Kernel Methods

Henrik I Christensen

Robotics & Intelligent Machines @ GT
Georgia Institute of Technology,
Atlanta, GA 30332-0280
hic@cc.gatech.edu

Outline

Introduction

- This far the process has been about data compression and optimal discrimination
- Once process complete the training set is discarded and the model is used for processing
- What if data were kept and used directly for estimation?
- Why you ask?
- The decision boundaries might not be simple or the modeling is too complicated
- Already discussed Nearest Neighbor (NN) as an example of direct data processing
- A complete class of memory based techniques
- Q: how to measure similarity between a data point and samples in memory?

Kernel Methods

- What if we could predict based on a linear combination of features?
- Assume a mapping to a new feature space using $\phi(\mathbf{x})$
- A kernel function is defined by

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

- Characteristics:
  - The function is symmetric: $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$
  - Can be used both on continuous and symbolic data
- Simple kernel

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

  the linear kernel.
- A kernel is basically an inner product performed in a feature/mapped space.

Kernels

- Consider a complete set of data in memory
- How can we interpolate new values based on training values? I.e.,

$$y(x) = \frac{1}{\sum k} \sum_{n=1}^{N} k(x, x_n) x_n$$

- consider $k(.,.)$ a weight function that determines contribution based on distance between x and x_n

Outline

Dual Representation

- Consider a regression problem as seen earlier

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

- with the solution

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

where $\mathbf{a}$ is defined by

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

- Substitute $\mathbf{w} = \Phi^t\mathbf{a}$ into $J(\mathbf{w})$ to obtain

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

which is termed the dual representation

Dual Representation II

- Define the Gram matrix - $\mathbf{K} = \Phi\Phi^T$ to get

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

  where

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

- $J(a)$ is then minimized by

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

- Through substitution we obtain

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

- We have in reality mapped the program to another (dual) space in which it is possible to optimize the regression/discrimination problem
- Typically $N >> M$ so the immediate advantage is not obvious. See later.

Outline

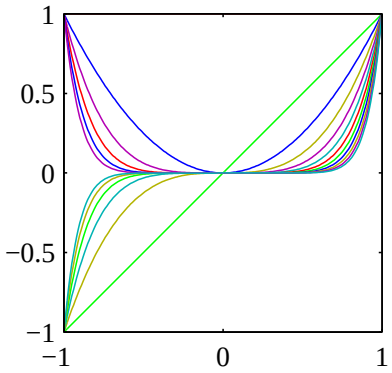
Constructing Kernels

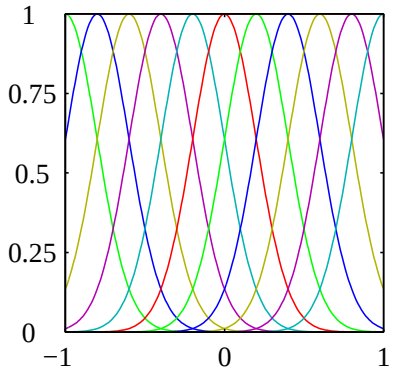
- How would we construct kernel functions?
- One approach is to choose a mapping and find corresponding kernels
- A one dimensional example

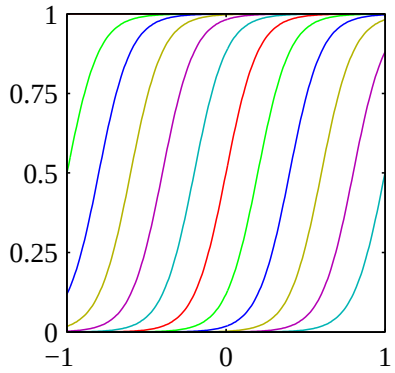
$$k(x, x') = \phi(x)^T \phi(x') = \sum_{n=1}^{M} \phi_i(x)\phi_i(x')$$

