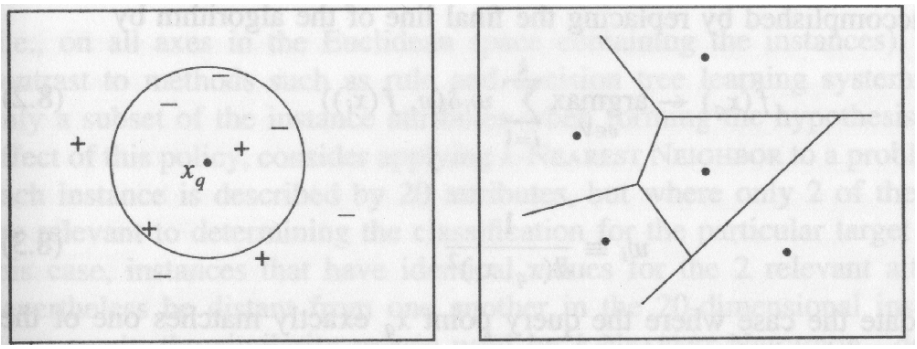
7.4 *k*-Nearest Neighbor Methods

[Mitchell, 1997]

k-Nearest Neighbor Rule



$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} (x_i - x_j)^2}$$

$$f : \Re^n \rightarrow V$$

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k}$$

- Training algorithm:
  - For each training example $(x, f(x))$, add the example to the list *training_examples*
- Classification algorithm:
  - Given a query instance x_q to be classified,
    - Let $x_1, \ldots, x_k$ denote the k instances from *training_examples* that are nearest to x_q
    - Return $\hat{f}(x_q) \leftarrow \arg\max_{v \in V} \sum_{i=1}^{k} \delta(v, f(x_i))$

      where $\delta(a, b) = 1$ if $a = b$ and $\delta(a, b) = 0$ otherwise

Distance-Weighted *k*-NN Algorithm

- Giving greater weight to closer neighbors
  - Discrete case

$$\hat{f}(x_q) \leftarrow \arg\max_{v \in V} \sum_{i=1}^{k} w_i \delta(v, f(x_i)) \qquad w_i = \frac{1}{d(x_q, x_i)^2}$$

  - Real case

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}$$

Remarks on *k*-NN Algorithm

- Robust to noisy training data
- Effective when sufficiently large set of training data
- The distance between instances is calculated based on *all* attributes of the instance.
  - Cf.: Decision trees (*subset* of instance attributes)
  - Cf.: Hyperedges are subsets of the attributes as well.
- **Curse of dimensionality**: When many irrelevant attributes are present, the distance between neighbors will be dominated by irrelevant attributes.
  - Solution: weight each attribute differently
- Indexing the stored training examples
  - *kd*-tree

The inductive bias of *k*-NN corresponds to an assumption that the classification of an instance will be most similar to the classification of other instances that are nearby in Euclidean distance.

k-NN의 계산복잡도를 줄이는 방법 3가지

1. Computing partial distance
2. Prestructuring
3. Editing the stored prototypes

부분거리를 사용한 *k*-NN

k-NN의 계산 복잡도를 줄이기 위해서 전체 d 차원의 부분집합인 r 차원만을 사용하여 거리를 계산할 수 있다. r 개의 선별된 차원(즉 $r < d$)에 기반한 부분 거리는 다음과 같이 정의된다.

$$D_r(\mathbf{a},\mathbf{b}) = \left(\sum_{k=1}^{r} (a_k - b_k)^2 \right)^{1/2}$$

여기서 유클리드 거리를 가정하였다. 보다 일반적인 거리 측도 중의 하나는 민코프스키 측도로서

$$L_k(\mathbf{a},\mathbf{b}) = \left(\sum_{i=1}^{d} | a_i - b_i |^k \right)^{1/k}$$

이는 L_k 놈(norm)이라고도 한다. 따라서 유클리드 거리는 L_2 놈이다. L_1 놈은 맨하탄 거리 또는 도시 블록 거리하고도 한다.

