Nonlinear Optimization Strategies

Chapter 5

Learning from Data

Outline

Introduction

- Finding an appropriate optimization technique is an important in step in creating a learning algorithm.
- For most challenging problems, minimization of the risk functional is a nonlinear optimization.
- For nonlinear optimization, no universal methods provide globally optimal solutions.

Learning Formulation

- Learning and nonlinear optimization
  - Given an inductive principle and a set of parameterized approximating functions, find the function that minimizes a risk functional.
  - ex)

$$\mathrm{Re}_{mp}(\omega) = \sum_{i=1}^{n} Q(\mathbf{z}i, \omega),$$

$Q(\mathbf{z}, \omega)$: loss function

for regression problem, $Q(\mathbf{z}, \omega) = (y - f(\mathbf{x}, \omega))^2$

$\mathbf{z} = [\mathbf{x}, y]$

Three Optimization Strategies

- Stochastic Approximation Methods (Gradient Descent)
- Iterative Methods (Expectation Maximization)
  - Generalized inverse training
  - Expectation Maximization
- Greedy Optimization
  - Network growing (or construction)
  - (with Back fitting and pruning)

5.1 Stochastic Approximation Methods

- Given initial parameter values, optimal parameter values are found by repeatedly updating the values.
- Move a small distance in the direction of steepest descent along the risk surface.
- It must be possible to determine the gradient of the risk functional.
- Can be applied for density estimation, regression, and classification learning problems.

5.1.1 Linear Parameter Estimation

- For regression using a linear approximating functions and L_2 loss function.

$$R_{emp}(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} L(\mathbf{x}_i, y_i, \mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} (y_i - f(\mathbf{x}_i, \mathbf{w}))^2$$

$$\hat{y} = f(\mathbf{x}, \mathbf{w}) = \sum_{j=1}^{m} w_j g_j(\mathbf{x})$$

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \gamma_k \frac{\partial}{\partial \mathbf{w}} L(\mathbf{x}(k), y(k), \mathbf{w})$$

$$\frac{\partial L}{\partial w_j} L(\mathbf{x}, y, \mathbf{w}) = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial w_j} = 2(\hat{y} - y) g_j(\mathbf{x})$$

Linear Parameter Estimation (Cont'd)

- Procedure
  - Step1. Forward pass computations

$$z_j(k) = g_j(\mathbf{x}(k))$$

$$\hat{y}(k) = \sum_{j=1}^{m} w_j(k) z_j(k)$$

  - Step2. Backward pass computations

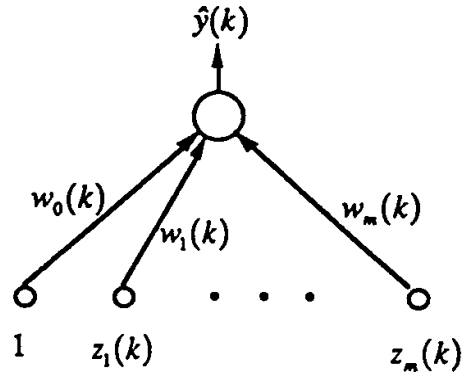
$$\delta(k) = \hat{y}(k) - y(k)$$

$$\mathbf{w}_j(k+1) = \mathbf{w}_j(k) - \gamma_k \delta(k) z_j(k)$$

Linear Parameter Estimation (Cont'd)

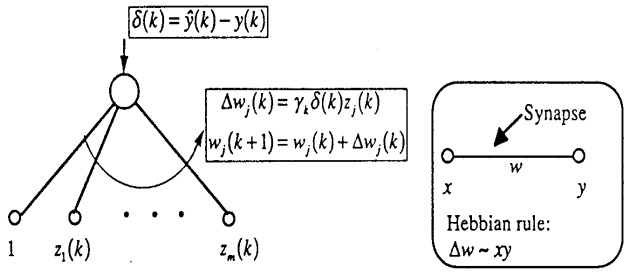
- Delta rule effectively implements LMS (least-mean-squares) minimization, updating parameters with every training sample.
- Change in connection strength is proportional to the error and to activation of the input unit.

Linear Parameter Estimation (Cont'd)



(a) Forward pass

Linear Parameter Estimation (Cont'd)



(b) Backward pass

Figure 5.1 Neural network interpretation of the delta rule.