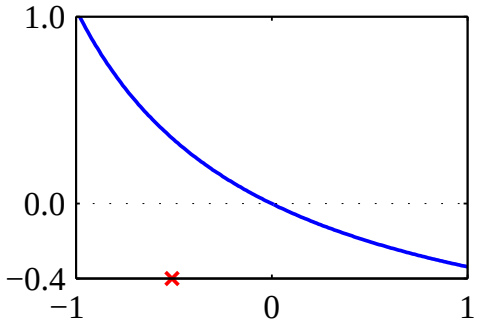
where $\phi_i(.)$ are basis functions

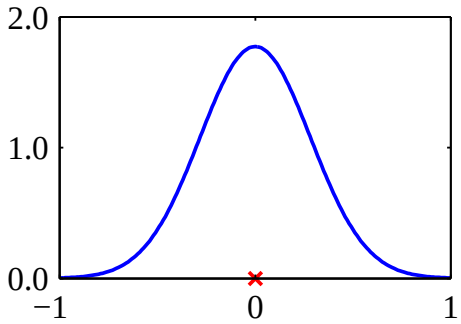
Kernel Basis Functions - Example

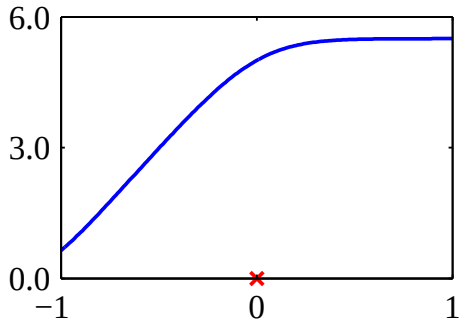












Construction of Kernels

- We can also design kernels directly.
- Must correspond to a scala product in "some" space
- Consider:

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

for a 2-dimensional space $\mathbf{x} = (x_1, x_2)$

$$\begin{aligned} k(\mathbf{x}, \mathbf{z}) &= (\mathbf{x}^T \mathbf{z})^2 = (x_1 z_1 + x_2 z_2)^2 \\ &= x_1^2 z_1^2 + 2 x_1 z_1 x_2 z_2 + x_2^2 z_z^2 \\ &= (x_1^2, \sqrt{2} x_1 x_2, x_2^2)(z_1^2, \sqrt{2} z_1 z_2, z_2^2)^T \\ &= \phi(\mathbf{x})^T \phi(\mathbf{z}) \end{aligned}$$

- In general if the Gram matrix, $\mathbf{K}$, is positive semi-definite the kernel function is valid

Techniques for construction of kernels

$$
\begin{aligned}
k(\mathbf{x},\mathbf{x}') &= c_1 k(\mathbf{x},\mathbf{x}') \\
k(\mathbf{x},\mathbf{x}') &= f(\mathbf{x}) k(\mathbf{x},\mathbf{x}') f(\mathbf{x}') \\
k(\mathbf{x},\mathbf{x}') &= q(k(\mathbf{x},\mathbf{x}')) \\
k(\mathbf{x},\mathbf{x}') &= exp(k(\mathbf{x},\mathbf{x}')) \\
k(\mathbf{x},\mathbf{x}') &= k_1(\mathbf{x},\mathbf{x}') + k_2(\mathbf{x},\mathbf{x}') \\
k(\mathbf{x},\mathbf{x}') &= k_1(\mathbf{x},\mathbf{x}') k_2(\mathbf{x},\mathbf{x}') \\
k(\mathbf{x},\mathbf{x}') &= \mathbf{x}^T \mathbf{A} \mathbf{x}'
\end{aligned}
$$

More kernel examples/generalizations

- We could generalize $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T\mathbf{x}')^2$ in various ways
  1. $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T\mathbf{x}' + c)^2$
  2. $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T\mathbf{x}')^M$
  3. $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T\mathbf{x}' + c)^M$
- Example correlation between image regions
- Another option is

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

  called the "Gaussian kernel" (see later)
- Several more examples in the book

Outline

Radial Basis Functions

- What is a radial basis function?

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

- How to average/smooth across data entirely based on distance?

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

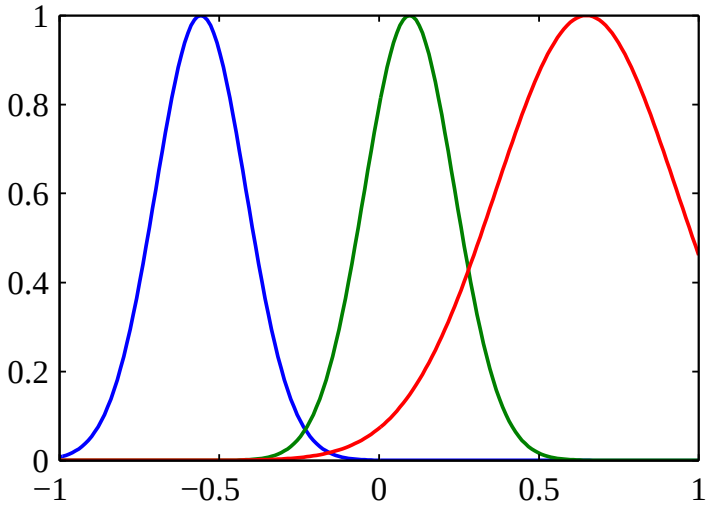
  the weights w_n could be estimated using LSQ
- A popular interpolation strategy is:

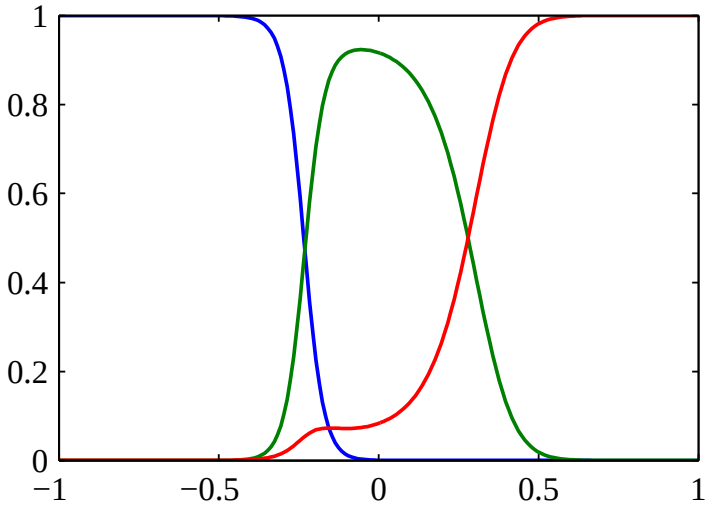
$$y(\mathbf{x}) = \sum_{n=1}^{N} t_n h(\mathbf{x} - \mathbf{x}_n)$$

  where

$$h(\mathbf{x} - \mathbf{x}_n) = \frac{\nu(\mathbf{x} - \mathbf{x}_n)}{\sum_j \nu(\mathbf{x} - \mathbf{x}_j)}$$

The effect of normalization?





