Chap. 7 Methods for Regression

7.1 Dictionary vs. Kernel
- Nonadaptive vs. Adaptive
- Linear method
- Nonadaptive method
- Adaptive dictionary method

7.1 Taxonomy: Dictionary vs. Kernel Representation

- Approximating function
- Parameterization
- Estimation
- Regression
- Mean squared-error

\[
 f_m(x, w, v) = \sum_{i=1}^{m} w_i g_i(x, v_i) + w_0
\]

- \( g_i(x, v_i) \): basis function
- \( v_i = [v_{i1}, v_{i2}, ... , v_{ip_i}] \): basis function parameters
- \( w = [w_0, ..., w_m] \): linear combination coefficients

Regression's goal
- Minimizes mean squared-error estimation
Dictionary method
- basis function
Basis function
1. Fixed basis function
   \[ f_m(x, w) = \sum_{i=1}^{m} w_i g(x) + w_0 \]
   - Non-adaptive method
2. Adaptive basis function
   - basis function
   - adaptive basis function
   \[ f_m(x, w, v) = \sum_{i=1}^{m} w_i g(x, v) + w_0 \]

**Radial basis function (RBF) network**
\[ g(x, v_i) = g(\| x - v_i \|) = K(\| x - v_i \|) \]
- univariate function
1. Local radial basis functions (kernel function \( K \))
   - Gaussian: \( g(t) = \exp(-\frac{t^2}{2\alpha}) \)
   - Multiquadratic: \( g(t) = (t^2 + b^2)^{-\alpha} \)
2. Non-local radial basis function
   \( g(t) = t^2 \ln(t) \)

**Multilayer perceptron (MLP)**
\[ g(x, v_i) = s(v_0 + \sum_{k=1}^{d} x_i v_{ik}) = s(x \cdot v_i) \]
- Sigmoid (logistic)
  \[ s(t) = \frac{1}{1 + \exp(-t)} \]
- Sigmoid (hyperbolic tangent)
  \[ s(t) = \tanh(t) = \frac{\exp(t) - \exp(-t)}{\exp(t) + \exp(-t)} \]
MLP, RBF network의 graphical form

- Parameters: network weights
- Input (output) variables: input (output) nodes
- Basis function: hidden layer unit

Adaptive basis functions
- univariate
- $x,v$ 에 대해서 symmetric

Basis function
- (nonlinear) feature
- Basis function의 optimal selection
  - Feature selection이라고 할 수 있다.

Kernel methods

$$f(x) = \sum_{i=1}^{n} K(x, x_i) y_i$$

- $K(x, x_i)$: 주로 다음의 특성을 만족시키는 symmetric function

$$K(x, x') \geq 0 \quad \text{Nonnegative}$$
$$K(x, x') = K(||x - x'||) \quad \text{Radially symmetric}$$
$$K(x, x) = \max x = x'\text{일때 최대값}$$
$$\lim_{t \to \infty} K(t) = 0 \quad \text{단조감소}$$

Kernel function의 두 종류
1. Kernel function의 값이 training data의 $x_i$ 값에만 의존하는 경우
   - Nonadaptive
   - Nonadaptive dictionary method와 대응되어, 서로 변환이 가능하다.
2. Kernel function의 선택이 training data의 $y$ 값에 도 영향을 받는 경우
   - Adaptive kernel method
Dictionary method에서 optimal least squares solution이 발견되면 이는 바로 동등한 kernel method의 형태로 변환될 수 있다.

대부분의 adaptive method에서는 kernel method보다는 dictionary method가 쓰인다.
* 이는 dictionary method에서는 global function을 사용하며 모든 training data를 이용하기 때문이 다.

Nonadaptive kernel method에서는 고정된 kernel span \( \alpha \)가 사용된다.