Note: A hyperedge E_i with cardinality *k* of product unit defines the *k* subdimensions and computes the similarity (or indicator function) $d_i^{(k)}(\mathbf{a},\mathbf{b})$ of two vectors **a** and **b**:

$$D_k(\mathbf{a},\mathbf{b}) = d_i^{(k)}(\mathbf{a},\mathbf{b}) = \begin{cases} 1, & \text{if } \sum_{j \in E_i} | a_j - b_j | = 0 \quad \text{(same)} \\ 0, & \text{if } \sum_{j \in E_i} | a_j - b_j | \neq 0 \quad \text{(different)} \end{cases}$$

7.5 Radial Basis Function Networks

[Mitchell, 1997; MacKay, Chap. 45]

Locally Weighted Linear Regression

- Local
  - The function is approximated based only on data near the query point.
- Weighted
  - The contribution of each training example is weighted by its distance from the query point.
- Regression
  - Approximating real-valued function

Global Approximation

$$\hat{f}(x) = w_0 + w_1 x_1 + \cdots + w_n x_n$$

x_i: ith attribute of the instance x

- Global approximation: minimize the squared error sum over all training examples (gradient descent)

$$E \equiv \frac{1}{2} \sum_{x \in D} \left(f(x) - \hat{f}(x) \right)^2$$

$$\Delta w_j = \eta \sum_{x \in D} \left(f(x) - \hat{f}(x) \right) x_j \qquad \text{: delta rule}$$

$f(x)$: target output

$\hat{f}(x)$: actual output

Local Approximation: Three Criteria

1. Minimize the squared error over k nearest neighbors

$$E_1(x_q) \equiv \frac{1}{2} \sum_{x \in \{k \text{ nearest nbrs of } x_q\}} \left(f(x) - \hat{f}(x) \right)^2$$

2. Minimize the squared error over entire set D, with weights

$$E_2(x_q) \equiv \frac{1}{2} \sum_{x \in D} \left(f(x) - \hat{f}(x) \right)^2 K(d(x_q, x))$$

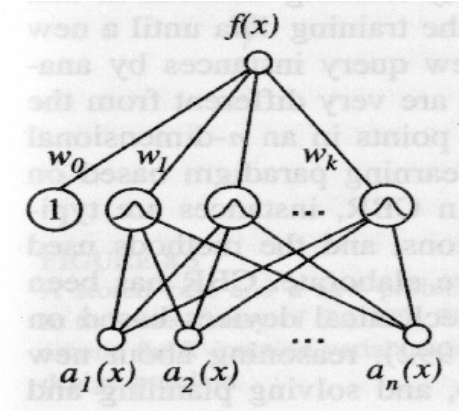
e.g.: $K(d(x_q, x)) = \exp(-d(x_q, x))$ kernel function

3. Combine 1 and 2

$$E_3(x_q) \equiv \frac{1}{2} \sum_{x \in \{k \text{ nearest nbrs of } x_q\}} \left(f(x) - \hat{f}(x) \right)^2 K(d(x_q, x))$$

- with training rule

$$\Delta w_j = \eta \sum_{x \in \{k \text{ nearest nbrs of } x_q\}} K(d(x_q, x))\left(f(x) - \hat{f}(x)\right)x_j$$



Note: $x_i = a_i(x)$

Radial Basis Function Networks

- Distance-weighted regression and artificial neural networks

$$\hat{f}(x) = w_0 + \sum_{u=1}^{k} w_u K_u(d(x_u, x))$$

  - x_u : instance from X
  - $K_u(d(x_u, x))$: kernel function

- The contribution from each of the $K_u(d(x_u, x))$ terms is localized to a region nearby the point x_u: Gaussian function
- Corresponding to a two-layer network
  - First layer: computes the values of the various $K_u(d(x_u,x))$
  - Second layer: computes a linear combination of the first-layer unit values.

- Training
  - Construct kernel functions
  - Adjust weights
- RBF networks provide a global approximation to the target function, represented by a linear combination of many local kernel functions.

