KERNEL METHODS

J. Elder

CSE 6390/PSYC 6225 Computational Modeling of Visual Perception

Outline

- Dual representation
- Constructing kernels
- Radial basis functions
- Gaussian processes

Parametric vs Kernel Methods

- In linear **parametric** regression or classification we model target variables t as functions $y(\mathbf{x},\mathbf{w})$ of the input vector $\mathbf{x}$, where $\mathbf{w}$ is a parameter vector.
  1. Estimate $\mathbf{w}$ from the training input vectors $\mathbf{x}_n$ and their associated target variables $\mathbf{t}_n$
  2. Throw away the training data $(\mathbf{x}_n, \mathbf{t}_n)$
  3. Given a new input vector $\mathbf{x}$, estimate the corresponding target variable $\mathbf{t}$, using the learned parameters $\mathbf{w}$:

     $$t \simeq y(x,\mathbf{w})$$

Parametric vs Kernel Methods

- In **kernel** methods we model new data ($\boldsymbol{x}$, $\boldsymbol{t}$) directly as a function of the training data ($\boldsymbol{x}_n$, $\boldsymbol{t}_n$).
  - This means we never throw away the training data.
  - The kernel method allows a high-dimensional (even infinite dimensional) feature space to be used implicitly.

The kernel function

The kernel function $k(\mathbf{x},\mathbf{x}')$ measures the 'similarity' of input vectors $\mathbf{x}$ and $\mathbf{x}'$ as an inner product in a feature space defined by the feature space mapping $\phi(\mathbf{x})$:

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

If $k(\mathbf{x},\mathbf{x}') = k(\mathbf{x}-\mathbf{x}')$ we say that the kernel is *stationary*

If $k(\mathbf{x},\mathbf{x}') = k\left(\left\|\mathbf{x}-\mathbf{x}'\right\|\right)$ we call it a *radial basis function*.

Dual Representations

- Many linear models for regression and classification can be reformulated as dual kernel models.
- For example, regularized linear least-squares regression, where the error function is given by:

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

- which has a minimum at

$$\mathbf{w} = -\frac{1}{\lambda}\sum_{n=1}^{N}\left\{\mathbf{w}^t\phi(\mathbf{x}_n) - \mathbf{t}_n\right\}\phi(\mathbf{x}_n)$$

Dual Representations

$$\mathbf{w} = -\frac{1}{\lambda}\sum_{n=1}^{N}\left\{\mathbf{w}^t\phi\left(\mathbf{x}_n\right)-\mathbf{t}_n\right\}\phi\left(\mathbf{x}_n\right)$$

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

we can reexpress $\mathbf{w}$ as

$$\mathbf{w} = \Phi^t\mathbf{a}$$

where $\phi\left(\mathbf{x}_n\right)^t$ is the n^{th} row of Φ

and a_n is the n^{th} element of $\mathbf{a}$

Dual Representations

Substituting, we can express the error function in terms of **a**:

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

which has a minimum at

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

where $\mathbf{K}$ is the Gram matrix $\mathbf{K} = \Phi\Phi^t$ with elements

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

Dual Representations

Predictions are then given by

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

where $\mathbf{k}(\mathbf{x})$ is the vector with elements $k_n(\mathbf{x}) = k(\mathbf{x}_n, \mathbf{x})$