5.1.2 Backpropagation Training of MLP Networks

- Consider a learning machine implementing ERM inductive principle
  - L_2 loss function
  - Approximating functions

$$f(\mathbf{x}, \mathbf{w}, \mathbf{V}) = w_0 + \sum_{j=1}^{m} w_j g(v_{0j} + \sum_{i=1}^{d} x_i v_{ij})$$

  - multilayer perceptrons
  - g is an *activation function* which is a differentiable monotonically increasing function.
  - nonlinear in the parameters $\mathbf{V}$

Backpropagation Training of MLP Networks (Cont'd)

- Risk functional

$$R_{emp} = \sum_{i=1}^{n} (f(\mathbf{x}_i, \mathbf{w}, \mathbf{V}) - y_i)^2$$

- Parameter updating rule

$$\mathbf{V}(k+1) = \mathbf{V}(k) - \gamma_k grad_{\mathbf{V}} L(\mathbf{x}(k), y(k), \mathbf{V}(k), \mathbf{w}(k))$$

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \gamma_k grad_{\mathbf{w}} L(\mathbf{x}(k), y(k), \mathbf{V}(k), \mathbf{w}(k))$$

$$L(\mathbf{x}(k), y(k), \mathbf{V}(k), \mathbf{w}(k)) = \frac{1}{2}(f(\mathbf{x}, \mathbf{w}, \mathbf{V}) - y)^2$$

Backpropagation Training of MLP Networks (Cont'd)

- Gradient calculation
  - Decomposition of the approximating function

$$a_j = \sum_{i=0}^{d} x_i v_{ij},$$

$$z_j = g(a_j),$$

$$z_0 = 1$$

$$\hat{y} = \sum_{j=0}^{m} w_j z_j$$

$$j = 1, \ldots, m$$

Backpropagation Training of MLP Networks (Cont'd)

- Relevant gradients

$$\frac{\partial L}{\partial v_{ij}} = \frac{\partial L}{\partial y} \frac{\partial \hat{y}}{\partial a_j} \frac{\partial a_j}{\partial v_{ij}} = (\hat{y} - y)(g'(a_j) w_j) x_i$$

$$\frac{\partial L}{\partial w_j} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial w_j} = (\hat{y} - y) z_j$$

Backpropagation Training of MLP Networks (Cont'd)

- Updating Procedure
  - Step 1. Forward pass computations
    - "Hidden layer"

$$a_j(k) = \sum_{i=0}^{d} x_i(k) v_{ij}(k), \qquad j = 1, \ldots, m$$

$$z_j(k) = g(a_j(k)), \qquad j = 1, \ldots, m$$

$$z_0(k) = 1$$

    - "Output layer"

$$\hat{y}(k) = \sum_{j=0}^{m} w_j(k) z_j(k)$$

Backpropagation Training of MLP Networks (Cont'd)

- Step 2. Backward pass computations
  - "Output layer"

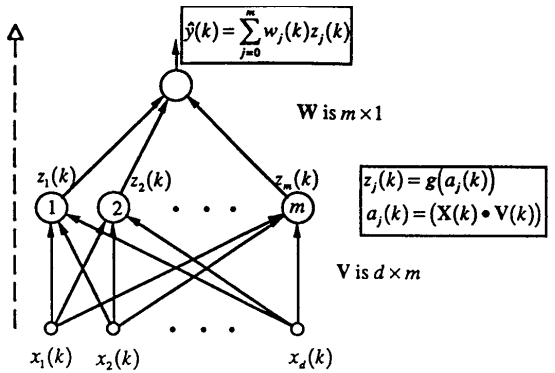
$$\delta_0(k) = \hat{y}(k) - y(k)$$
$$w_j(k+1) = w_j(k) - \gamma_k \delta_0(k) z_j(k), \qquad j = 0, \ldots, m$$

  - "Hidden layer"