  $\hat{f}(x)$: global approximation to $f(x)$
  $K_u(d(x_u, x))$ terms are localized to x_u

- Case-Based Reasoning (CBR)
  - Lazy learning: defer the decision of how to generalize beyond the training data until a new query instance is observed.
  - Classify new query instances by similar instances
  - However, CBR does not represent instances as real-valued points in an *n*-dimensional Euclidean space.
  - In CBR, instances are represented using symbolic descriptions.
- Applications of CBR
  - Conceptual design of mechanical devices
  - Reasoning about new legal cases based on previous rulings
  - Solving planning and scheduling problems by reusing and combining portions of previous solutions to similar problems

Remarks on Lazy and Eager Learning

- Lazy learning: generalization at query time
  - *k*-NN
  - Locally weighted regression
  - Case-based reasoning
- Eager learning: generalization at training time
  - Radial basis function networks
  - Back-propagation networks (multilayer perceptrons)
- Computation time
  - training: eager > lazy
  - query: eager < lazy
- Classifications produced for new queries.
- Target function
  - Eager: a single linear function that covers the entire instance space
  - Lazy: a combination of many local approximations

Summary

- Locally weighted regression methods are a generalization of *k*-nearest neighbor in which an explicit local approximation to the target function is constructed for each query instance

- Radial basis function (RBF) networks are a type of artificial neural network constructed from spatially localized kernel functions.

RBF Approaches to Nonlinear Regression: A More General Formulation

1. Problem Formulation

Given the training data set:

$$D = \{\mathbf{x}^{(n)}, t_n\}_{n=1}^{N} = \{\mathbf{X}_N, \mathbf{t}_N\}$$

with $\mathbf{X}_N \equiv \{\mathbf{x}^{(n)}\}_{n=1}^{N}$ and $\mathbf{t}_N \equiv \{t_n\}_{n=1}^{N}$

We wish to predict

t_{n+1} given the unknown (query) input $\mathbf{x}^{(n+1)}$

$$P\left(y(\mathbf{x}) \mid \mathbf{t}_N, \mathbf{X}_N\right) = \frac{P\left(\mathbf{t}_N \mid y(\mathbf{x}), \mathbf{X}_N\right) P\left(y(\mathbf{x})\right)}{P\left(\mathbf{t}_N \mid \mathbf{X}_N\right)}$$

Remark: For the prediction of t, parameterization of the function $y(\mathbf{x};\mathbf{w})$ is irrelevant. This is the basic idea underlying the Gaussian processes, as we will see later on.

2. Using a radial basis function network

$$y(\mathbf{x};\mathbf{w}) = \sum_{h=1}^{H} w_h \phi_h(\mathbf{x}) = \sum_{h=1}^{H} w_h \exp\left[-\frac{\left(\mathbf{x}-\mathbf{c}_h\right)^2}{2r^2}\right]$$

RBFs centered at fixed points $\{\mathbf{c}_h\}_{h=1}^{H}$

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left\{t_n - y(\mathbf{x}^{(n)};\mathbf{w})\right\}^2$$

$$\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = 0$$

Learning = error minimization

Important to note: The error function $E(\mathbf{w})$ is related with the probability function $P\left(\mathbf{t}_N \mid \mathbf{w}, \mathbf{X}_N\right)$

in the following way. Assuming the errors are Guassian distributed (which is reasonable), we have

$$P(\mathbf{t}_N \mid \mathbf{w}, \mathbf{X}_N) = Z^{-1} \exp(-\beta E(\mathbf{w}))$$
$$= Z^{-1} \exp\left(-\beta \frac{1}{2} \sum_{n=1}^{N} \{t_n - y(\mathbf{x}^{(n)}; \mathbf{w})\}^2 \right)$$