### 7.2 Linear Estimator

Regression estimator가 다음의 superposition principle을 만족시키면 linear라 한다.

\[
\hat{f}_0(a'y' + b'y'') | \mathbf{X} = af_1(y' | \mathbf{X}) + bf_2(y'' | \mathbf{X})
\]

- nonzero \( a, b \)
- \( f_0, f_1, f_2 \): 하나의 approximating functions 집합에서의 estimate
- \( \mathbf{X} = (x_1, \ldots, x_n) \): predictor sample
- \( y', y'' \): 두 개의 response 값들

#### Linear approximating function의 두 가지 표현 방법

1. Fixed basis function의 linear combination
   - fixed basis functions의 선택: learning problem에 대한 priori knowledge
   - learning: empirical risk나 penalized risk의 최소화에 따른 linear coefficients의 선택
2. Training data의 kernel average
   - kernel function의 형식: priori knowledge

#### Kernel function의 두 가지 예

- Kernel density estimation
- Equivalent basis function representation of a linear estimator

#### Kernel density estimation

- Approximating function
  \[
  \hat{\rho}(x) = \frac{1}{n} \sum_{i=1}^{n} K_{\alpha}(x, x_i)
  \]
  - Kernel function에 또 필요한 특성
  \[
  \int_{-\infty}^{\infty} K(x, x') dx' = 1 \quad \text{for any} \ x
  \]
- Kernel regression approximating function
  \[
  f_{\alpha}(x, w_n | x_n) = \sum_{i=1}^{n} w_i K_{\alpha}(x, x_i)
  \]
  - Local symmetric neighborhood near \( x \)
7.2.1 Estimation of Linear Models and Equivalence of Representations

Least squares solution for estimating \( w \)

\[
Zw \equiv y
\]

\[
Z = \begin{bmatrix}
g_1(x_1) & \ldots & g_m(x_1) \\
\vdots & \ddots & \vdots \\
g_1(x_n) & \ldots & g_m(x_n)
\end{bmatrix} = [g_1(X) \mid g_2(X) \mid \ldots \mid g_m(X)]
\]

Empirical risk

\[
R_{emp}(w) = \frac{1}{n} \|Zw - y\|^2
\]

Solution

\[
Z^TZw = Z^Ty
\]

\( Z \)가 linearly independent하면 unique solution이 존재한다.(대부분의 경우가 그렇다. \( m \leq n \))

\[
w^* = (Z^TZ)^{-1}Z^Ty
\]

Parametric penalization 0 linear estimator에 적용된 경우

Penalized risk functional

\[
R_{pen}(w) = \frac{1}{n} (\|Zw - y\|^2 + w^T\Phi w)
\]

\( \Phi: m \times m \) penalty matrix

Regularization parameter \( \lambda \) is absorbed in \( \Phi \)

Solution

\[
w^* = (Z^TZ + \Phi)^{-1}Z^Ty
\]
modified least squares

1. 다음의 행렬을 만드는다.
   \[ U = \begin{bmatrix} Z \\ A \end{bmatrix}, \quad v = \begin{bmatrix} y \\ 0 \end{bmatrix} \]

- \( Z \): 주어진 data matrix
- \( \Phi = A^T A \)
- \( 0: m \)개의 0을 가지는 column vector
- 결국은 관찰된 data에 인공적인 data를 첨가한 것과 같다.

2. 다음의 empirical risk functional을 minimize

   \[ R_{emp} = \frac{1}{n} \| Uw - v \|_2^2 \]

\[ \text{Solution} \]

\[ w^* = (U^T U)^{-1} U^T v = (Z^T Z + A^T A)^{-1} Z^T y \]

\[ \text{Training data에 artificial example을 포함시키는 방법} \]

- training sample의 개수에 따라 얼마나 artificial example을 포함시킬 것인가에 대한 문제가 있다.