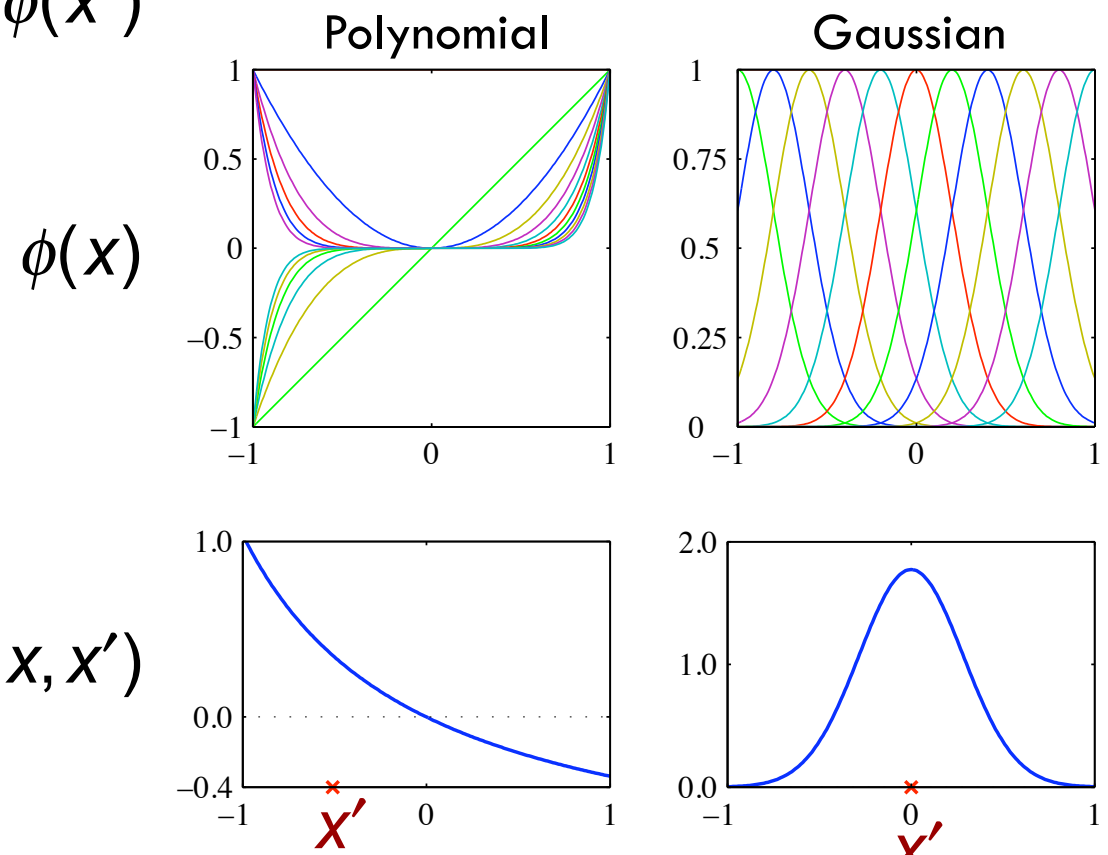
and $k\left(\mathbf{x}_n, \mathbf{x}_m\right) = \phi\left(\mathbf{x}_n\right)^t \phi\left(\mathbf{x}_m\right)$

- Thus predictions are expressed entirely in terms of kernel functions on input vector pairs.

Constructing Kernels

- One can construct a kernel by selecting a feature space mapping $\phi(x)$ and then defining

$$k(x, x') = \phi(x)^t \phi(x')$$

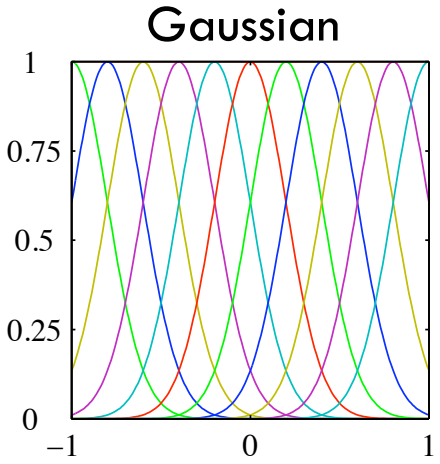


Constructing Kernels

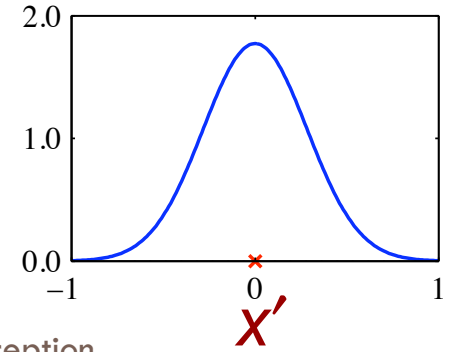
- We can construct a kernel by selecting a feature space mapping $\phi(x)$ and then defining

$$k(x, x') = \phi(x)^t \phi(x')$$

$\phi(x')$



$k(x, x')$



Constructing Kernels

- Alternatively, we can construct the kernel function directly, ensuring that it corresponds to an inner product in some (possibly infinite-dimensional) feature space.

Constructing Kernels

$$k(x) = \phi(x)^t \phi(x')$$

Example 1: $k(x,z) = x^t z$

Example 2: $k(x,z) = x^t z + c, \; c > 0$

Example 3: $k(x,z) = \left(x^t z\right)^2$

Constructing Kernels

More generally, $k(\mathbf{x}, \mathbf{x}')$ is a valid kernel if the Gram matrix $\mathbf{K}$ whose elements are given by $k(\mathbf{x}_n, \mathbf{x}_m)$ is positive semidefinite for all possible choices of $\{\mathbf{x}_n\}$.

Kernel Properties

- Kernels obey certain properties that make it easy to construct complex kernels from simpler ones.