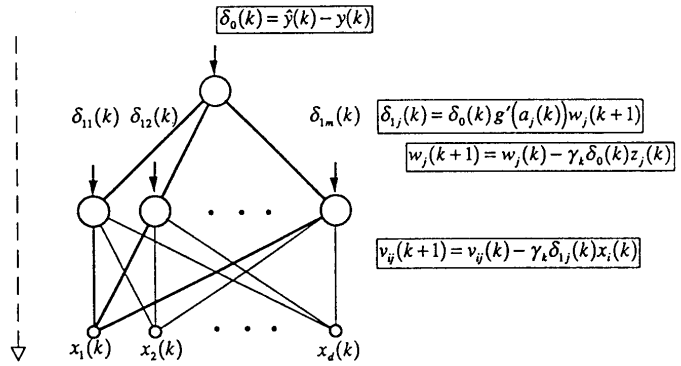
$$\delta_{ij}(k) = \delta_0(k) g'(a_j(k)) w_j(k+1)$$
$$v_{ij}(k+1) = v_{ij}(k) - \gamma_k \delta_{ij}(k) x_i(k), \quad i = 0, \ldots, d, \quad j = 0, \ldots m$$

Backpropagation Training of MLP Networks (Cont'd)

- When the number of training samples is finite, asymptotic conditions of stochastic approximation are satisfied by *recycling*.
- Other loss functions can be used as long as partial derivatives of the risk functional can be calculated via the chain rule.



(a) Forward pass



(b) Backward pass

Figure 5.2 Backpropagation training.

5.2 Iterative Methods

- Parameters are estimated iteratively.
- At each iteration the value of empirical risk is decreased.
- Not use the gradient estimates.
- Rely on a particular form of approximation functions and/or the loss function.
- Two steps (expectation, maximization) are iterated until some convergence criterion is met.

5.2.1 Expectation-Maximization (EM) Methods for Density Estimation

- Commonly used to estimate parameters of a mixture model via maximum likelihood.
- Example
  - $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ are generated independently from some unknown density.
  - approximating functions for density estimation

$$f(\mathbf{x}, \mathbf{v}, \mathbf{w}) = \sum_{j=1}^{m} w_j g_j(\mathbf{x}, v_j)$$

v_j : parameters of the individual densities

w_j : mixing weights

EM Methods (Cont'd)

- Mixture density

$$p(\mathbf{x}) = \sum_{j=1}^{m} p(\mathbf{x} \mid j, v_j) P(j), \qquad \sum_{j=1}^{m} P(j) = 1$$

- Log likelihood function for the density $p(\mathbf{X})$

$$P(X \mid \mathbf{v}) = \sum_{i=1}^{n} \ln \sum_{j=1}^{m} P(j) p(\mathbf{x}_i \mid j, v_j)$$

EM Methods (Cont'd)

- If it is known which component generated a given data point,

$$P_c(X, Z \mid \mathbf{v}) = \sum_{i=1}^{n} \sum_{j=1}^{m} z_{ij} \ln p(\mathbf{x}_i \mid \mathbf{z}_i, v_j) P(\mathbf{z}_i)$$

  - $P_c(X,Y,\mathbf{v})$: log likelihood for the "complete" data
  - z_{ij} : indicator variable to indicate whether sample *i* originate from component density *j*.
  - Hidden variable z_{ij} is unavailable (since the available data are "incomplete".)
  - Expectations with respect to Z is maximized.

EM Algorithm (1/2)

- Initialization of $\mathbf{v}(0)$ and $\mathbf{w}(0)$
- Repeat until stopping criterion
  - $k=k+1$
  - E-Step
    - Compute expectation of complete data log likelihood

$$Q(\mathbf{x}, \mathbf{v}(k)) = \sum_{i=1}^{n}\sum_{j=1}^{m} \pi_{ij}[\ln g_j(\mathbf{x}_i, v_j(k)) + \ln w_j(k)]$$

$$\pi_{ij} = E[z_{ij} \mid \mathbf{x}_i] = \frac{w_j(k) g_j(\mathbf{x}_i, v_j(k))}{\sum_{l=1}^{m} w_l(k) g_l(\mathbf{x}_i, v_l(k))}$$

probability that component density j generated data point i

EM Algorithm (2/2)

- M-Step
  - Find the parameters $\mathbf{w}(k+1)$ and $\mathbf{v}(k+1)$ that maximize the expected complete data log likelihood

$$w_j(k+1) = \frac{1}{n}\sum_{i=1}^{n} \pi_{ij}$$

$$v_j(k+1) = \arg\max_{v_j} \sum_{i=1}^{n} \pi_{ij} \ln g_j(\mathbf{x}_i, v_j(k))$$