\[ \text{Matrix S는 다음과 같다.} \]

\[ S = Z(Z^T Z)^{-1} Z^T, \quad S_\phi = Z(Z^T Z + \Phi)^{-1} Z^T \]

\[ (\text{penalized}) \text{ least squares solution} \text{에는 임의의 벡터} y \text{를 column space} Z \text{에 project시키는 projection matrix} S \text{가 존재한다.} \]

\[ \hat{y} = Z w^* = S y \]
Matrix $S$ is optimal basis function estimate of $w^*$ and is called kernel. The kernel function is defined as:

$$S(x, x_i) = g(x)(Z^T Z)^{-1} g^T(x_i)$$

$$S_\Phi(x, x_i) = g(x)(Z^T Z + \Phi)^{-1} g^T(x_i)$$

Kernel function is symmetric, and it has the property of basis function expansion. This can only hold if the kernel function is symmetric. The eigen function decomposition of the eigen values and eigen functions can be expressed as:

$$K(x, x') = \sum_{i=1}^\infty \epsilon_i g_i(x) g_i(x')$$

- $\epsilon_i$: eigen values
- $g_i(x)$: eigen function
7.2.2 Analytic Form of Cross-Validation

Matrix $S$ is defined by the linear estimate and leave-one-out cross-validation's expected risk can be computed analytically. This is especially useful for resampling as compared to the computational cost.

$$y_i' = \hat{y}_i = \frac{1}{1-s^*} \sum_{j=1, j \neq i}^n s_{ij} y_j$$

$$\hat{y}' = S'y$$

7.2.3 Estimating Complexity of Penalized Linear Models

Basis function is used in linear model. In this case, the free parameter's number is the model complexity.

Free parameter's number is determined when kernel representation's eigenvalue is significant term. This is done to determine the model complexity.

- Eigen function is orthogonal and symmetric kernel's eigenvalue is nonnegative.
- Degree of freedom is determined from eigen decomposition.
7.3 Nonadaptive Methods

- Nonadaptive methods (linear estimators)
- Statistics: local polynomial estimators, splines
- Neural nets: RBF networks
- Signal processing: wavelet methods

\[ f_m(x, w) = \sum_{i=1}^{m} w_i g_i(x) + w_0 \]

7.3.1 Local Polynomial Estimators and Splines

- Spline: 일련의 local하게 정의된 low order polynomial들

\[ f_m(x, w, v) = \sum_{j=1}^{m} w_j g_j(x, v_j) + w_0 \]

- basis function: spline basis
- \( v_j \): knot location
- \( m \): knot의 갯수

![Fig. 7.8 A ninth order polynomial and a cubic spline interpolation of two data points. The cubic spline provides an interpolation with minimum curvature.](image)
1. Nonadaptive knot selection
   - x만을 이용하여 knot 결정
   - knot이 결정되면 linear least square 문제

2. Adaptive knot selection
   - 함수값 y도 knot 결정에 이용

Regularization Framework와 Cubic Splines의 연결

- Regularization problem

\[ R_{\text{pen}}(f) = \sum_{i} [f(x_i) - y_i]^2 + \lambda \int [f'(t)]^2 dt \]

- \( \lambda \): fixed complexity parameter
- \( a \leq x_1 < \ldots < x_n \leq b \)

- Solution

\[ f(x) = \sum_{j=1}^{n} w_j B_j(x) \]

λ: fitting the data와 smoothness 간의 trade-off 조절

- \( w_j \)'s 결정: linear estimation problem with penalty

- Nonparametric penalty도 parametric하게 할 수 있다.
  - Penalty matrix

\[ \phi_{ij} = \lambda \int B_i(t) B_j'(t) dt \]

Multivariate function approximation을 위한 generalization of univariate splines

- Tensor product of \( d \) univariate splines

- Example1: Gaussian radial basis

\[ g(x, v) = \prod_{j=1}^{d} \exp \left( -\frac{(x_j - v_j)^2}{\alpha} \right) = \exp \left( -\frac{\|x - v\|^2}{\alpha} \right) \]

- Example2: Tensor-product truncated power basis

\[ g(x, v, \alpha) = \prod_{j=1}^{d} [B_j(x_j - v_j)]^8 \]
7.3.2 Radial Basis Function Networks

