results the linear kernel:

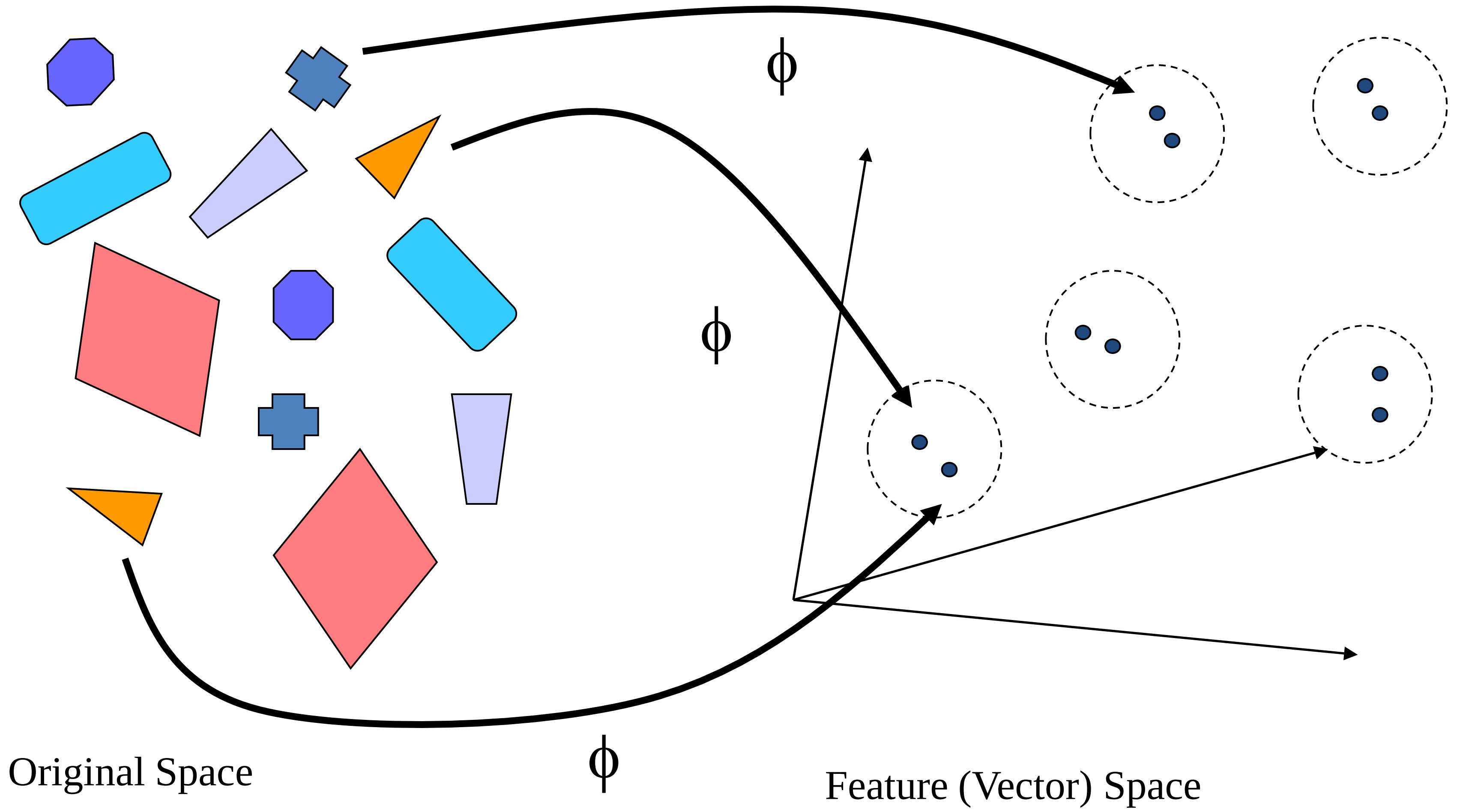
$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^{\mathrm{T}}\mathbf{x}'$$

- The kernel can thus be formulated as an inner product in the feature space

Kernel Methods – Intuitive Idea

- Find a mapping ϕ such that, in the new space, problem solving is easier (e.g. linear)
- The *kernel* represents the similarity between two objects (documents, terms, …), defined as the dot-product in this new vector space
- But the mapping is left implicit
- Easy generalization of a lot of dot-product (or distance) based pattern recognition algorithms

Kernel Methods: The Mapping



Kernel – A more formal definition

- But still informal
- A kernel k(x, y):
  - is a similarity measure
  - defined by an implicit mapping ϕ,
  - from the original space to a vector space (feature space)
  - such that: $k(x,y)=\phi(x)\bullet\phi(y)$
- This similarity measure and the mapping include:
  - Invariance or other a priori knowledge
  - Simpler structure (linear representation of the data)
  - The class of functions the solution is taken from
  - Possibly infinite dimension (hypothesis space for learning)
  - … but still computational efficiency when computing $k(x,y)$

Usual Kernels

- Stationary kernels: use a function of only the difference between the arguments
  - Invariable to translations in the input space

$$k(x, y) = k(x - y)$$

- Homogeneous kernels or radial basis functions: depend only on the magnitude of the distance between the arguments

$$k(x, y) = k(\|x - y\|)$$

Dual Representation

- Many linear models for regression and classification can be reformulated in terms of a *dual representation* in which the *kernel function arises naturally*
- Remember the regularized sum-of-squares error for a linear regression model:

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

- We want to minimize the error

Dual Representation

- Setting 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 $\boldsymbol{\Phi}$ is the design matrix, whose n^{th} row is given by $\boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}}$. Here the vector $\mathbf{a} = (a_1, \ldots, a_N)^{\mathrm{T}}$, and we have defined

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

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}$$

Dual Representation

- Reformulate the sum-of-squares error in terms of the vector **a** instead of **w**
- **=> Dual Representation:**