Maximization of the probability is equivalent to the minimization of the error function since the following holds:

$$\arg\max_{\mathbf{w}} \{P(\mathbf{t}_N \mid \mathbf{w}, \mathbf{X}_N)\}$$
$$= \arg\max_{\mathbf{w}} \{\ln P(\mathbf{t}_N \mid \mathbf{w}, \mathbf{X}_N)\}$$
$$= \arg\min_{\mathbf{w}} \{-E(\mathbf{w})\}$$
$$= \arg\min_{\mathbf{w}} \{E(\mathbf{w})\}$$
$$= \arg\min_{\mathbf{w}} \left\{ \sum_{n=1}^{N} \{t_n - y(\mathbf{x}^{(n)}; \mathbf{w})\}^2 \right\}$$
$$= \arg\min_{\mathbf{w}} \left\{ \sum_{x \in D} (f(x) - \hat{f}(x))^2 \right\} \qquad \text{(using the previous notation)}$$

$$E_2(x_q) \equiv \frac{1}{2} \sum_{x \in D} \left(f(x) - \hat{f}(x) \right)^2 K(d(x_q, x))$$

e.g.: $K(d(x_q, x)) = \exp(-d(x_q, x))$

Thus:

General (Bayesian) Methods for Nonlinear Regression

$$P(\mathbf{w} \mid \mathbf{t}_N, \mathbf{X}_N) = \frac{P(\mathbf{t}_N \mid \mathbf{w}, \mathbf{X}_N) P(\mathbf{w})}{P(\mathbf{t}_N \mid \mathbf{X}_N)}$$

1. Laplace method

$$M(\mathbf{w}) = -\ln\left[P(\mathbf{t}_N \mid \mathbf{w}, \mathbf{X}_N) P(\mathbf{w}) \right] \qquad (P(\mathbf{w}): \text{prior dist. of } \mathbf{w})$$

$$\frac{\partial M(\mathbf{w})}{\partial \mathbf{w}} = 0$$

$$P(t_{N+1} \mid \mathbf{t}_N, \mathbf{X}_{N+1}) \cong P(t_{N+1} \mid \mathbf{w}^*, \mathbf{x}^{(N+1)})$$
$$\text{with} \quad \mathbf{w}^* = \arg\min_{\mathbf{w}} \{M(\mathbf{w})\}$$

Note: The error minimization approach described above is a simplified version of the Laplace method.

Why? And in what sense?

2. Markov chain Monte Carlo (MCMC) method

- Create samples from $P(\mathbf{w} \mid \mathbf{t}_N, \mathbf{X}_N)$

- Predict

$$\begin{cases} P(t_{N+1} \mid \mathbf{t}_N, \mathbf{X}_{N+1}) = \int_W P(t_{N+1}, \mathbf{w} \mid \mathbf{t}_N, \mathbf{X}_{N+1}) d^H \mathbf{w} \\ \qquad\qquad = \int d^H \mathbf{w} P(t_{N+1} \mid \mathbf{w}, \mathbf{x}^{(N+1)}) P(\mathbf{w} \mid \mathbf{t}_N, \mathbf{X}_N) \\ P(t_{N+1} \mid \mathbf{t}_N, \mathbf{X}_{N+1}) \cong \frac{1}{R} \sum_{r=1}^{R} P(t_{N+1} \mid \mathbf{w}^{(r)}, \mathbf{x}^{(N+1)}) \end{cases}$$

Note: If time permits, we shall learn later on more about the stochastic sampling methods including MCMC.

Summary: Criteria for Learning

1. $P(\mathbf{w} \mid \mathbf{t}_N, \mathbf{X}_N) = \dfrac{P(\mathbf{t}_N \mid \mathbf{w}, \mathbf{X}_N) P(\mathbf{w})}{P(\mathbf{t}_N \mid \mathbf{X}_N)}$: full Bayesian approach (distribution)

   (= MCMC methods)

2. $\mathbf{w}^* = \arg\max_{\mathbf{w}} \{P(\mathbf{w} \mid \mathbf{t}_N, \mathbf{X}_N)\}$

   $= \arg\max_{\mathbf{w}} \{P(\mathbf{t}_N \mid \mathbf{w}, \mathbf{X}_N) P(\mathbf{w})\}$: maximum a posteriori (MAP) approach