Kernel Properties

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

$$
\begin{aligned}
k(\boldsymbol{x}, \boldsymbol{x}') &= c k_1(\boldsymbol{x}, \boldsymbol{x}') && (6.13)\\
k(\boldsymbol{x}, \boldsymbol{x}') &= f(\boldsymbol{x}) k_1(\boldsymbol{x}, \boldsymbol{x}') f(\boldsymbol{x}') && (6.14)\\
k(\boldsymbol{x}, \boldsymbol{x}') &= q(k_1(\boldsymbol{x}, \boldsymbol{x}')) && (6.15)\\
k(\boldsymbol{x}, \boldsymbol{x}') &= exp(k_1(\boldsymbol{x}, \boldsymbol{x}')) && (6.16)\\
k(\boldsymbol{x}, \boldsymbol{x}') &= k_1(\boldsymbol{x}, \boldsymbol{x}') + k_2(\boldsymbol{x}, \boldsymbol{x}') && (6.17)\\
k(\boldsymbol{x}, \boldsymbol{x}') &= k_1(\boldsymbol{x}, \boldsymbol{x}') k_2(\boldsymbol{x}, \boldsymbol{x}') && (6.18)\\
k(\boldsymbol{x}, \boldsymbol{x}') &= k_3(\boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{x}')) && (6.19)\\
k(\boldsymbol{x}, \boldsymbol{x}') &= \boldsymbol{x}^T \mathbf{A} \boldsymbol{x}' && (6.20)\\
k(\boldsymbol{x}, \boldsymbol{x}') &= k_a(\boldsymbol{x}_a, \boldsymbol{x}'_a) + k_b(\boldsymbol{x}_b, \boldsymbol{x}'_b) && (6.21)\\
k(\boldsymbol{x}, \boldsymbol{x}') &= k_a(\boldsymbol{x}_a, \boldsymbol{x}'_a) k_b(\boldsymbol{x}_b, \boldsymbol{x}'_b) && (6.22)
\end{aligned}
$$

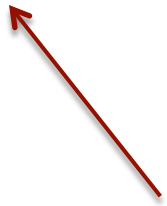
where $c > 0$, $f(\cdot)$ is any function, $q(\cdot)$ is a polynomial with nonnegative coefficients, $\phi(\mathbf{x})$ is a mapping from $\mathbf{x} \rightarrow \mathbb{R}^M$, A is a symmetric positive semidefinite matrix, $\mathbf{x}_a$ and $\mathbf{x}_b$ are variables such that $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$ and k_a, k_b are valid kernels over their respective spaces.

Constructing Kernels

- Examples:

$$k(x^t, x') = \left(x^t x' + c\right)^M, c > 0 \qquad \text{(Use 6.18)}$$

$$k(x, x') = \exp\left(-\left\|x - x'\right\|^2 / 2\sigma^2\right) \qquad \text{(Use 6.14 and 6.16.)}$$



Corresponds to infinite-dimensional feature vector

Radial Basis Functions

- Widely used choice of basis function for linear regression.

For example, given training input vectors $\{x_1,\ldots,x_N\}$ and corresponding target values $\{t_1,\ldots t_N\}$, we can define an interpolation function $f(x)$ as:

$$f(x)=\sum_{n=1}^{N} w_n h\left(\|x-x_n\|\right)$$

Since **w** and **t** have the same dimensionality, we can fit the training data exactly, while providing predictions for input vectors we haven't yet seen.

END OF LECTURE
NOV 3, 2010

J. Elder

CSE 6390/PSYC 6225 Computational Modeling of Visual Perception

Kernel Regression

- Can also be used for prediction when there is noise in **x**, as well as **t** (the **Nadaraya-Watson** model):

Let ξ be the noise in **x**, drawn from distribution $\nu(\xi)$. Then

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

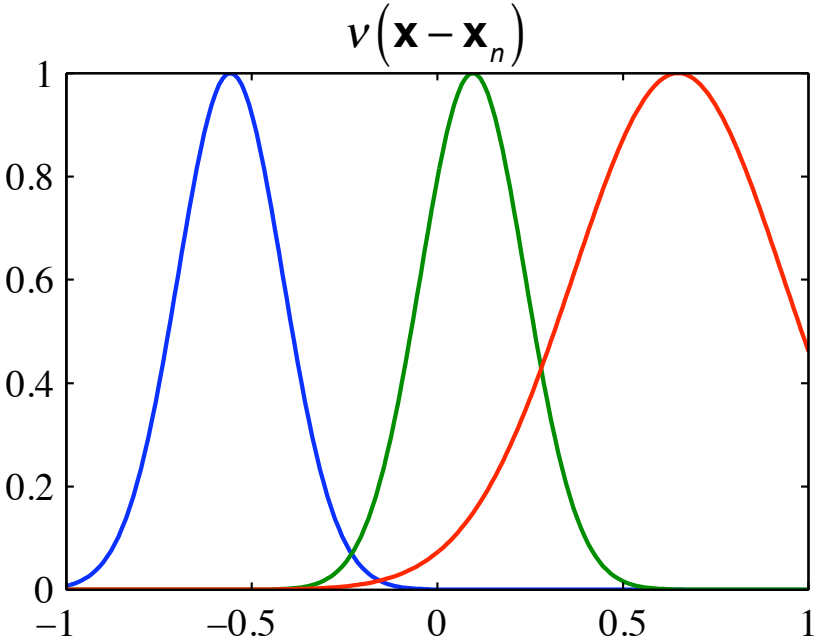
Using the calculus of variations (Appendix D),
we can find the optimal interpolation function $y(\mathbf{x})$:

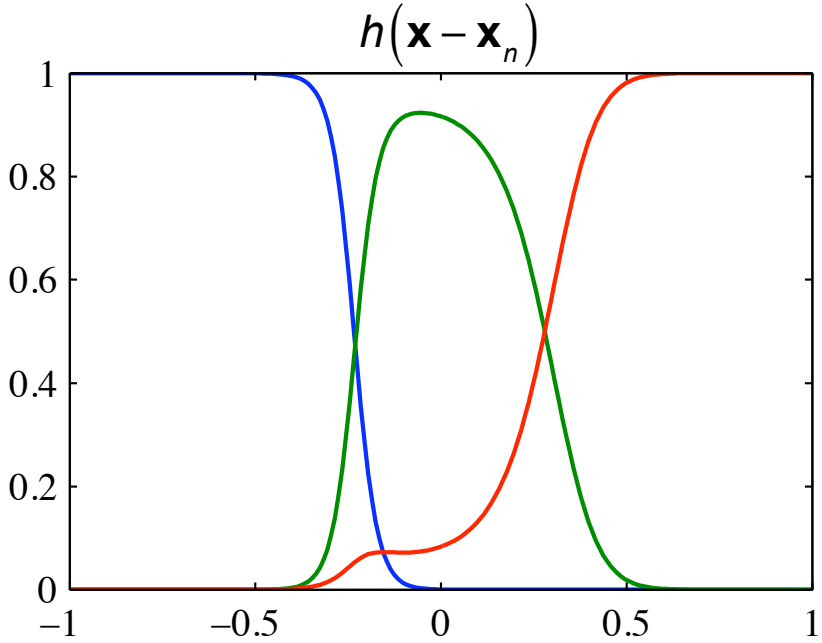
$$y(\mathbf{x}) = \sum_{n=1}^{N} t_n h\left(\mathbf{x} - \mathbf{x}_n\right)$$

where

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

Kernel Regression





$$y(\mathbf{x}) = \sum_{n=1}^{N} t_n h(\mathbf{x}-\mathbf{x}_n), \quad \text{where} \quad h(\mathbf{x}-\mathbf{x}_n) = \frac{v(\mathbf{x}-\mathbf{x}_n)}{\sum_{n=1}^{N} v(\mathbf{x}-\mathbf{x}_n)}$$

Nadaraya-Watson Model

- The Nadaraya-Watson model can also be understood as a form of kernel density estimation:

Given a training set $\{\mathbf{x}_n, t_n\}$, we model the joint distribution $p(\mathbf{x}, t)$ using a Parzen density estimator

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

Then the regression fuction $y(\mathbf{x})$ is given by

$$y(\mathbf{x}) = E[t \mid \mathbf{x}] = \sum_n k(\mathbf{x}, \mathbf{x}_n) t_n$$

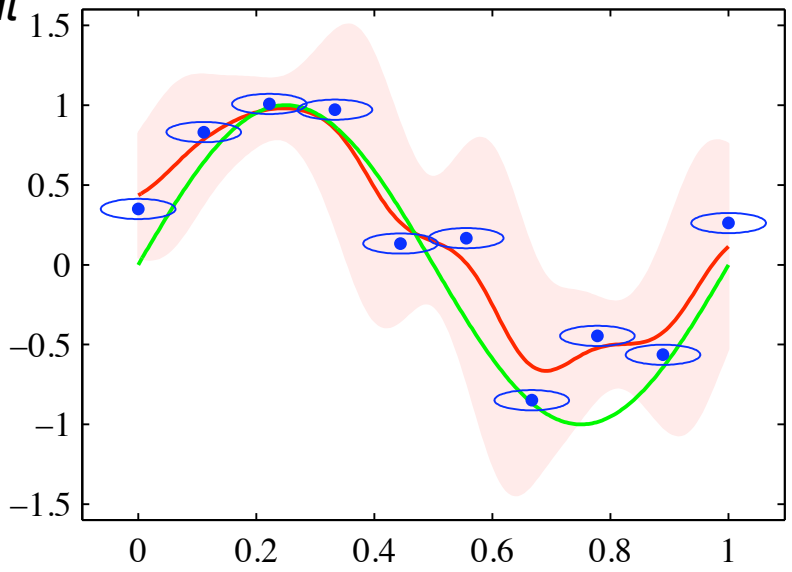
where $k(\mathbf{x}, \mathbf{x}_n) = \frac{g(\mathbf{x} - \mathbf{x}_n)}{\sum_m g(\mathbf{x} - \mathbf{x}_m)}$ and $g(\mathbf{x}) = \int_{-\infty}^{\infty} f(\mathbf{x}, t)\, dt$