Nadaraya-Watson Models

- Lets interpolate across all data!
- Using a Parzen density estimator we have

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

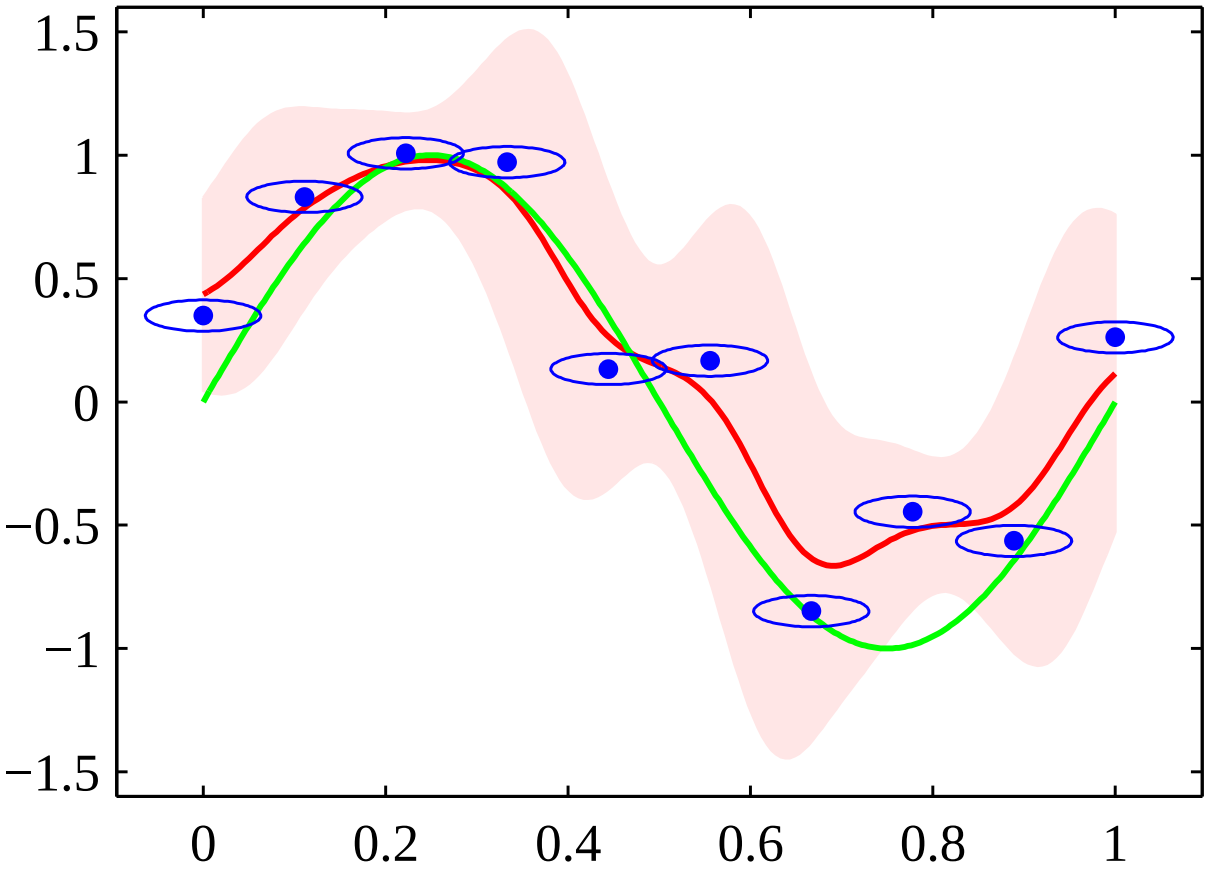
- We can then estimate

$$\begin{aligned} y(\mathbf{x}) &= E[t|\mathbf{x}] = \int_{-\infty}^{\infty} t p(t|\mathbf{x}) dt \\ &= \frac{\int t p(\mathbf{x}, t) dt}{\int p(\mathbf{x}, t) dt} \\ &= \frac{\sum_n g(\mathbf{x} - \mathbf{x}_n) t_n}{\sum_m g(\mathbf{x} - \mathbf{x}_m)} \\ &= \sum_n k(\mathbf{x}, \mathbf{x}_n) t_n \end{aligned}$$

Gaussian Mixture Example

- Assume a particular one-dimensional function (here sine) with noise
- Each data point is an iso-tropic Gaussian Kernel
- Smoothing factors are determined for the interpolation

Gaussian Mixture Example



Gaussian Kernels

- We have so far considered basic of kernels - a distance metric
- Transformations to a new space
- Lets consider Gaussian Processes
  - Rather than direct regression / classification
  - What if the mapping is probabilistic over a function space
  - Ex; training with noisy training data

Outline

Linear Regression Revisited

- In Regression we are used to

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

- what is the weights were probabilistic

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

- we can reformulate the optimization to be

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

Gaussian Models

- Considering basic Gaussian parameters

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

- where K is the Gram matrix which defines the kernel, i.e.

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

Gaussian Processes

- Determined in full by 1. and 2. order moments
- Typically no knowledge of mean so

$$E[p(w|t)] = 0$$

  is assumed a good guess
- Specification of the co-variance is thus adequate

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

- One could also define the Gaussian kernels directly as a set of basis functions.

Outline

Gaussian Processes for Regression

- If we have noisy training data

$$t_n = y_n + \epsilon_n$$

- then we can model the data as

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

- for a vector of data (your training set) we have

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

- the marginalized distribution is then

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

where K is the Gram matrix

Gaussian Processes for Regression II

- We can then compute the marginal for the target

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

where

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

- We can thus express the distribution of $\mathbf{y}$ entirely based on the kernel function

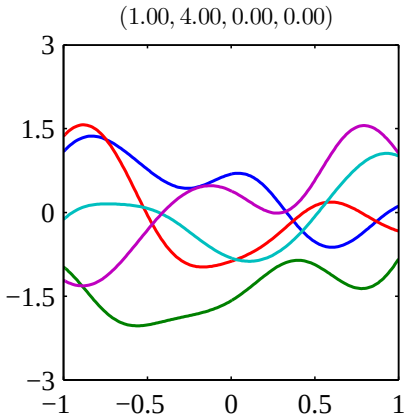
Exponential quadratic Gaussian Processes