Example: Gaussian Mixture

- A set of approximating functions in the form of gaussian mixture.
- Each gaussian component has a covariance

$$\Sigma_j = \sigma^2 \mathbf{I}$$

- approximating density function is

$$f(\mathbf{x}) = \sum_{j=1}^{m} w_j g_j(\mathbf{x}, v_j) = \sum_{j=1}^{m} w_j \frac{1}{(2\pi\sigma_j^2)^{d/2}} \exp\left\{\frac{-\|\mathbf{x}-\mu_j\|^2}{2\sigma^2}\right\}$$

$$\text{with} \quad v_j = (\mu_j, \sigma)$$

Example: Gaussian Mixture (Cont'd)

- In E-Step, compute $\pi_{ij}=\mathrm{E}[z_{ij}|\mathbf{x}_i, \mathbf{v}(k)]$ as

$$\pi_{ij} = \frac{w_j \sigma_j^{-d}(k) \exp\{-\|\mathbf{x}_i - \mu_j(k)\|^2 / 2\sigma_j^2(k)\}}{\sum_{l=1}^{m} w_l \sigma_l^{-d}(k) \exp\{-\|\mathbf{x}_i - \mu_l(k)\|^2 / 2\sigma_l^2(k)\}}$$

- In M-Step, new mixing weights are estimated as well as the means and variances of the gaussians

$$w_j(k+1) = \frac{1}{n}\sum_{i=1}^{n} \pi_{ij} \qquad \mu_j(k+1) = \frac{\sum_{i=1}^{n} \pi_{ij} \mathbf{x}_i}{\sum_{i=1}^{n} \pi_{ij}}$$

$$\sigma_j(k+1) = \sqrt{\frac{\sum_{i=1}^{n} \pi_{ij} \|\mathbf{x}_i - \mu_j(k+1)\|^2}{d\sum_{i=1}^{n} \pi_{ij}}}$$

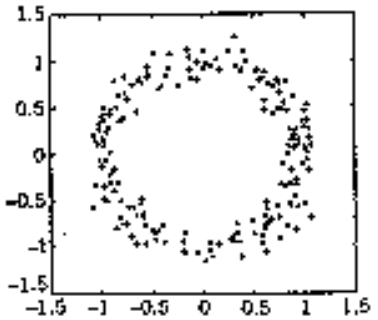
Example 5.1

- Density estimation problem
- Data points are generated according to

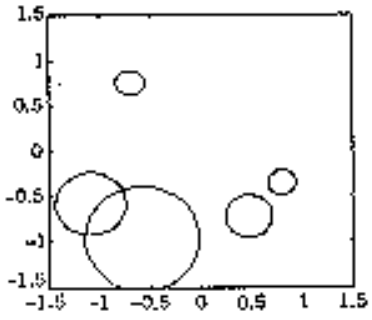
$$\mathbf{x} = [\cos(2\pi z), \sin(2\pi z)] + \varepsilon$$

Z : *uniformly distributed*

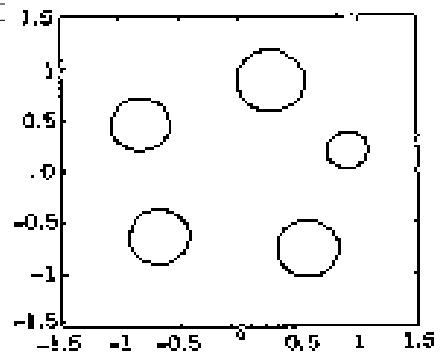
ε : *bivariate gaussian distributed* $\sum = \sigma^2 \mathbf{I},\ \sigma = 0.01$



200 data points



Initial configuration



After 20 iterations

5.2.2 Generalized Inverse Training of MLP Networks

- Given an MLP network implementing the ERM inductive principle

$$f(\mathbf{x}, \mathbf{w}, \mathbf{V}) = \sum_{i=1}^{m} w_i s(\mathbf{x} \cdot \mathbf{v}_i) + w_0$$

$$s(t) = \frac{1}{1 + \exp(-t)}$$

or

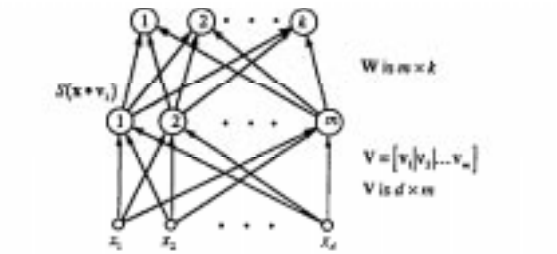
$$= \tanh(t) = \frac{\exp(t) - \exp(-t)}{\exp(t) + \exp(-t)}$$