Nadaraya-Watson Model

- The model provides not only the expected target value, but the entire posterior:

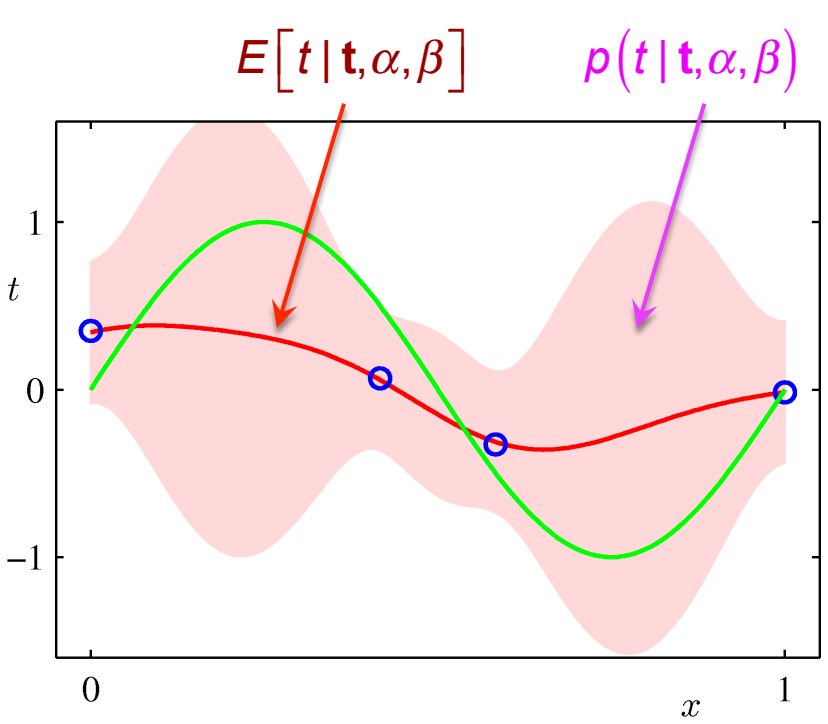
Given a training set $\{\mathbf{x}_n, t_n\}$, we model the joint distribution $p(\mathbf{x}, t)$ using a Parzen density estimator

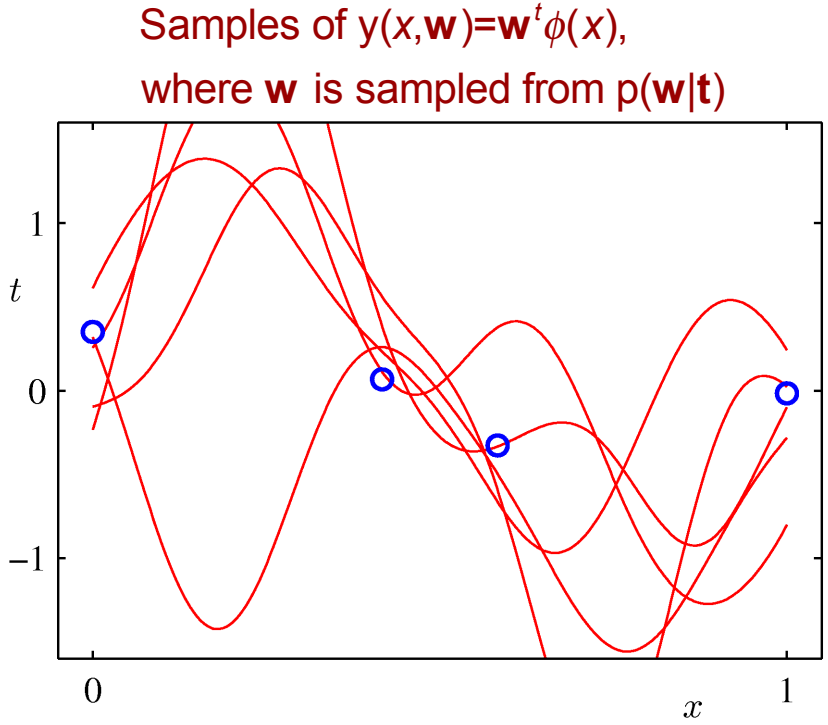
$$p(t \mid \mathbf{x}) = \frac{p(t,\mathbf{x})}{\int p(t,\mathbf{x})\,dt} = \frac{\sum_n f(\mathbf{x}-\mathbf{x}_n, t-t_n)}{\sum_m \int f(\mathbf{x}-\mathbf{x}_m, t-t_m)\,dt}$$



Gaussian Processes

- In linear regression, we estimate a posterior distribution over parameters **w**.
- This determines a prior distribution p(y) over the function y.