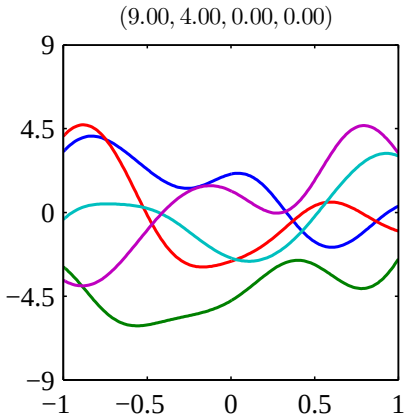
- A popular family of Gaussian processes are defined by

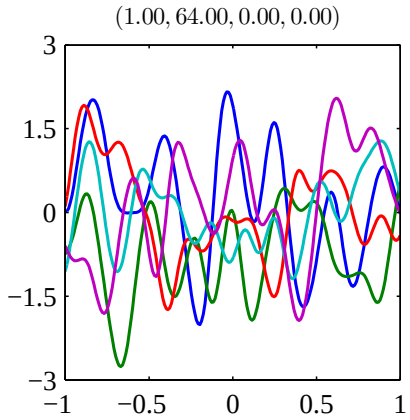
$$k(\mathbf{x}_n, \mathbf{x}_m) = \theta_0 exp\left\{-\frac{\theta_1}{2}||\mathbf{x}_n - \mathbf{x}_m||^2\right\} + \theta_2 + \theta_3 \mathbf{x}_n^T \mathbf{x}_m$$

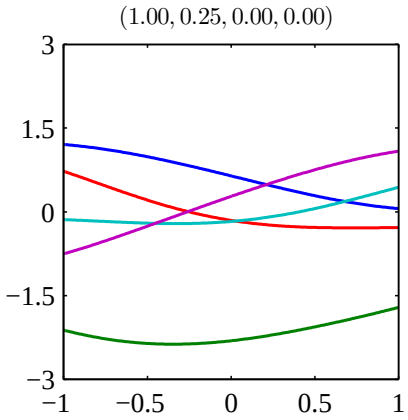
- Which has a bias term, a linear term and the quadratic exponential
- Allows representation of a broad family of functions

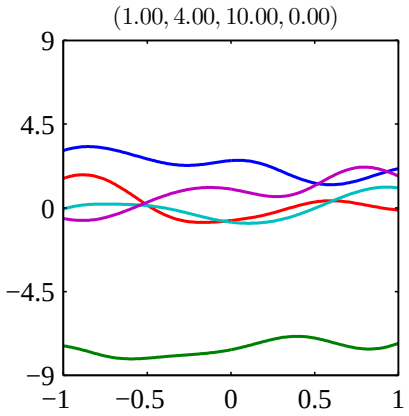
Quadratic Exponential Gaussian Processes

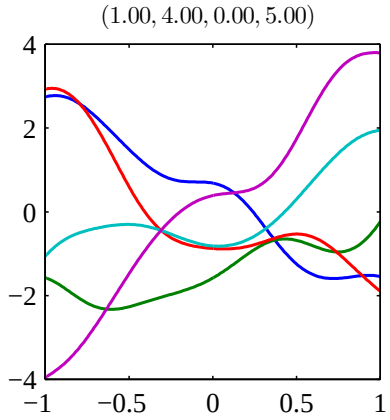












Recursive Process Regression

- In temporal processes we would like to model

$$p(t_{N+1}|\mathbf{t}_N, \mathbf{x}_{N+1})$$

- For the process we have

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

- We can partition $\mathbf{C}$

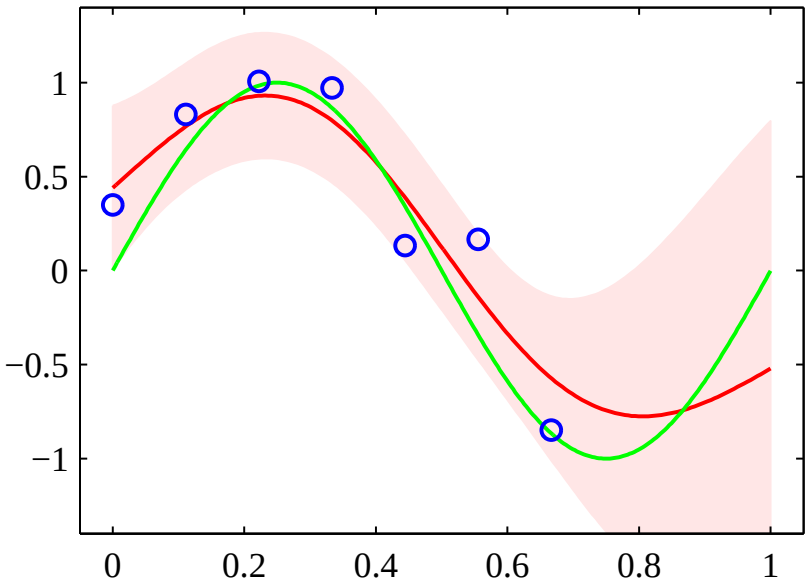
$$\mathbf{C}_{N+1} = \begin{bmatrix} \mathbf{C}_N & \mathbf{k} \\ \mathbf{k}^T & c \end{bmatrix}$$