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

$$\mathbf{t} = (t_1, \ldots, t_N)^{\mathrm{T}}$$

- Define the **Gram** matrix: $\mathrm{K} = \Phi\Phi^{\mathrm{T}}$
  - NxN symetric matrix with elements of the form:

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

Dual Representation

- Gram matrix uses the kernel function
- The error function using the Gram matrix:

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

- The gradient of J(a) is equal to zero when:

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

- Thus the linear regression model for a new data point x:

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

- Where k(x) is a vector:

$$k(x) = [k(x_1, x) \quad k(x_2, x) \quad \ldots \quad k(x_N, x)]$$

Dual Representation - Conclusions

$$\mathbf{w}_{\mathrm{ML}} = \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}$$

- Either compute $\mathbf{w}_{\mathbf{ML}}$ or **a**
- The dual formulation allows the solution to the least-squares problem to be expressed entirely in terms of the kernel function **k(x, x')**
- The solution for **a** can be expressed as a linear combination of the elements of **ϕ(x)**
- We can recover the original formulation in terms of the parameter vector **w**
- **The prediction at x is given by a linear combination of the target values from the training set**

Dual Representation - Conclusions

- In the dual representation, we determine the parameter vector **a** by inverting a NxN matrix
- In the original parameter space, we determine the parameter vector w by inverting a MxM matrix
- Usually, N >> M
  - Disadvantage: The dual representation seems more difficult to compute
  - Advantage: The dual representation can be expressed by using only the kernel function

Dual Representation - Conclusions

- Work directly in terms of kernels and avoid the explicit introduction of the feature vector **$\phi(x)$*,*** which allows us implicitly to use feature spaces of high, even infinite, dimensionality.
- The existence of a dual representation based on the Gram matrix is a property of many linear models, including the perceptron

Constructing Kernels

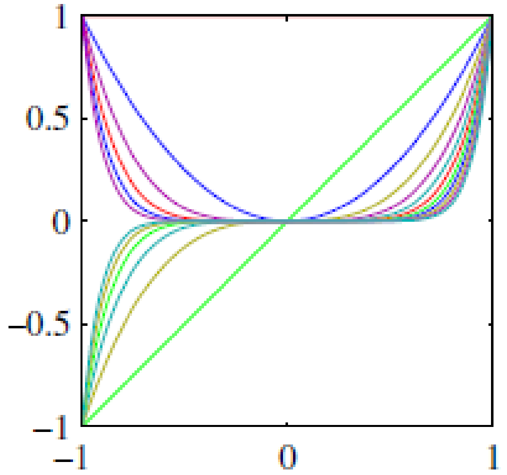
- To exploit kernel substitution, we need to construct valid kernel functions
- First approach:
  - Choose a feature space mapping **$\phi(x)$**
  - Use it to construct the corresponding kernel:

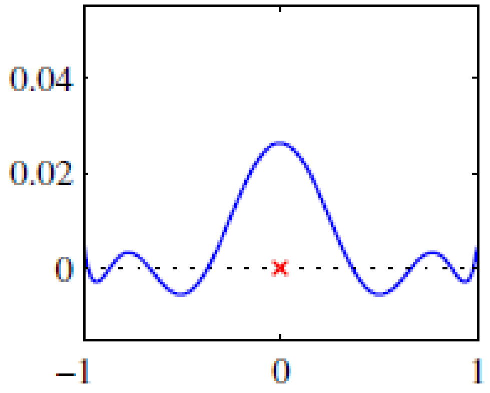
$$k(x, x') = \phi(x)^{\mathrm{T}}\phi(x') = \sum_{i=1}^{M} \phi_i(x)\phi_i(x')$$