- **Approximating function**
  \[ f_m(x, w) = \sum_{j=1}^{m} w_j g \left( \frac{\| x - v_j \|}{\alpha_j} \right) + w_0 \]

- **Normalized RBF**
  \[ f_m(x, w) = \frac{\sum_{j=1}^{m} w_j g_j}{\sum_{k=1}^{m} g_k} \]

7.4 Adaptive Dictionary Methods

- **Multivariate problem: adaptive method**
- **Adaptive method**
  - All basis functions of the same/different type
  - Type of basis functions
    - type of dimensionality reduction
    - bounded/unbounded basis functions
  - Optimization strategy

7.4.1 Additive Methods and Projection Pursuit Regression

- **Additive model**
  \[ f(x, V) = \sum_{j=1}^{m} g_j(x, v_j) + w_0 \]
  - \( v_j \): internal parameters (kernel width)
  - \( g_j(x, v_j) \): can be kernel smoother
  - kernel width: data에 맞도록 조정된다.
Projection Pursuit

- Additive model의 specific form (univariate function)
  \[ f(x, V, W) = \sum_{j=1}^{m} g_j(w_j \cdot x, v_j) + w_0 \]

- Invariant to affine coordinate transformation (rotation, scaling) of the input variables

Backfitting (find local minimum of the empirical risk)

- additive approximating function에 대한 empirical risk의 분해
  \[ R_{emp}(V) = \frac{1}{n} \sum (y_i - f(x_i, V))^2 \]
  \[ = \frac{1}{n} \sum (y_i - \sum g_j(x_i, v_j) - w_0 - g_0(x_i, v_0))^2 \]
  \[ = \frac{1}{n} \sum (r_i - g_j(x_i, v_j))^2 \]

- \( k \)번째를 제외한 basis function을 고정시키면 risk는 “unexplained” variance로 분해
- Initial set of basis function들을 가지고 잔의의 \( k \)에 대한 \( r_i \)를 구할 수 있다.
- \( g_j \)의 parameter는 이를 variance를 minimize하도록 조정된다.
- 차례로 각 basis function을 estimate한다.

Backfitting algorithm

1. 모든 \( x \)에 대해 \( g_j(x, v_j) = 0 \)이 되도록 initialize
   \( w_0 \)는 \( 1/n \sum y_i \)
2. 모든 \( k \)에 대해 \( r_i \)를 계산하고 risk를 최소화하는 \( v_k \)를 구한다.
3. stopping criteria를 만족할 때까지 반복

- Projection pursuit
  - Specific form of backfitting
  - Steepest descent method
Projection pursuit algorithm

1. Backfitting
   a. Residual $r_i$
   b. Projection pursuit
      - $w_k$, $\gamma$를 찾는다.
        i. univariate smoother
        ii. $w_k$를 경신

2. Residual
   a. Projection pursuit
      - $w_k$를 경신

3. Backfitting

Implementation

- SMART (smooth multiple additive regression technique)
- Supersmoother (Friedman, 1984)
- Hermite polynomial
- Linear method $g_j$에 대해서는 global minimum을 구함
- Nonlinear $g_j$에 대한 convergence는 보장되지 않는데다.
- Basis function의 growing, pruning 사용
- Complexity의 estimation은 어렵다.