   (= Laplace methods)

3. $\mathbf{w}^* = \arg\max_{\mathbf{w}} \{P(\mathbf{t}_N \mid \mathbf{w}, \mathbf{X}_N)\}$: maximum likelihood (ML) approach

4. $\mathbf{w}^* = \arg\max_{\mathbf{w}} \{P(\mathbf{t}_N \mid \mathbf{w}, \mathbf{X}_N)\}$

   $= \arg\max_{\mathbf{w}} \{Z^{-1} \exp[-\beta E(\mathbf{w} \mid \mathbf{t}_N, \mathbf{X}_N)]\}$

   $= \arg\min_{\mathbf{w}} \{E(\mathbf{w} \mid \mathbf{t}_N, \mathbf{X}_N)\}$: error minimization approach

   or least mean squared (LMS) error approach

7.6 Gaussian Processes

[MacKay, Chap. 45]

이번 학기(2006-2) 시험범위에서는 제외. 그렇지만 RBF와 Kernel Machine의 관계를 이해하는데 있어서 아주 중요.

Problem formulation for nonlinear regression (again):

Given N data points: $(\mathbf{X}_N, \mathbf{t}_N) = \{\mathbf{x}^{(n)}, \mathbf{t}_n\}_{n=1}^N$

We wish to predict:

$$P\left(y(\mathbf{x}) \mid \mathbf{t}_N, \mathbf{X}_N\right) = \frac{P\left(\mathbf{t}_N \mid y(\mathbf{x}), \mathbf{X}_N\right) P\left(y(\mathbf{x})\right)}{P\left(\mathbf{t}_N \mid \mathbf{X}_N\right)}$$

Note that for the prediction of t, parameterization of the function $y(\mathbf{x};\mathbf{w})$ is irrelevant. That is, we just need to know the values of $y(\mathbf{x})$, not the exact form of $y(\mathbf{x})$. This is the basic idea underlying the Gaussian processes, as we shall see below.

From Parametric Models to Gaussian Processes

Generally, we consider a regression problem using H fixed basis functions $\{\phi_n(\mathbf{x})\}_{h=1}^H$, such as radial basis functions $y(\mathbf{x};\mathbf{w}) = \sum_{h=1}^H \mathbf{w}_h \phi_h(\mathbf{x})$.

Given: $\{\mathbf{x}^{(n)}\}$: a list of input points (specified)

$\mathbf{R} = [R_{nh}]$: matrix of one values of the basis functions $\{\phi_n(\mathbf{x})\}_{h=1}^H$ at the points $\{\mathbf{x}^{(n)}\}$

$$R_{nh} \equiv \phi_h(\mathbf{x}^{(n)})$$

$$\mathbf{R} = [R_{nh}] = \left[\phi_h(\mathbf{x}^{(n)})\right] = \begin{array}{c} \\ \mathbf{x}^{(1)} \\ \mathbf{x}^{(2)} \\ \vdots \\ \mathbf{x}^{(N)} \end{array} \begin{array}{c} \begin{array}{cccc} \phi_1 & \phi_2 & \cdots & \phi_H \end{array} \\ \begin{pmatrix} R_{11} & R_{12} & \cdots & R_{1H} \\ R_{21} & R_{22} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ R_{N1} & R_{N2} & \cdots & R_{NH} \end{pmatrix} \end{array}$$

$\mathbf{y}_N$: vector of values of $y(\mathbf{x})$ at the N points.

$$y_n \equiv \sum_h R_{nh} w_h$$

1. If the prior distribution of $\mathbf{w}$ is Gaussian with zero mean

$$P(\mathbf{w}) = Normal(\mathbf{w}; \mathbf{0}, \sigma_w^2 \mathbf{I}),$$

then $\mathbf{y}$, being a linear function of $\mathbf{w}$, is also Gaussian distributed with mean zero.