  - Where **$\phi_i(x)$** are the basis functions

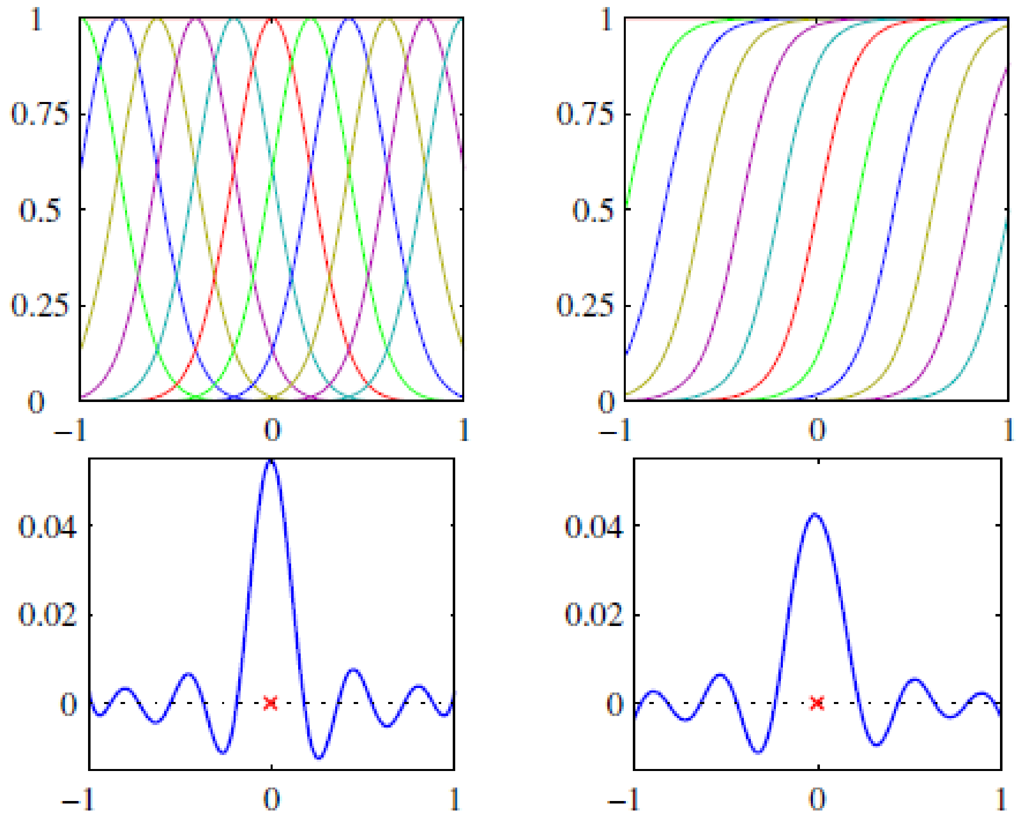
Examples

- Polynomial basis functions
- k(x, x’) for x’=0 and various values of x





Examples



Constructing Kernels

- Alternative approach:
  - Construct **valid** kernel functions directly
  - DEF1! If it corresponds to a scalar product in some (perhaps infinite dimensional) feature space
  - DEF2! If there exists a mapping ϕ into a vector space (with a dot-product) such that *k* can be expressed as $k(x,y)=\phi(x)\bullet\phi(y)$

Simple Example

- Consider the kernel function:

$$k(\mathbf{x}, \mathbf{z}) = \left(\mathbf{x}^{\mathrm{T}}\mathbf{z}\right)^2$$

- Consider a particular example: 2-dimensional input space **x**=(x_1, x_2)
- Expand the terms to find the nonlinear feature mapping: $\phi(\mathbf{x}) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)^{\mathrm{T}}$

$$\begin{aligned} k(\mathbf{x}, \mathbf{z}) &= \left(\mathbf{x}^{\mathrm{T}}\mathbf{z}\right)^2 = (x_1z_1 + x_2z_2)^2 \\ &= x_1^2z_1^2 + 2x_1z_1x_2z_2 + x_2^2z_2^2 \\ &= (x_1^2, \sqrt{2}x_1x_2, x_2^2)(z_1^2, \sqrt{2}z_1z_2, z_2^2)^{\mathrm{T}} \\ &= \phi(\mathbf{x})^{\mathrm{T}}\phi(\mathbf{z}). \end{aligned}$$

Simple Example

- Kernel maps from a 2-dimensional space to a 3-dimensional space that comprises of all possible second order terms (weighted)

Valid Kernel Functions

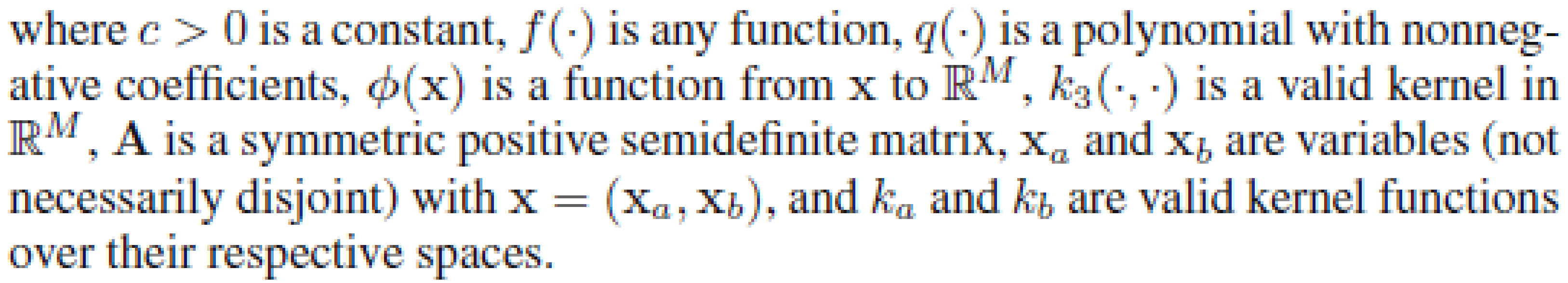
- Need a simpler way to test whether a function constitutes a valid kernel without having to construct the function **ϕ(x)** explicitly
- Necessary and sufficient condition for **k(x, x')** to be a valid kernel is to be symmetric and the Gram matrix **K** to be **positive semidefinite** for all possible choices of the set $\{x_n\}$
- Positive semidefinite matrix M **if** $z^T Mz >= 0$ for all non-zero vectors *z* with real entries