Generalized Inverse Training of MLP Networks (Cont'd)

- Decomposition of three successive mappings
  - Linear mapping $\mathbf{xV}$
    - $\mathbf{V}=[\mathbf{v}_1|\mathbf{v}_2|\ldots|\mathbf{v}_m]$: $d\times m$ matrix
    - perform linear projection from d-dimensional input space to m-dimensional space
  - Nonlinear mapping $s(\mathbf{xV})$
    - result is an m-dimensional vector of the m hidden layer unit outputs
  - Linear mapping $s(\mathbf{xV})\bullet\mathbf{w}$
    - at output layer

Generalized Inverse Training of MLP Networks (Cont'd)

- Multiple-output MLP



$$f(\mathbf{x},\mathbf{W},\mathbf{V})=\sum_{i=1}^{m} w_i s(\mathbf{x}\bullet\mathbf{v}_i)+w_0 \quad \text{for each output unit}$$

In matrix notation: $F(\mathbf{x},\mathbf{W},\mathbf{V})=s[(\mathbf{x}\bullet\mathbf{V})]\bullet\mathbf{W}$

Figure 5.4 A multilayer perceptron network presented in matrix notation.

Generalized Inverse Training of MLP Networks (Cont'd)

- Mapping in matrix notation

$$F(\mathbf{x},\mathbf{W},\mathbf{V})=s(\mathbf{xV})\mathbf{W}$$

- Empirical Risk

$$R_{emp}=\frac{1}{n}\| s(\mathbf{X}_t\mathbf{V})\mathbf{W}-\mathbf{Y}_t\|^2$$

$$=\frac{1}{n}\sum_{i=1}^{n}\| s(\mathbf{x}_i\mathbf{V})\mathbf{W}-\mathbf{y}_i\|^2$$

$[\mathbf{X}_t|\mathbf{Y}_t]$: n * (d+0) matrix of training samples

Generalized Inverse Training of MLP Networks (Cont'd)

- Optimization Procedure
  - If an estimate of $\mathbf{V}$ is available

$$R_{emp}(\mathbf{W})=\frac{1}{n}\| s(\mathbf{X}_t\hat{\mathbf{V}})\mathbf{W}-\mathbf{Y}_t\|^2$$

Generalized Inverse Training of MLP Networks (Cont'd)

– An optimal estimate of **W** is found as

$$\mathbf{B} = s(\mathbf{X}_t \hat{\mathbf{V}})$$

$$\hat{\mathbf{W}} = \mathbf{B}^{+}\mathbf{Y}_t$$

$\mathbf{B}^{+}$: generalized inverse of $n \times m$ matrix $\mathbf{B}$,

$$\mathbf{B}^{+}\mathbf{B} = \mathbf{I}_m$$

$\hat{\mathbf{W}}$: is unique since typically n > m.

(#examples > #hiddens)

Generalized Inverse Training of MLP Networks (Cont'd)

– If an estimate of matrix **W** is available, the outputs of the hidden layer **B** can be estimated via linear least squares minimization of

$$R_{emp}(\mathbf{B}) = \| \mathbf{B}\hat{\mathbf{W}} - \mathbf{Y}_t \|^2$$

$$\hat{\mathbf{B}} = \mathbf{Y}_t \hat{\mathbf{W}}^{+}$$

$\hat{\mathbf{W}}^{+}$: right generalized inverse of matrix $\hat{\mathbf{W}}$ so that

$$\hat{\mathbf{W}}\hat{\mathbf{W}}^{+} = \mathbf{I}_m$$

$\hat{\mathbf{B}}$ is unique only if m ⩽ 0.

(#hiddens ⩽ #outputs)

– If m > 0, poor solutions will be produced.