2. The covariance matrix of $\mathbf{y}$ is

$$\mathbf{Q} = \left\langle \mathbf{y}\mathbf{y}^T \right\rangle = \left\langle \mathbf{R}\mathbf{w}\mathbf{w}^T\mathbf{R}^T \right\rangle = \mathbf{R}\left\langle \mathbf{w}\mathbf{w}^T \right\rangle \mathbf{R}^T$$
$$= \sigma_w^2 \mathbf{R}\mathbf{R}^T$$

where $<\cdot>$ denotes the expectation operator.

3. So the prior distribution of $\mathbf{y}$

$$P(\mathbf{y}) = Normal(\mathbf{y}; \mathbf{0}, \mathbf{Q}) = Normal(\mathbf{y}; \mathbf{0}, \sigma_w^2 \mathbf{R}\mathbf{R}^T)$$

This is true for any selected points $\mathbf{X}_N$.

For any $\mathbf{X}_N$, $P(y(\mathbf{x}^{(1)}), y(\mathbf{x}^{(2)}), \cdots, y(\mathbf{x}^{(N)}))$ is Gaussian distributed.

Probability distribution of a function $y(\mathbf{x})$ is a Gaussian process.

What about the Target Values t_n ?

$$t_n = y_n + \varepsilon_n, \qquad \varepsilon_n \sim Normal(0, \sigma_v^2),$$

$$P(\mathbf{t}) = Normal(\mathbf{t}; \mathbf{0}, \mathbf{Q} + \sigma_v^2 \mathbf{I})$$

The covariance matrix of $\mathbf{t}$

$$\mathbf{C} = \mathbf{Q} + \sigma_v^2 \mathbf{I} = \sigma_w^2 \mathbf{R}\mathbf{R}^T + \sigma_v^2 \mathbf{I}$$

In general, (n, n')-entry of $\mathbf{Q}$ and $\mathbf{C}$ are

$$Q_{nn'} = \left[\sigma_w^2 \mathbf{R}\mathbf{R}^T\right]_{nn'} = \sigma_w^2 \sum_h \phi_h\left(\mathbf{x}^{(n)}\right)\phi_h\left(\mathbf{x}^{(n')}\right)$$: product of basis functions ➔ kernels

$$C_{nn'} = \sigma_w^2 \sum_h \phi_h\left(\mathbf{x}^{(n)}\right)\phi_h\left(\mathbf{x}^{(n')}\right) + \delta_{nn'}\sigma_v^2$$: product of basis functions ➔ kernels

This can be generalized as follows:

Given any valid covariance function $C(\mathbf{x}, \mathbf{x}')$, we can define the covariance matrix Q for N function values at locations $\mathbf{X}_N$ by

$$Q_{nn'} = C(\mathbf{x}^{(n)}, \mathbf{x}^{(n')})$$

and the covariance matrix for N corresponding target values to be matrix $\mathbf{C}$ given by

$$C_{nn'} = C(\mathbf{x}^{(n)}, \mathbf{x}^{(n')}) + \sigma_v^2 \delta_{nn'}$$

$$\delta_{nn'} = \begin{cases} 1 & \text{if } n = n' \\ 0 & \text{otherwise} \end{cases}$$

In conclusion, the prior probability of the N target values $\mathbf{t}$ in the data set is

$$P(\mathbf{t}) = Normal(\mathbf{t}; \mathbf{0}, \mathbf{C}) = \frac{1}{Z} \exp\left[-\frac{1}{2} \mathbf{t}^T \mathbf{C}^{-1} \mathbf{t} \right]$$