Constructing New Kernels

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

Constructing New Kernels

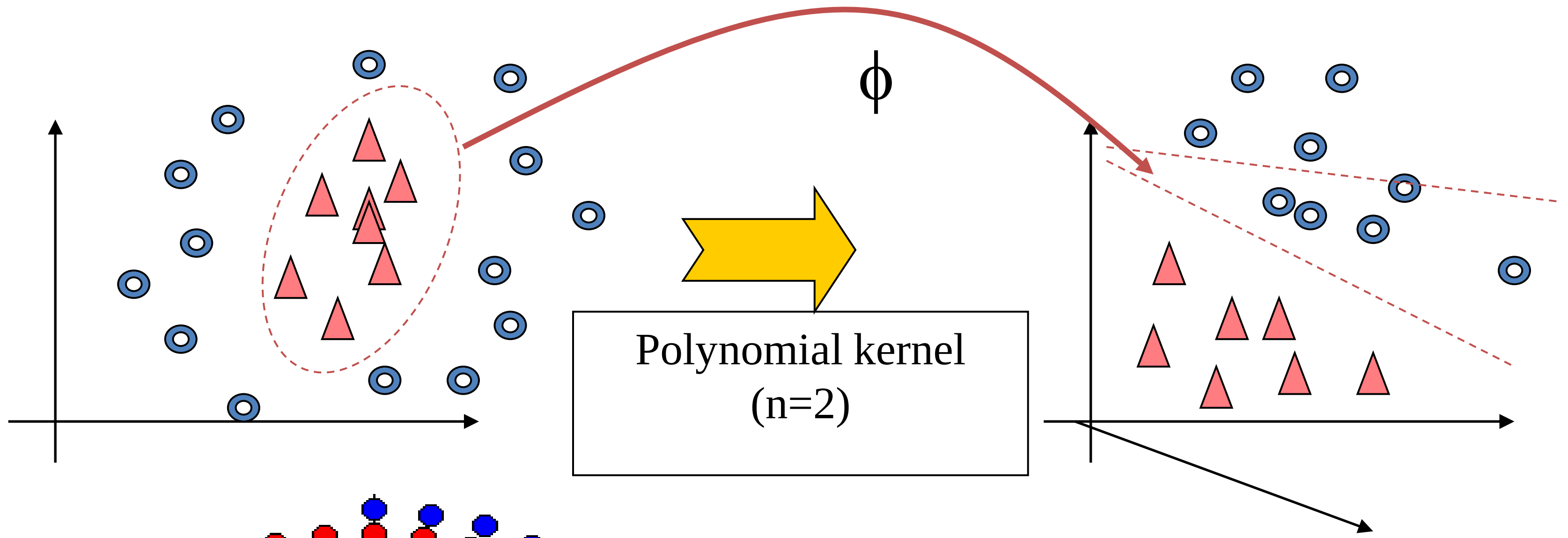
- Given valid kernels **k_1(x, x')** and **k_2(x, x')**



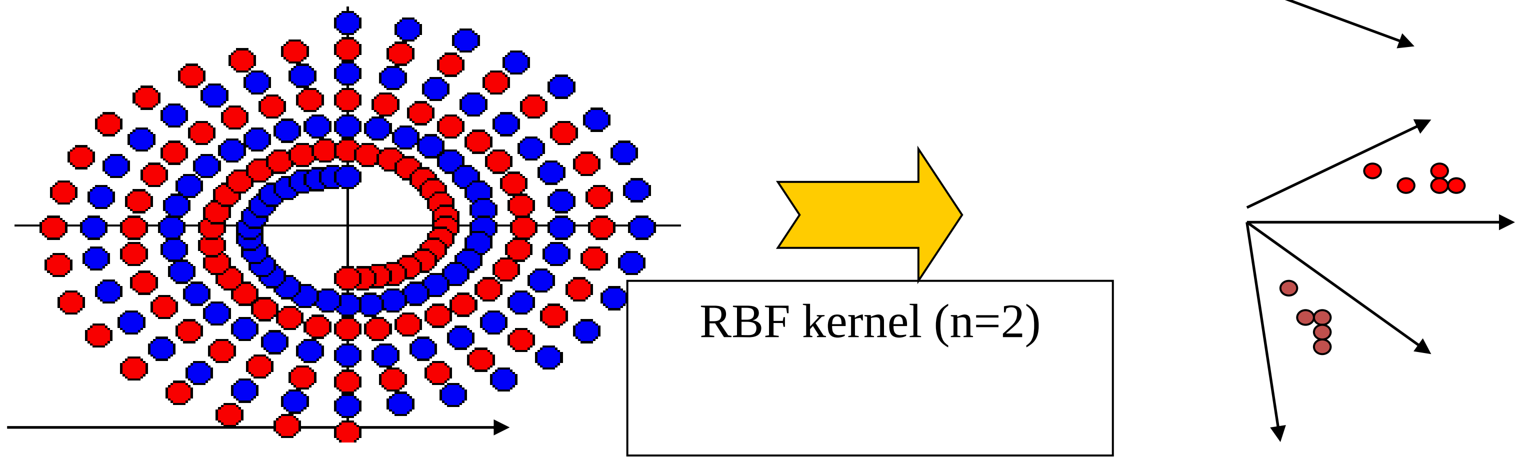
where $c > 0$ is a constant, $f(\cdot)$ is any function, $q(\cdot)$ is a polynomial with nonnegative coefficients, $\phi(\mathbf{x})$ is a function from $\mathbf{x}$ to $\mathbb{R}^M$, $k_3(\cdot, \cdot)$ is a valid kernel in $\mathbb{R}^M$, $\mathbf{A}$ is a symmetric positive semidefinite matrix, $\mathbf{x}_a$ 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.

- The kernel that we use should correctly express the similarity between x and x' according to the intended application
- Wide domain called "KERNEL ENGINEERING"

Examples of Kernels



Polynomial kernel (n=2)



RBF kernel (n=2)

Other Examples of Kernels

- All 2nd order terms+linear terms+constants:

$$\left(\mathbf{x}^{\mathrm{T}}\mathbf{x}' + c\right)^2$$

- All monomials of order M:

$$k(\mathbf{x}, \mathbf{x}') = \left(\mathbf{x}^{\mathrm{T}}\mathbf{x}'\right)^M$$

- All terms up to degree M:

$$k(\mathbf{x}, \mathbf{x}') = \left(\mathbf{x}^{\mathrm{T}}\mathbf{x}' + c\right)^M$$

- Consider what happens if x and x’ are two images and we use the second kernel

Other Examples of Kernels

=> The kernel represents a particular weighted sum of all possible products of M pixels in the first image with M pixels in the second image

Gaussian Kernel

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

- It is not a probability density
- Is valid taking into account the 2nd and 4th properties and because:

$$\|\mathbf{x}-\mathbf{x}'\|^2 = \mathbf{x}^{\mathrm{T}}\mathbf{x} + (\mathbf{x}')^{\mathrm{T}}\mathbf{x}' - 2\mathbf{x}^{\mathrm{T}}\mathbf{x}'$$