Generalized Inverse Training of MLP Networks (Cont'd)

– by minimizing the following, one can find input layer weights

$$\| \mathbf{X}_t \mathbf{V} - s^{-1}(\hat{\mathbf{B}}) \|^2$$

$$\hat{\mathbf{V}} = \mathbf{X}_t^{+} s^{-1}(\hat{\mathbf{B}})$$

$\mathbf{X}_t^{+}$: left generalized inverse of matrix $\mathbf{X}_t$

Generalized Inverse Training of MLP Networks (Cont'd)

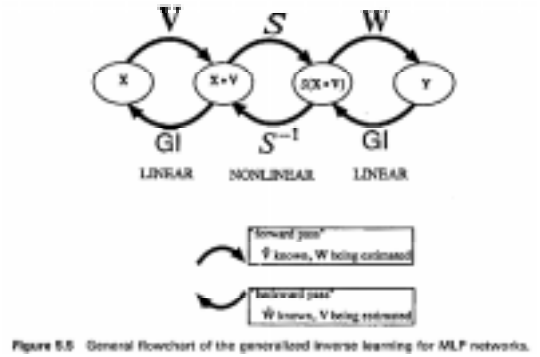


Figure 5.5 General flowchart of the generalized inverse learning for MLP networks.

The main advantage of GIL is computational speed, compared to traditional backpropagation training.

Generalized Inverse Training of MLP Networks (Cont'd)

Initialize $\hat{\mathbf{V}}$ to small (random) values
Set iteration step $k = 0$
Iterate: $k = k + 1$
'forward pass'
$\mathbf{B}(k) = s(\mathbf{X}_t\hat{\mathbf{V}}(k))$
$\hat{\mathbf{W}}(k) = \mathbf{B}^{+}(k)\mathbf{Y}_t$
compute empirical risk $R_{emp}(\hat{\mathbf{W}}(k))$ of the model
if ($R_{emp}(\hat{\mathbf{W}}(k))$ < preset value) then STOP **else** CONTINUE

'backward pass'
$\hat{\mathbf{B}}(k) = \mathbf{Y}_t\hat{\mathbf{W}}^{+}(k)$
$\hat{\mathbf{V}}(k) = \mathbf{X}_t^{+}s^{-1}(\hat{\mathbf{B}}(k))$
if (number of iterations k < preset limit) **then** go to *iterate* **else** STOP

5.3 Greedy Optimization

- Used when the set of approximating functions is a linear combination of the basis functions.
- Can be applied to density estimation, regression, or classification.
- Update parameters incrementally.
- *Backfitting*, *pruning* are possible.
- In neural network, "*network growing*" algorithms or "*constructive*" procedure.
- Enable fast learning, but optimization can be suboptimal.

5.3.1 Neural Network Construction Algorithms

- Use greedy strategy to adjust the number of hidden units.
- Reduce training time by making incremental changes to the network configuration and reusing past parameter values.
- Sequential Network Construction (SNC, Moody 1994)

Sequential Network Construction (SNC)

- Its description is given for networks with a single output for regression formulation.
- Grow network by adding m_2 hidden units at a time.
- Utilize the weights of a smaller network for training the larger network.

SNC (Cont'd)

- Optimization algorithm

$$f_k(\mathbf{x}, \mathbf{w}(k), \mathbf{V}(k)) = w_0 + \sum_{j}^{m_1+km_2} w_j g\left(v_{0j} + \sum_{i=1}^{d} x_i v_{ij}\right)$$

  - Initialization (k=0)
    - for $f_0(\mathbf{x})$, apply gradient descent steps with $\mathbf{w}(0)$, $\mathbf{V}(0)$

SNC (Cont'd)

  - For iterations k=1, 2,…

$$\begin{aligned}
&w_j(k) = w_j(k-1) && \text{for } j = 0, 1, 2, \ldots, m_1 + (k-1)m_2 \\
&w_j(k) = \varepsilon && \text{for } j = (1 + m_1 + (k-1)m_2), \ldots, (m_1 + km_2) \\
&v_{ij}(k) = v_{ij}(k-1) && \text{for } i = 0,1,\ldots,d \quad j = 0,1,\ldots, m_1 + (k-1)m_2 \\
&v_{ij}(k) = \varepsilon && \text{for } i = 0,1,\ldots,d \\
& && j = (1 + m_1 + (k-1)m_2), \ldots, (m_1 + km_2)
\end{aligned}$$