Gaussian Processes

- In the Gaussian process approach, we dispense with the parameters **w** and define the prior over y directly.

More precisely, a Gaussian process defines a probability distribution over functions $y(\mathbf{x})$ such that the set of values of $y(\mathbf{x})$ evaluated at an arbitrary set of points $\mathbf{x}_1,\ldots\mathbf{x}_N$ are jointly Gaussian.

Equivalence with Classical Regression

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

Assume a Gaussian prior over the weight vector: $p(\mathbf{w}) = N\left(\mathbf{w} \mid \mathbf{0}, \alpha^{-1}\mathbf{I}\right)$.

We have training data $\mathbf{x}_1, \ldots \mathbf{x}_N$

For given $\mathbf{w}$, this determines a vector of function values $\mathbf{y} = \left(y\left(\mathbf{x}_1\right), \ldots y\left(\mathbf{x}_N\right)\right)^t$:

$$\mathbf{y} = \Phi\mathbf{w}, \quad \text{where } \Phi_{nk} = \phi_k\left(\mathbf{x}_n\right)$$

Since y is a linear function of $\mathbf{w}$, it is also Gaussian, with:

$$E\left[y\right] = 0$$

$$\text{cov}\left[y\right] = \frac{1}{\alpha}\Phi\Phi^t = \mathbf{K},$$

$$\text{where } K_{nm} = k\left(\mathbf{x}_n, \mathbf{x}_m\right) = \frac{1}{\alpha}\phi\left(\mathbf{x}_n\right)\phi\left(\mathbf{x}_m\right)$$

Gaussian Processes

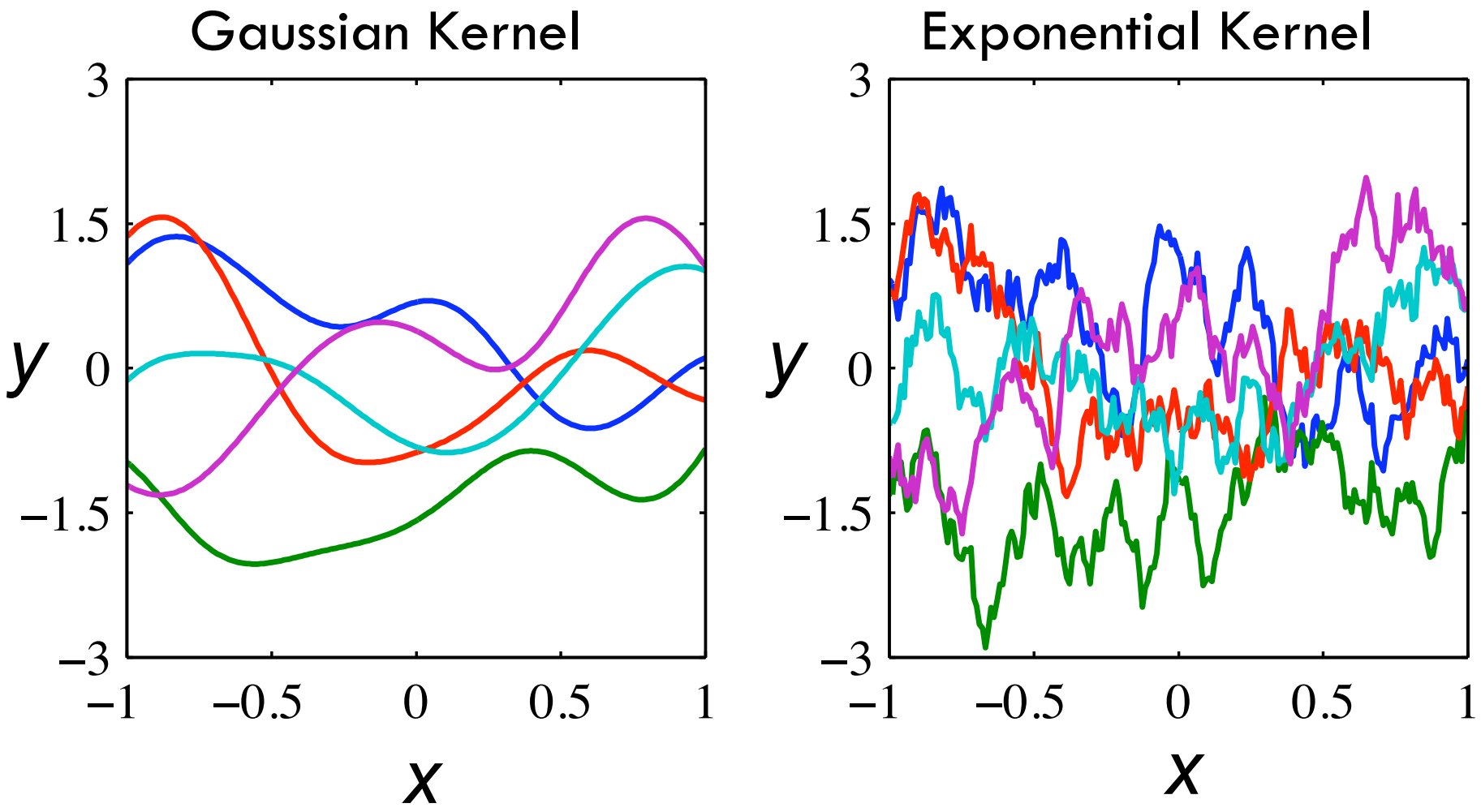
- Thus we have a direct probabilistic model for $y(\mathbf{x}_n)$:

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

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

- Notes
  - All we need to specify this model is the kernel function evaluated over all pairs of training input vectors.
  - The kernel function can be defined in terms of feature vectors, as above, or directly.
  - Normally, the kernel function is defined so that $y(\mathbf{x}_n)$ and $y(\mathbf{x}_m)$ are strongly correlated if $\mathbf{x}_n$ and $\mathbf{x}_m$ are close to each other.

Gaussian Processes: Examples



Using Gaussian Processes for Prediction

- To use Gaussian processes for prediction, we need to account for the noise in the observed target values:

$$t_n = y_n + \varepsilon_n$$

where ε_n is Gaussian iid, so that:

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

Using Gaussian Processes for Prediction

Knowing $p(\mathbf{y})$ and $p(\mathbf{t}|\mathbf{y})$, we can use the results from Chapter 2 to determine $p(\mathbf{t})$:

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

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

Predictions can now be made using the partitioned Gaussian equations from Chapter 2.3.2:

$$p(t_{N+1}|\mathbf{t}) = N(t_{N+1}|m(\mathbf{x}_{N+1}),\sigma^2(\mathbf{x}_{N+1}))$$

where

$$m(\mathbf{x}_{N+1}) = \mathbf{k}^t\mathbf{C}_N^{-1}\mathbf{t} \qquad k_n = k(\mathbf{x}_n,\mathbf{x}_{N+1})$$

$$\sigma^2(\mathbf{x}_{N+1}) = c - \mathbf{k}^t\mathbf{C}_N^{-1}\mathbf{k} \qquad c = k(\mathbf{x}_{N+1},\mathbf{x}_{N+1}) + \beta^{-1}$$

Learning the Hyperparameters

We have assumed that the kernel functions $k(\mathbf{x}_n, \mathbf{x}_m)$ are fixed.

This corresponds to a fixed feature space mapping $\phi(\mathbf{x})$.

In practice, we may instead wish to define the covariance function through a parameterized family of kernels, and then infer the parameters θ from the data.

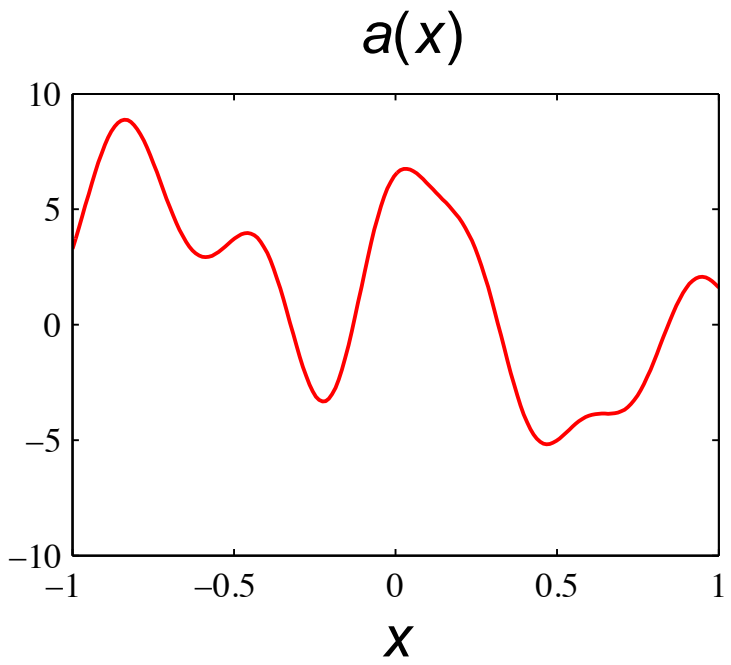
In general, this will require iterative optimization and approximation methods.