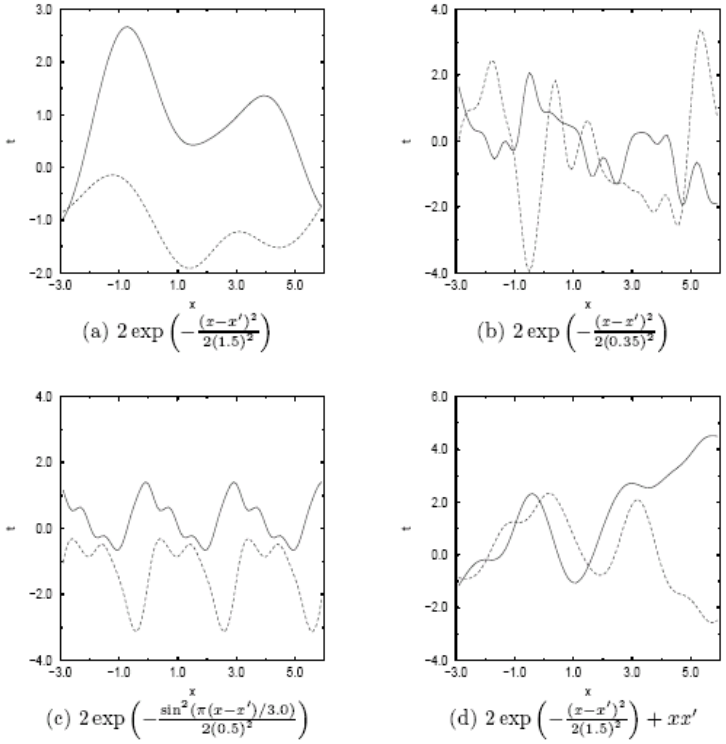


Figure 45.1. Samples drawn from Gaussian process priors. Each panel shows two functions drawn from a Gaussian process prior. The four corresponding covariance functions are given below each plot. The decrease in lengthscale from (a) to (b) produces more rapidly fluctuating functions. The periodic properties of the covariance function in (c) can be seen. The covariance function in (d) contains the non-stationary term xx' corresponding to the covariance of a straight line, so that typical functions include linear trends. From Gibbs (1997).

Using a Given Gaussian Process Model in Regression

[MacKay, Chap. 45.3]

The task is to infer t_{N+1} given the observed vector $\mathbf{t}_N$:

$$P(t_{N+1} \mid \mathbf{t}_N) = \frac{P(t_{N+1}, \mathbf{t}_N)}{P(\mathbf{t}_N)}$$

$$C_{N+1} = \begin{bmatrix} [\mathbf{C}_N] & [\mathbf{k}] \\ [\mathbf{k}^T] & [\kappa] \end{bmatrix}$$

The posterior distribution is given by

$$P(t_{N+1}\,|\,\mathbf{t}_N) \propto \exp\left[-\frac{1}{2}\left[\mathbf{t}_N t_{N+1}\right]\mathbf{C}_{N+1}^{-1}\begin{bmatrix}\mathbf{t}_N\\ t_{N+1}\end{bmatrix}\right] \quad (45.36)$$

Compute $\mathbf{C}_{N+1}^{-1}$ by brute-force inversion.

$\mathbf{C}_{N+1}^{-1}$ is computed from $\mathbf{C}_{N+1}$ and $\mathbf{C}_{N+1}^{-1}$ using partitioned inverse equation (Barnett, 1979)

$$\mathbf{C}_{N+1}^{-1} = \begin{bmatrix}\mathbf{M} & \mathbf{m}\\ \mathbf{m}^T & m\end{bmatrix}$$

where $m = (\kappa - \mathbf{k}^T\mathbf{C}_N^{-1}\mathbf{k})^{-1}$

$$\mathbf{m} = -m\mathbf{C}_N^{-1}\mathbf{k}$$

$$\mathbf{M} = \mathbf{C}_N^{-1} + \frac{1}{m}\mathbf{m}\mathbf{m}^T$$

Substituting this matrix into (45.36), we find

$$P(t_{N+1}\,|\,\mathbf{t}_N) = \frac{1}{Z}\exp\left[-\frac{\left(t_{N+1}-\hat{t}_{N+1}\right)^2}{2\sigma_{\hat{t}_{N+1}}^2}\right]$$

where $\hat{t}_{N+1} = \mathbf{k}^T\mathbf{C}_N^{-1}\mathbf{t}_N$ is the predictive mean and $\sigma_{\hat{t}_{N+1}}^2 = \kappa - \mathbf{k}^T\mathbf{C}_N^{-1}\mathbf{k}$ represents the error bar.