    - apply backpropagation algorithm only to parameters initialized with random values
    - apply backpropagation algorithm to all parameters $\mathbf{w}(k)$ and $\mathbf{V}(k)$

5.3.2 Classification and Regression Trees (CART)

- Breiman et al. (1984)
- Version for regression

$$f(\mathbf{x}) = \sum_{j=1}^{m} w_j I(\mathbf{x} \in \mathbf{R}_j), \quad : \text{approximating functions are piecewise constant functions}$$

$$\mathbf{R}_j : \text{hyper-rectangular region in the input space}$$

$$I(\mathbf{x} \in \mathbf{R}_j) = \prod_{l=1}^{d} I(a_{jl} \le x_l \le b_{jl})$$

  - constant estimate w_j for all values $\mathbf{x}$ in region $\mathbf{R}_j$
  - If regions $\mathbf{R}_j$ are known, the best estimate for w_j is $w_j = \frac{1}{n_j} \sum_{\mathbf{x}_i \in \mathbf{R}_j} y_i$

CART (Cont'd)

- Recursive partitioning
  - Initial region $\mathbf{R}_0$ of entire input space

$$\begin{aligned}
&\textit{if } \mathbf{x} \in \mathbf{R}_0 \quad \textit{then} \\
&\quad \textit{if } x_k \le v \quad \textit{then} \quad \mathbf{x} \in \mathbf{R}_1 \\
&\quad \textit{else} \quad \mathbf{x} \in \mathbf{R}_2
\end{aligned}$$

  - For given k and v, the optimum parameter values for w_1 and w_2 are the means of the samples falling into the regions
  - Iterate recursively until a relatively large number of regions are created
  - Regions are recombined through unions with adjacent regions

CART (Cont'd)

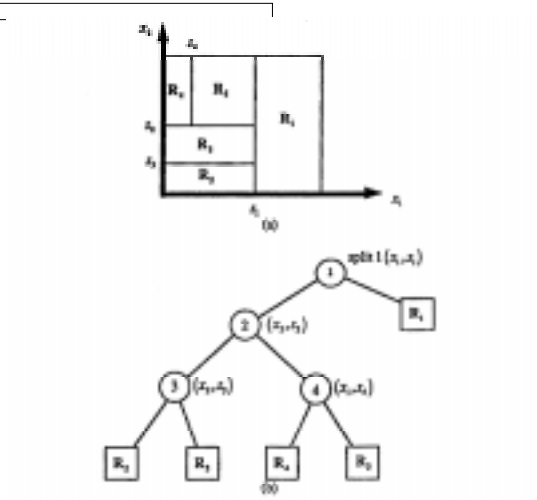
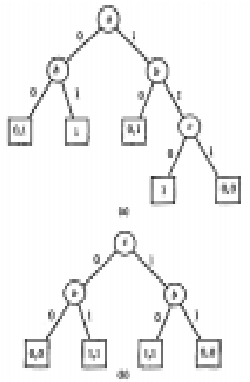


Figure 5.6 An example of CART partitioning for a function of two variables: (a) Partitioning in x space; (b) the resulting tree.

CART (Cont'd)

- Counterexample : $y=f(a, b, c)$



y	a	b	c
0	0	0	0
0	1	0	0
1	0	0	1
1	1	0	1
1	0	1	0
1	1	1	0
0	1	1	1
0	1	1	1

Figure 5.7 Counterexample for CART: (a) Suboptimal tree produced by CART; (b) optimal binary tree.

5.4 Feature Selection, Optimization, and Statistical Learning Theory

- Nonlinear optimization can be interpreted as an automatic feature selection.
- Most practical methods (MLP, CART,MARS etc) have a parameterization called *dictionary representation*:

$$f(\mathbf{x}, \mathbf{W}, \mathbf{v}) = \sum_{i=1}^{m} w_i g(\mathbf{x}, \mathbf{v}_i) + w_0$$

- Difference in terms of model selection
  - Implementations of stochastic approximation and iterative optimization strategy
    - minimize risk functional for a fixed m.
    - task of optimization is decoupled from model selection.
  - Implementations of greedy optimization strategy
    - include model selection as a part of optimization procedure.