where k is composed of $k(\mathbf{x}_i, \mathbf{x}_{N+1})$ and $c = k(\mathbf{x}_{N+1}, \mathbf{x}_{N+1}) + \beta^{-1}$

Recursive Process Regression II

- The mean and variance is then

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



Outline

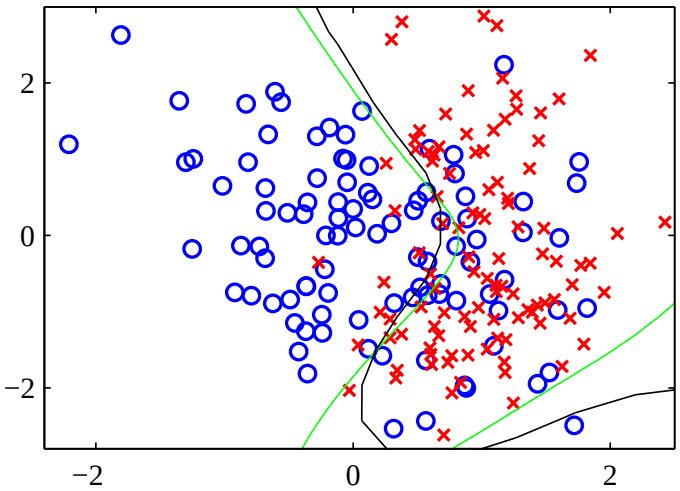
Gaussian Processes for Classification

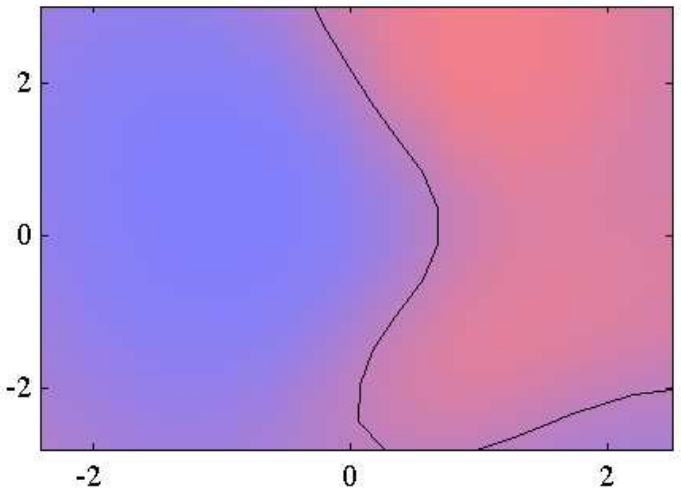
- This far we have considered regression over the full space
- For classification the optimization would be with respect to miss classification

$$p(t|a) = \sigma(a)^t(1 - sigma(a))^{1-t}$$

- Very similar derivations can be performed as detailed in the book.

Gaussian Process Classification Example





Outline

Summary

- Memory based methods - keeping the data!
- Design of distance metrics for weighting of data in learning set
- Kernels - a distance metric based on dot-product in some feature space
- Being creative about design of kernels
- Gaussian processes represent a broad class of stochastic processes
- Estimation of Gaussian Processes is a way to optimize fit to data and to obtain estimate of uncertainty as interpolation is performed away from learning data
- A good source:
  - C. E. Rasmussen & C. K. I. Williams, "Gaussian Processes for Machine Learning", the MIT Press, 2006
  - Available from `http://www.GaussianProcess.org/gpml`
  - Includes a good Matlab toolkit