- Thus, it is derived from the linear kernel:

$$k(\mathbf{x},\mathbf{x}') = \exp\left(-\mathbf{x}^{\mathrm{T}}\mathbf{x}/2\sigma^2\right)\exp\left(\mathbf{x}^{\mathrm{T}}\mathbf{x}'/\sigma^2\right)\exp\left(-(\mathbf{x}')^{\mathrm{T}}\mathbf{x}'/2\sigma^2\right)$$

The feature vector that corresponds to the Gaussian kernel has infinite dimensionality

Gaussian Kernel

- The linear kernel can be replaced by any nonlinear kernel, resulting:

$$k(\mathrm{x}, \mathrm{x}') = \exp\left\{-\frac{1}{2\sigma^2}\left(\kappa(\mathrm{x}, \mathrm{x}) + \kappa(\mathrm{x}', \mathrm{x}') - 2\kappa(\mathrm{x}, \mathrm{x}')\right)\right\}$$

Kernels for Symbolic Data

- Kernels can be extended to inputs that are symbolic, rather than simply vectors of real numbers
- Kernel functions can be defined over objects as diverse as graphs, sets, strings, and text documents
- Consider a simple kernel over sets:

$$k(A_1, A_2) = 2^{|A_1 \cap A_2|}$$

Kernels for Generative Models

- Given a generative model p(x):

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

- Valid kernel: inner product in the 1D feature space defined by p(x)
- Two inputs are similar if they both have high probabilities
- Can be extended to (where **i** is a considered as a latent variable):

$$k(\mathbf{x}, \mathbf{x}') = \sum_i p(\mathbf{x}|i)p(\mathbf{x}'|i)p(i)$$

- Kernels for HMM:

$$k(\mathbf{X}, \mathbf{X}') = \sum_{\mathbf{Z}} p(\mathbf{X}|\mathbf{Z})p(\mathbf{X}'|\mathbf{Z})p(\mathbf{Z})$$

Radial Basis Function Networks

- **Radial basis functions -** each basis function depends only on the radial distance (typically Euclidean) from a centre

$$\phi_j(\mathbf{x}) = h(\|\mathbf{x} - \boldsymbol{\mu}_j\|)$$

- Used for exact interpolation:

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

- Because the data in ML are generally noisy, exact interpolation is not very useful

Radial Basis Function Networks

- However, when using regularization, the solution no longer interpolates the training data exactly
- RBF are also useful when the input variables are noisy, not the target
- We have the noise on x described by a variable ξ, with distribution ν(ξ), the sum-of-squares error becomes:

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

- Results:

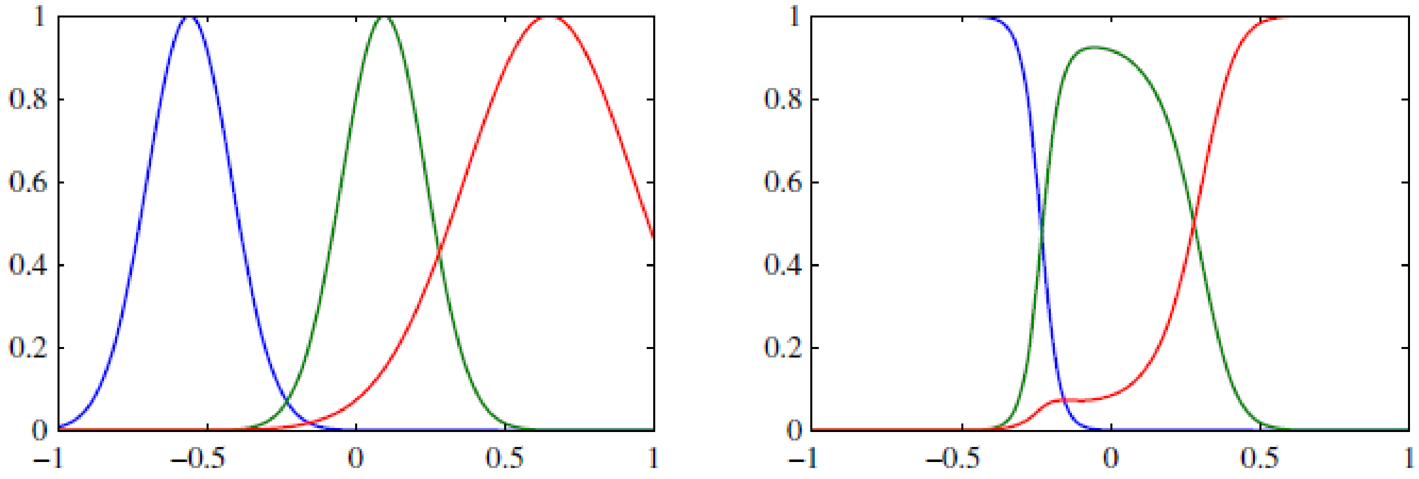
$$y(\mathrm{x}_n) = \sum_{n=1}^{N} t_n h(\mathrm{x} - \mathrm{x}_n) \qquad h(\mathrm{x} - \mathrm{x}_n) = \frac{\nu(\mathrm{x} - \mathrm{x}_n)}{\sum_{n=1}^{N} \nu(\mathrm{x} - \mathrm{x}_n)}.$$

Radial Basis Function Networks

$\Rightarrow$ Nadaraya-Watson model

- Uses normalized radial functions as basis functions if $\nu(\xi) = || \xi ||$
- Normalization is sometimes used in practice as it avoids having regions of input space where all of the basis functions take small values, which would necessarily lead to predictions in such regions that are either small or controlled purely by the bias parameter

Normalization of Basis Functions



Nadaraya-Watson Model

$$p(\mathbf{x}, t) = \frac{1}{N} \sum_{n=1}^{N} f(\mathbf{x} - \mathbf{x}_n, t - t_n)$$