7.4.2 Multilayer Perceptrons and Backpropagation

- Basis function
  \[ g_j(x, v_j) = s(x \cdot v_j) \]
  - Universal approximator
  - In terms of representation
    - MLP is a specific case of projection pursuit(고정된 basis function)
  - Projection is a specific case of MLP
    - univariate basis function can be represented as a sum of shifter sigmoids

- Target function that vary significantly only in a few directions
  - projection pursuit outperforms MLP
  - Estimating a large number of projections
    - MLP outperforms projection pursuit
  - Two properties of MLP network
    - Smooth well-behaved sigmoid(saturation limit)
    - Regularization properties of the backpropagation algorithm
Backpropagation algorithm

- **Output layer**
  \[ \delta_o(k) = y(k) - y(k) \]
  \[ w_j(k+1) = w_j(k) - \gamma \delta_o(k) z_j(k) \]

- **Hidden layer**
  \[ \delta_{1j}(k) = \delta_o(k) s'(a_j(k)) w_j(k+1) \]
  \[ v_{ij}(k+1) = v_{ij}(k) - \gamma \delta_{1j}(k) x_i(k) \]

- **Forward pass**
  \[ a_j = \sum_{i=0}^{d} x_i v_{ij} \]
  \[ z_j = g(a_j) \]
  \[ z_0 = 1 \]

Adding momentum

\[ w(k+1) = w(k) - \gamma \delta(k) v + \mu (w(k) - w(k-1)) \]

---

On-line (stochastic approximation) and batch

- **On-line**: less likely to be trapped in a local minimum
- **Batch**: accurate estimate of the true gradient

**Learning rate**

- Need to be reduced
- Problem dependent

**Premature saturation**

- Input value, weight가 큰 경우
- Sigmoid의 derivate가 0에 가까운 경우
- 시간이 오래 걸린다.
- Weight를 작게 준다(so tricky and important)
Complexity Control in MLP

1. One-dimensional path through parameter space
   - The solution is on this path
   - and it depends on
     1. training data and its order
     2. the set of nonlinear approximating functions
     3. starting point on the path (initial parameter values)
     4. final point on the path (the stopping rules)

2. Structure on a set of approximating functions
   1. Initialization of parameters
      \[ S_i = \{ A : f(x, w), \| w^0 \| \leq c_i \} \]
      - \( w^0 \): vector of initial parameter values
      - \( A \): optimization algorithm
      - \( c_i \): regularization effect
      2. Stopping rules
      - avoid overfitting
      - early stopping rules: 분석이 어렵다.

3. Dictionary representation
   - optimal number of hidden units
4. Penalization of (large) parameterization values
   - model complexity = “penalized” risk functional
   \[ R_{pen}(\omega, \lambda_i) = R_{emp}(\omega) + \lambda_i \| w \|^2 \]
   - 위의 식은 아래와 같다.
   \[ S_i = \{ f(x, w), \| w \|^2 \leq c_i \}, \quad c_1 < c_2 < c_3 ... \]
   - on-line version에서는 weight decay와 같다.
   \[ w(k + 1) = w(k) - \gamma (\delta(k) z(k) + \lambda w(k)) \]

5. Regularization effect of initialization(실험)
   - Training data
   \[ y = \frac{(x - 2)(2x + 1)}{1 + x^2}, \quad x = [-5, 10] \]
   - 15 training samples
   - \( y \) values: corrupted with gaussian noise
   - input \( x \) is prescaled to [-0.5, 0.5]
6. Network topology
   - 1 input, output node
   - 8 hidden unit, logical sigmoid
7. Backpropagation implementation
   - on-line, no momentum
   - learning rate: 0.5 (fixed)
   - the number of training epochs: 100,000
7.4.3 Multivariate Adaptive Regression Splines

- MARS algorithm
  - Training data에서 adaptive하게 knot locations과 small subset of univariate splines를 선택
  - Recursive partitioning regression
  - Tensor-product splines

- Single linear (\(q = 1\)) tensor product spline basis function

\[
g(\mathbf{x}, \mathbf{u}, \mathbf{v}, \Pi) = \prod_{i=1}^{k} b(x_i, u_i, v_i)
\]
• $b$: univariate basis function
• $v$: knot locations
• $u$: orientation vector $\{-1,1\}$
• $\Pi$: subset of input variables (adaptively selected)
• knot locations: all possible combinations of individual coordinate values existing in the data

$$f_m(x,v,U,v,\{\Pi_1,\ldots,\Pi_m\}) = \sum_{j=1} w_j \prod_{k=\Pi} b(x_k, u_{jk}, v_{jk}) + w_0$$