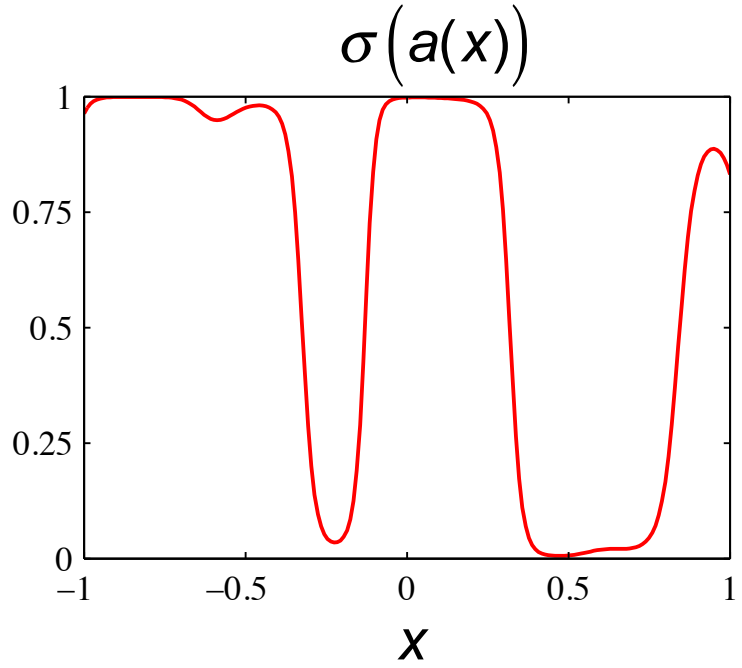
Gaussian Processes for Classification

For Gaussian process regression, y is Gaussian, and assumes values on $\mathbb{R}$.

For binary classification, where $t \in \{0,1\}$, we let a(**x**) be a Gaussian process,

and define $y(\mathbf{x}) = \sigma(a)$, where $\sigma(a)$ is the sigmoid function $\dfrac{1}{1+\exp(-a)}$





Gaussian Processes for Classification

Now $p(t \mid a) = \sigma(a)^t \left(1 - \sigma(a)\right)^{1-t}$

Given training data $\mathbf{x}_1, \ldots \mathbf{x}_N$, define

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

$\mathbf{a}_{N+1} = \left(a(\mathbf{x}_1), \ldots, a(\mathbf{x}_{N+1})\right)^t$

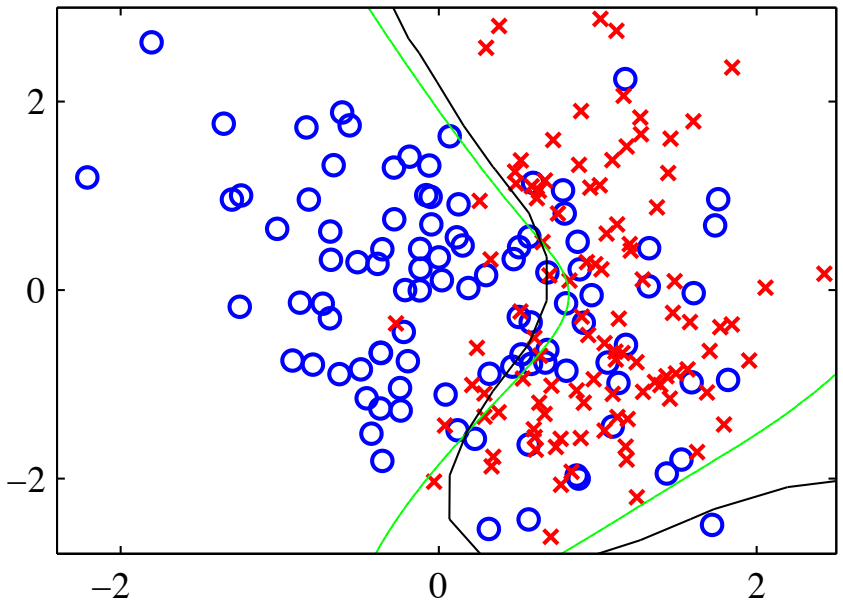
where $p(\mathbf{a}_{N+1}) = N(\mathbf{a}_{N+1} \mid \mathbf{0}, \mathbf{C}_{N+1})$

and $\mathbf{C}_{nm} = k(\mathbf{x}_n, \mathbf{x}_m) + \nu\delta_{nm}$

Then $p(t_{N+1} = 1 \mid \mathbf{t}_N) = \int p(t_{N+1} = 1 \mid a_{N+1}) p(a_{N+1} \mid \mathbf{t}_N) da_{N+1} = \int \sigma(a_{N+1}) p(a_{N+1} \mid \mathbf{t}_N) da_{N+1}$

This integrable is intractable - must approximate. See text for details.

Example



$p\left(t_{N+1}=1\,|\,\mathbf{t}_N\right)$