- One component density function centered on each data point

$$\begin{aligned} y(\mathbf{x}) &= \mathbb{E}[t|\mathbf{x}] = \int_{-\infty}^{\infty} t p(t|\mathbf{x})\,\mathrm{d}t \\ &= \frac{\int t p(\mathbf{x}, t)\,\mathrm{d}t}{\int p(\mathbf{x}, t)\,\mathrm{d}t} \\ &= \frac{\sum_n \int t f(\mathbf{x} - \mathbf{x}_n, t - t_n)\,\mathrm{d}t}{\sum_m \int f(\mathbf{x} - \mathbf{x}_m, t - t_m)\,\mathrm{d}t}. \end{aligned}$$

Nadaraya-Watson Model

$$\begin{aligned} y(\mathbf{x}) &= \frac{\sum_n g(\mathbf{x}-\mathbf{x}_n)t_n}{\sum_m g(\mathbf{x}-\mathbf{x}_m)} \\ &= \sum_n k(\mathbf{x},\mathbf{x}_n)t_n \end{aligned}$$

- m, n = 1 .. N
- Kernel function:

$$k(\mathbf{x},\mathbf{x}_n) = \frac{g(\mathbf{x}-\mathbf{x}_n)}{\sum_m g(\mathbf{x}-\mathbf{x}_m)}$$

$$g(\mathbf{x}) = \int_{-\infty}^{\infty} f(\mathbf{x},t)\,\mathrm{d}t.$$

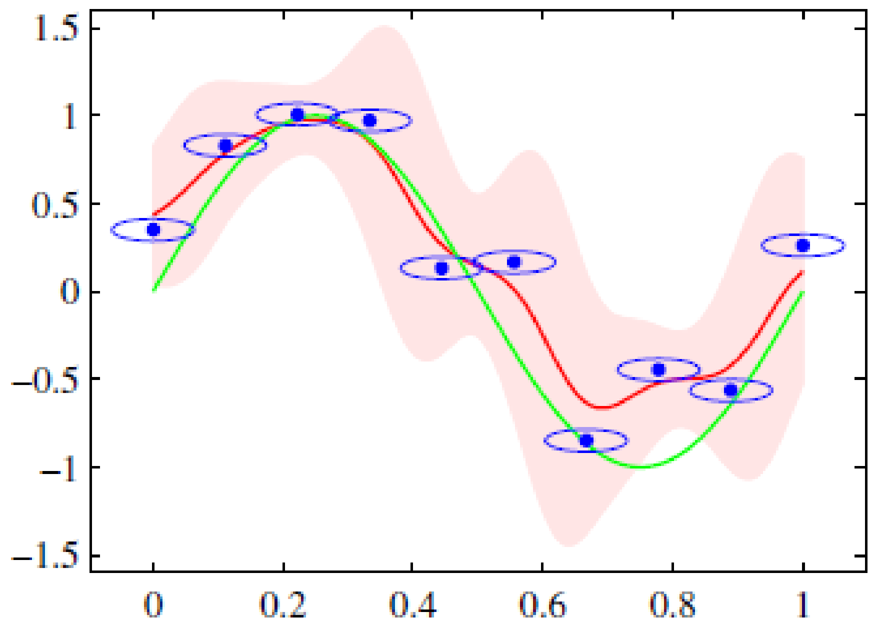
Nadaraya-Watson Model

- Also called **kernel regression**
- For a localized kernel function, it has the property of giving more weight to the data points $\mathbf{x}_n$ that are close to $\boldsymbol{x}$
- The model defines not only a conditional expectation but also a full conditional distribution:

$$p(t|\mathrm{x}) = \frac{p(t,\mathrm{x})}{\int p(t,\mathrm{x})\,\mathrm{d}t} = \frac{\sum_n f(\mathrm{x}-\mathrm{x}_n, t-t_n)}{\sum_m \int f(\mathrm{x}-\mathrm{x}_m, t-t_m)\,\mathrm{d}t}$$

Example

- Isotropic Gaussian kernels centered around the data points defined by $z_n = (x_n, t_n)$



Gaussian Processes

- We have seen kernels as a dual model for a non-probabilistic model for regression
- Extend kernels to probabilistic discriminative models
- In linear models for regression, we have introduced a prior distribution over **w**
- Given the training data set, we evaluated the posterior distribution over **w** => posterior distribution over the regression functions => predictive distribution p(t|x) for new input **x**

Gaussian Processes

- Dispense with the parametric model
- Define a prior probability distribution over functions directly
- Might seem difficult to work with a distribution over the uncountable infinite space of functions
- However, for a finite training set, we only need to consider the values of the function at the discrete set of input values $\mathbf{x}_n$ corresponding to the training set and test set data points, and so in practice we can work in a finite space

Revisiting Linear Regression

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

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

$$\mathbf{y} = \mathbf{\Phi}\mathbf{w} \qquad y_n = y(\mathbf{x}_n)$$

- Because we have a distribution over **w** => distribution over **y(x)** => distribution over $\mathbf{y_1}$=y(x1), … $\mathbf{y_n}$ (the elements of the vector **y**)
- **y** is a linear combination of Gaussian distributed variables given by the elements of **w** and hence is itself Gaussian

Revisiting Linear Regression

- Thus:

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

- Gram matrix and kernel function:

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

- This model provides us with a particular example of a Gaussian process

Gaussian Processes - Definition

- **Gaussian process** is defined as a probability distribution over functions ***y(x)*** such that the set of values of ***y(x)*** evaluated at an arbitrary set of points $\boldsymbol{x_1, \ldots, x_N}$ jointly have a Gaussian distribution
- **x** – 2D => Gaussian random field
- Generally, **stochastic process** ***y(x)*** is specified by giving the joint probability distribution for any finite set of values $\boldsymbol{y(x_1), \ldots, y(x_N)}$ in a consistent manner