- Distinction between linear (nonadaptive) and nonlinear (adaptive) learning methods

$$f(\mathbf{x}, \mathbf{w}, \mathbf{v}) = \sum_{i=1}^{m} w_i g(\mathbf{x}, \mathbf{v}_i) + w_0$$

  - linear methods
    - features or basis functions of f are prespecified.
    - learning amounts to estimating w_i via minimization of empirical risk.
    - basis functions themselves can be nonlinear in input variables.
  - nonlinear methods
    - basis functions or features are selected from a large (infinite) set of possible basis functions.
    - feature selection is performed via nonlinear optimization of the empirical risk.

- Sparse feature selection

$$P_m(x, \mathbf{w}) = \sum_{1}^{m} w_i x^{k_i}$$

  - univariate regression using sparse algebraic polynomials (Vapnik, 1995)
  - example of nonlinear methods using parameterization with basis functions linear in parameters:

$$f(\mathbf{x}, \mathbf{w}, \mathbf{v}) = \sum_{i=1}^{m} w_i g(\mathbf{x}, \mathbf{v}_i) + w_0$$

  - estimating the best sparse polynomial for a given data set is equivalent to selecting m features from an infinite set of features $1, x, x^2, x^3, \ldots$

- Adaptive methods perform feature selection using training data via nonlinear optimization of risk functional.
- Nonadaptive methods use prespecified set of features independent of training data.
- In both types of methods, selecting the right number of features m (model selection) is done using training data.

- Relation to SRM
  - Dictionary representation can be viewed as a *structure* in the sense of SRM.
  - All adaptive methods can be related to the SRM inductive principle.
  - Adaptive methods seek to minimize an empirical risk on an element of a structure for a fixed m.
- Problems in application of SRM formalism
  - Estimating VC-dimension of a set of functions of dictionary representation with nonlinear basis functions is very difficult.
  - Minimization of empirical risk leads to nonlinear optimization for which only a local minimum can usually be found.

- Questions related to SLT interpretation
  - Can one specify a structure for which the SRM principle can be rigorously applied?
    - global minimization of the empirical risk is possible?
    - VC-dimension can be estimated?
  - How can different nonlinear optimization strategies be interpreted using SRM?
- Correspondence between SRM and adaptive methods
  - gradient-descent and iterative methods
    - provide a priori specification of a structure in the form of dictionary representation as following

$$f(\mathbf{x}, \mathbf{w}, \mathbf{v}) = \sum_{i=1}^{m} w_i g(\mathbf{x}, \mathbf{v}_i) + w_0$$

    - elements of a structure is indexed by m.

  - greedy optimization
    - no clear correspondence with SRM
    - two ways to relate to SRM
      - view greedy methods as a strictly computational procedure for optimization
        - first specifies an element of a structure, that is a fixed number of basis functions
        - amounts to selecting an optimal set of basis functions (features) minimizing empirical risk
        - selects basis functions one at a time, and this does not result in a thorough optimization over all basis functions
        - final model depends on the first decision, and is very sensitive to small changes in the training samples
        - produce unstable models that are not robust with respect to small variations in the training data and tuning parameters

  - View greedy procedures as an implementation of a statistical strategy for fitting data
    - training data is decomposed into structure(model fit) and noise (residual)

> (1) DATA = (model)FIT 1 + RESIDUAL 1
>
> (2) RESIDUAL 1 = FIT 2 + RESIDUAL 2
>
> and so on. The final model for the data would be
>
> MODEL = FIT 1 + FIT 2 + · · · .

    - rooted in a statistical strategy of partitioning variability into two distinct parts : explained and unexplained
    - results in minimizing residual error, hence has superficial similarity to minimization of empirical risk via SRM
    - approximating functions are added as dictated by the data, unlike SRM
    - useful for data fitting and data analysis

- no theory and little evidence to suggest usefulness as an inductive principle for predictive learning
- some greedy strategies are used for predictive learning

5.5 Summary

- Problems of nonlinear optimization approaches
  - Sensitivity to initial conditions
  - Sensitivity to stopping rules
  - Multiple local minima
    - simulated annealing to escape from local minima
    - nonlinear parameter estimation from many randomly chosen initializations
- There is no single best method for all problems

Homework

- EM Algorithm의 구현
- Example 5.1 (p. 143)에 대한 실험 결과 제출