The predictions produced by a Gaussian process depends *entirely* on the covariance matrix $\mathbf{C}$.

Questions: Where is the prior knowledge? Where are the parameters gone? Is this a learning from data?

Examples of Covariance Functions (= Kernels)

$$C(\mathbf{x},\mathbf{x}';\boldsymbol{\theta}) = D(\mathbf{x}-\mathbf{x}';\boldsymbol{\theta})$$

$$C(\mathbf{x},\mathbf{x}';\boldsymbol{\theta}) = \theta_1\exp\left[-\frac{1}{2}\sum_{i=1}^{I}\frac{(x_i-x_i')^2}{r_i^2}\right] + \theta_2, \quad \boldsymbol{\theta} = \left(\theta_1,\theta_2,\{r_i\}\right)$$

$$C(\mathbf{x},\mathbf{x}') = \exp\left(-|\,\mathbf{x}-\mathbf{x}'\,|^{\nu}\right), \qquad 0 < \nu \le 2$$

$$C(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}) = \theta_1 \exp\left[-\frac{1}{2} \sum_{i=1}^{I} \left(\frac{\sin\left(\frac{\pi}{\lambda_i}(x_i - x_i') \right)}{r_i} \right)^2 \right]$$

$$C_{\text{lin}}(\mathbf{x}, \mathbf{x}'; \{\sigma_w, \sigma_c\}) = \sum_{i=1}^{I} \sigma_w^2 x_i x_i' + \sigma_c^2$$

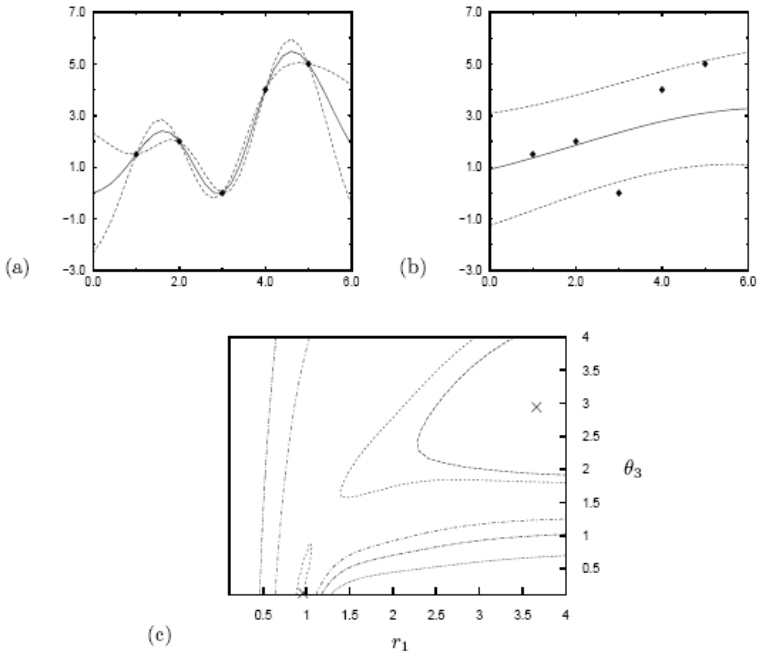


Figure 45.2. Multimodal likelihood functions for Gaussian processes. A data set of five points is modelled with the simple covariance function (45.47), with one hyperparameter θ_3 controlling the noise variance. Panels a and b show the most probable interpolant and its 1σ error bars when the hyperparameters θ are set to two different values that (locally) maximize the likelihood $P(\mathbf{t}_N \mid \mathbf{X}_N, \theta)$: (a) $r_1 = 0.95$, $\theta_3 = 0.0$; (b) $r_1 = 3.5$, $\theta_3 = 3.0$. Panel c shows a contour plot of the likelihood as a function of r_1 and θ_3, with the two maxima shown by crosses. From Gibbs (1997).

Problems of Gaussian processes: Computational cost for inverting an N x N matrix is prohibitive, where N is the number of training examples.