- Greedy optimization 사용
- Tree 형태
  - left-right pair of basis function $b^+, b^-
  - If $g_{\text{parent}}(x)$ denotes a parent node
    $$g_{\text{daughter}^+}(x) = b^+ (x_k, v_j) \cdot g_{\text{parent}}(x)$$
    $$g_{\text{daughter}^-}(x) = b^- (x_k, v_j) \cdot g_{\text{parent}}(x)$$
  - $v_j$ is knot for $x_k$
  - Depth of a tree: interaction level

Measure of fit: generalized cross validation (gcv) estimate

Model complexity estimate:
1. Determine the degrees of freedom assuming a nonadaptive basis
2. Add a correction factor to take into account the adaptive basis construction
   $$h_{\text{mars}} = (1 + \eta)m$$
   - $m$: estimate for the equivalent degrees of freedom of estimating parameters $w$
   - $\eta$: adaptive correction factor $2 - 4$
Search strategy
- \( m_{\text{max}} \): the maximum number of basis functions
- \( t_{\text{max}} \): limit in the interaction degree

1. Initialization, root node is \( g_0(x) = 1, w_0 \) via mean of the response data
2. Forward stepwise selection, repeat following until the tree has \( m_{\text{max}} \) nodes
   a. Exhaustive search over all valid nodes, all valid split variables, all valid knots. Create a pair of daughters, estimate \( w \) (linear problem), and estimate complexity \( h_{\text{max}} \)
   b. Incorporate the daughters into a tree that result in the largest decrease of prediction risk estimated using \( \text{gcv} \).
3. Backward stepwise selection, repeat the following for \( m_{\text{max}} \) iterations
   a. Exhaustive search over all nodes in the tree, the change in model selection criterion \( \text{gcv} \) resulting from removal of each node.
   b. Delete the node that leads to the largest decrease of \( \text{gcv} \) or smallest increase.
4. Of the series of models created by the backward stepwise selection, choose the best \( \text{gcv} \) score model as the final model.

Interpretation of MARS (by ANOVA decomposition)
- Regrouping the additive terms in function approximation
  \[ \hat{f}(x) = \sum_{i \in I} w_i g_i(x) + w_0 \]
  - Isolate the effect of a particular input variable

Coordinate rotation
- \( \alpha \) controls model complexity
- Large \( \alpha \): low complexity, relatively automatic smoothing parameter selection

7.5 Adaptive Kernel Methods and Local Risk Minimization

- Local estimation
- \( \omega \)는 \( f \) 함수를 고정시키고 \( \alpha \)를 찾는다.
- Neighborhood size \( \alpha \) controls model complexity
- Large \( \alpha \): low complexity

Local empirical risk for estimation point \( x_0 \)
\[
R_{\text{emp-local}}(\omega) = \frac{1}{n} \sum_{i=1}^{n} K_{\alpha}(x_i, x_0)(y_i - f(x_i, \omega))^2
\]
- If approximating function \( f(x, w_0) = w_0 \), zero-order model, risk is minimized when
  \[ f(x_0) = w_0 = \frac{1}{n} \sum_{i=1}^{n} y_i K_{\alpha}(x_i, x_0) \]
- And this is local average, kernel approximation
Approximation is a method to learn by example.
- Memory based: training data is used for prediction. It's important to have enough training data.