Gaussian Processes - Definition

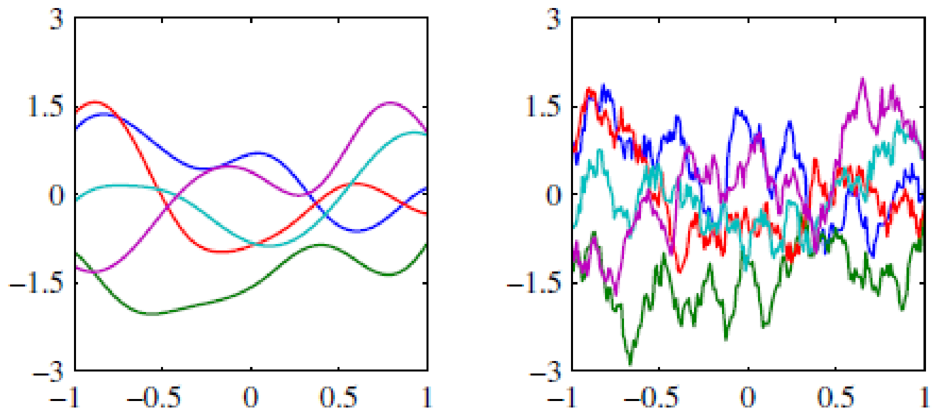
- Gaussian stochastic processes - the joint distribution over N variables $y_1, \ldots, y_N$ is specified completely by the mean and the covariance
- Usually, we do not have any prior information about the **mean** of y(x), so we'll take it to be zero
- The specification of the Gaussian process is then completed by giving the **covariance** of y(x) evaluated at any two values of x, which is given by the kernel function:

$$\mathbb{E}\left[y(\mathbf{x}_n)y(\mathbf{x}_m)\right] = k(\mathbf{x}_n, \mathbf{x}_m)$$

-

Gaussian Processes - Definition

- We can also define the kernel function directly, rather than indirectly through a choice of basis function
- Gaussian kernel vs exponential kernel $k(x, x') = \exp\left(-\theta\,|x - x'|\right)$



Gaussian Processes for Regression

- Take into account the noise on the target:

$$t_n = y_n + \epsilon_n$$

- Random noise under a Gaussian distribution:

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

- Because the noise independent on each data point, the joint distribution is still Gaussian (n dimensions):

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

- Because y(x) is a Gaussian process:

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

- Where the kernel is chosen such that for similar point x_n and x_m, the corresponding values $y(x_n)$ and $y(x_m)$ are strongly correlated

Gaussian Processes for Regression

- Similarity depends on the application
- Using the previous information, the marginal probability is:

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

- Where the covariance matrix is:

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

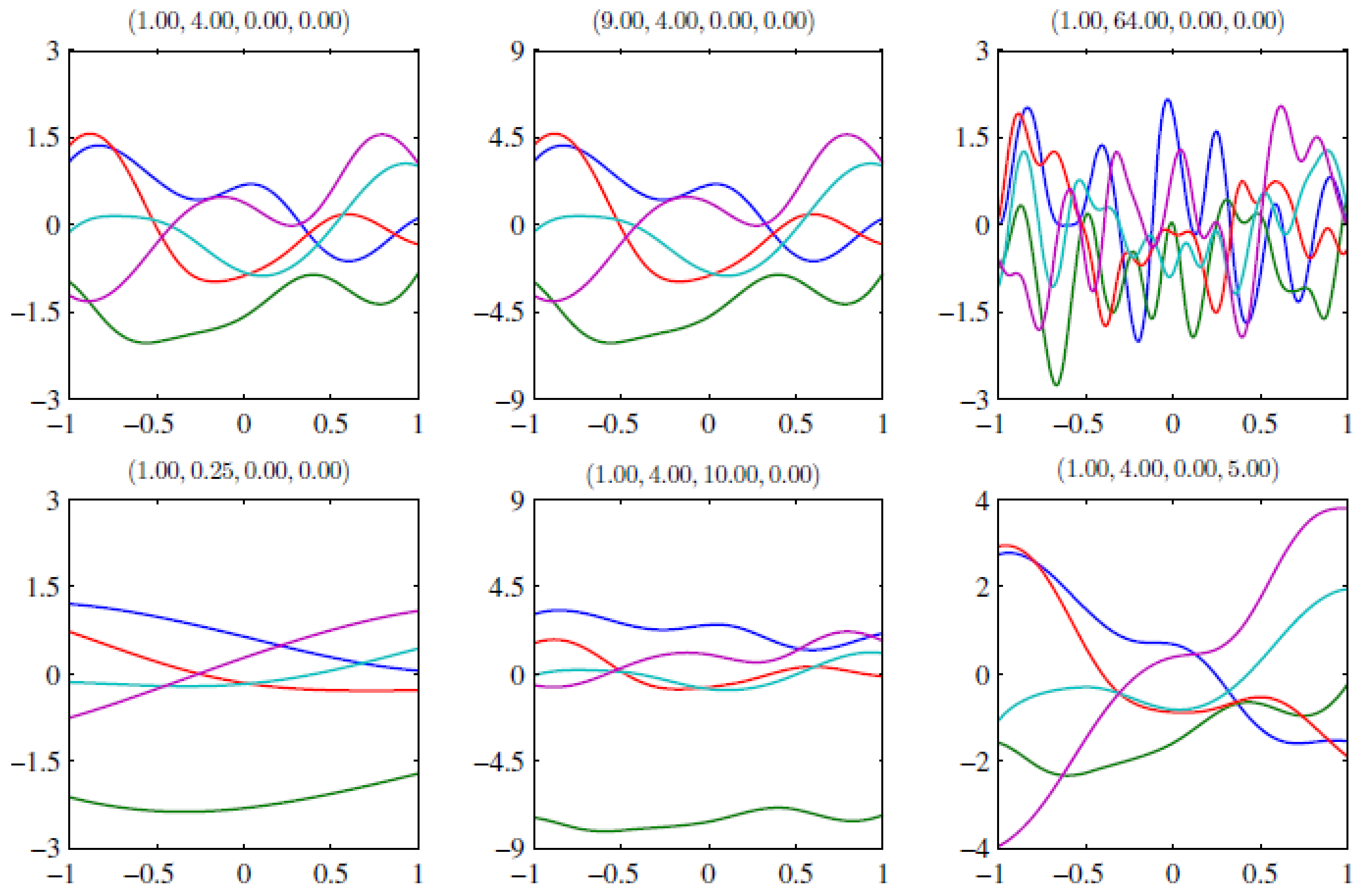
- The two Gaussian sources of randomness, namely that associated with y(x) and that associated with the noise, are independent and so their covariances simply add