**Practical method for kernel width selection**
- $k$-nearest neighbors regression

**$k$-nearest neighbors technique**
- Local risk minimization (take local average of the data)
- Locality: estimation point with the nearest $k$ data points

**$k$'s selection methods**
1. Nonadaptive approach
2. Global adaptive approach
3. Local adaptive approach (for the estimation point)

### 7.5.1 Generalized Memory-Based Learning

- Learn by example
- Local approximation using past data
- Kernel width, distance scale: global
- $k$'s selection (by cross-validation)

- $k$'s selection method
  1. Nonadaptive approach
  2. Global adaptive approach
  3. Local adaptive approach (for the estimation point)

**Risk minimization (locally weighted scatterplot smoothing)**
1. Weighing the data by the kernel function
   \[ x'_i = x_i K_\alpha(x_i, x_0), \quad y'_i = y_i K_\alpha(x_i, x_0) \]
2. Linear estimation

**GMBL kernel**
\[ K(x, x', v) = \left( \sum_{k=1}^{n} (x_k - x'_k)^2 v_k^q \right)^{-q} \]
- $v$: distance scaling parameter
- $q(>0)$: width of kernel function parameter

**Local model selection (small sample problem)**
- $k$'s selection
- Difficult
- Practical approach: global model selection

1. Choose $k$ such that each $x_i$ has a local estimate
2. Global empirical risk is calculated
   \[ R_{emp}(k) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \]
3. Choose $k$ such that the smallest $R_{emp}$ is found
   
   
   
   

**k**'s selection methods
1. Nonadaptive approach
2. Global adaptive approach
3. Local adaptive approach (for the estimation point)**

**Local empirical risk**
\[ R_{emp}(w, w_0) = \frac{1}{n} \sum_{i=1}^{n} K_\alpha(x_i, x_0) [w \cdot x_i + w_0 - y'_i]^2 \]
7.5.2 Constrained Topological Mapping

- CTM:
  - Regression에 적합하도록 SOM을 수정한 것 (clustering via SOM + regression via piecewise-constant splines)
- Center of SOM:
  - Dynamically movable knots for spline regression
- Piecewise-constant spline approximation:
  - Training the SOM with m-dimensional feature space \( (m \leq d) \) using data samples \( x' = (x_i, y_i) \) in \( (d+1) \)-dimensional input space

- Functionality가 보장되지 않는다는 \( (x, y \)의 구별없음)
- Overcome:
  - Dimensionality reduction in the x-space
  - Feature space를 input으로 하여 각 knot에 대한 y 값을 kernel averaging으로 구한다.

Number of knots:
- model complexity
- during the process, neighborhood width decreases

CTM algorithm
- discrete feature space \( \Psi = \{ \Psi_1, ..., \Psi_b \} \)
- data point \( x' = (x_i, y_i) \)
- units \( c_j(k), j = 1, ..., b \)
  1. Determine the nearest unit to the data point
  \[
  z(k) = \Psi (\text{arg min}_i \sum_{j=1}^d (x_i(k) - c_j(k-1))^2)
  \]
  2. Update all units
  \[
  c_j(k) = c_j(k-1) + \beta(k) K_{\text{uni}}(\Psi(j), z(k))(x'(k) - c_j(k-1)),
  j = 1, ..., b \quad \text{for } k = k + 1
  \]
  3. Decrease the learning rate and neighborhood width

CTM lacks in some key features
1. Piecewise-linear vs. piecewise-constant
   - Less accurate than piecewise-linear
2. Control of model complexity
   - By user, iterative cross-validation
3. Adaptive regression via global variable selection
   - No information about variable importance
   - Adaptive predictor variable scaling
4. Batch vs. flow-through implementation
   - Flow-through has some disadvantage
CTM batch version

0. Initialization. Center $c_j$ and distance scale parameter $v_j = 1$

1. Projection.
   \[ \| e_j - x_j \|^2 = \sum_{i=1}^{d} v_j^2 (c_{ij} - x_{ij})^2 \]

2. Conditional expectation (smoothing) in x-space. Update centers
   \[ F(z, \alpha) = \frac{\sum_{i=1}^{n} x_{i} K_{\alpha}(z, z_i)}{\sum_{i=1}^{n} K_{\alpha}(z, z_i)} \]
   \[ c_j = F(\Psi(j), \alpha), \quad j = 1, \ldots, b \]

3. Conditional expectation in y-space. Minimize the following.

4. Adaptive scaling. Determine new scaling parameter $v$ for each of the $d$ input variables using average sensitivity for each predictor dimension
   \[ v_j = \frac{1}{d} \sum_{i=1}^{d} |w_{ij}| \]
   If scaling parameters are normalized, they can be interpreted as variable importance.