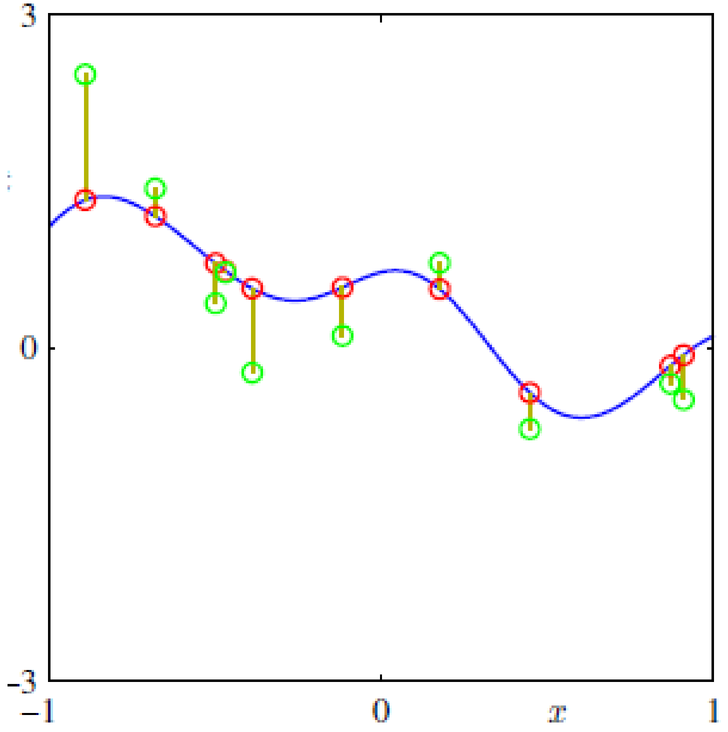
Gaussian Processes for Regression

- Kernel widely used for regression:

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

Gaussian Processes for Regression




3
0
−3
−1
0
x
1

GPR – Making Predictions

- Make prediction for a new data input, given the training data
- Goal: predict t_{N+1} given x_{N+1}
- Need to evaluate the predictive distribution:
- The distribution is also condition by $x_1, \dots, x_N, x_{N+1}$ $p(t_{N+1}|\mathbf{t}_N)$
- Consider $\mathbf{t}_{N+1} = (t1, \dots, t_N, t_{N+1})^T$

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

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

- Where: **k** is a vector of: $k(\mathbf{x}_n, \mathbf{x}_{N+1})$ for $n = 1, \dots, N$
- c is a scalar: $c = k(\mathbf{x}_{N+1}, \mathbf{x}_{N+1}) + \beta^{-1}$

GPR – Making Predictions

$p(t_{N+1}|\mathbf{t})$ **The predective distribution**

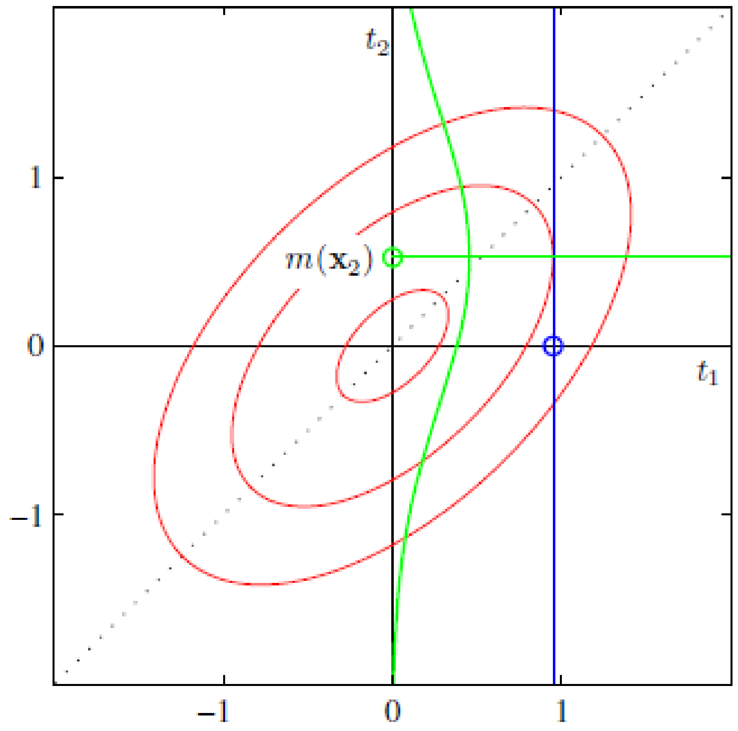
- Is a Gaussian distribution with mean and covariance:

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

- Key results for the Gaussian process regression
- The mean and variance both depend on x_{N+1}
- Matrix **C** has to be positive definite <=> the kernel function is positive semi-definite => we can use kernel functions and properties as discussed in the previous slides to construct new kernels

Example

- One training point, one test point



Better View on Gaussian Processes

- Video lecture from Cambridge:
  - http://videolectures.net/gpip06_mackay_gpb/