5. Model selection. Decrease $\alpha$, and repeat 1 - 4 until the leave-one-out cross-validation reaches a minimum.

7.6 Empirical Comparisons

Comparison of algorithms
- Almost fully automatic methods
- Artificial datasets

Robustness:
- Small change in training data

7.6.1 Experimental Setup

Comparison goal
- Predictive performance
- Non-expert user
- Computational time

Comparison methodology
- Each method is evaluated with different settings and the one with the highest performance is chosen. This way, each method's performance is compared.
Experiment design
- Types of functions used to generate samples
- Properties of the training and test data sets
- Specification of performance metric used for comparisons
- Description of modeling methods

Functions used
- 8개의 representative 2-variable functions
- Gaussian noise: no noise, medium noise, high noise

Test data
- No noise

Performance metric
- RMS of the test set

7.7 Combining Predictive Models

Typical model combination
1. 각각의 model을 training data로 training. Model들의 parameter가 고정.
2. Linear combination of individual models

Committee of networks
- 각 stage에서의 risk minimization

Stacking predictors
- Resampling technique

Learning method implementations
- Projection Pursuit Regression
- MLP
- Multivariate Adaptive Regression Spline
- k-Nearest Neighbor
- Generalized Memory Based Learning
- Constrained Topological Mapping

<table>
<thead>
<tr>
<th>Method</th>
<th>Repl</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projection Pursuit Regression</td>
<td>MLP</td>
<td>KNN, GMBL</td>
</tr>
<tr>
<td>MLP</td>
<td>MARS, PP</td>
<td>KNN, GMBL</td>
</tr>
<tr>
<td>Multivariate Adaptive Regression Spline</td>
<td>CTM, MLP</td>
<td>PP</td>
</tr>
<tr>
<td>k-Nearest Neighbor</td>
<td>MLP, PP</td>
<td>KNN</td>
</tr>
<tr>
<td>Generalized Memory Based Learning</td>
<td>MLP, GMBL</td>
<td>PP</td>
</tr>
<tr>
<td>Constrained Topological Mapping</td>
<td>MLP, GMBL, PP, MARS</td>
<td></td>
</tr>
</tbody>
</table>

Committee of networks

\[
 f_{\text{com}}(x, \alpha) = \sum_{j=1}^{k} \alpha_j f_j(x, \alpha_j) \\
\]
- \( \alpha_j \): degree of belief

\[
 R(\alpha) = \frac{1}{n} \sum_{i=1}^{n} (f_{\text{com}}(x, \alpha) - y_i)^2 \\
\sum_{j=1}^{k} \alpha_j = 1, \quad \alpha_j \geq 0 
\]
Predictor stacking algorithm

1. Resampling. “Left-out” sample \((x_i, y_i)\)와 각각의 candidate method \(f_j(x_i, \omega_j)\)에 대해
   
   a. \(n-1\)개의 sample로 각 model estimate
   
   \[
   f_{ij}^*(x, \omega_{ij}^*)
   \]

   b. prediction for “left-out” sample을 저장
   
   \[
   \hat{y}_{ij} = f_{ij}^*(x, \omega_{ij}^*)
   \]

2. Estimation of linear coefficients.

\[
R(\alpha) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \sum_{j=1}^{b} \alpha_j \hat{y}_{ij})^2
\]

• Additional step
  - 모든 sample을 이용하여 final model을 estimate

\[
f_j^*(x, \omega_j^*)
\]

• Combined model의 구성

\[
f(x) = \sum_{j=1}^{b} \alpha_j f_j^*(x, \omega_j^*)
\]