함수 근사를 위한 점증적 서포트 벡터 학습 방법

임 채 환, 박 주 영
고려대학교 제어계측공학과
전화 : 02-922-0863 / 핸드폰 : 011-9706-1354

Incremental Support Vector Learning Method for Function Approximation

Chae-Hwan Leem, Jooyoung Park
Dept. Control and Instrumentation Engineering, Korea University
E-mail : mirage@cie01.korea.ac.kr

Abstract

This paper addresses incremental learning method for regression. SVM(support vector machine) is a recently proposed learning method. In general training a support vector machine requires solving a QP (quadratic programing) problem. For very large dataset or incremental dataset, solving QP problems may be inconvenient. So this paper presents an incremental support vector learning method for function approximation problems.

I. 서 론

서포트 벡터 학습 방법(support vector machine)은 최근에 패턴 분류 및 비선형 함수 근사 등의 문제에서 매우 우수한 성능을 보여주고 있는 학습방법이다[1,2].

이 학습 방법은 커널함수를 사용함으로써, 비선형 문제를 원형 문제에 커다란 변화 없이 쉽게 풀이할 수 있는 장점을 갖는다. 그러나, 이러한 서포트 벡터 학습 방법은 최종적으로 QP 문제(quadratic programing problem)형태로 풀이되어지는데, 많은 양의 학습 데이터가 주어지게 되는 경우에는 오랜 계산 시간을 갖는 문제점을 가질 수 있다. 게다가 한번에 데이터가 주어지는 문제의 형태가 아닌, 데이터가 하나씩 주어지는 형태의 문제를 풀려면 매번 모든 데이터를 이용하여 최적화 문제를 풀어야 하는 부담이 생기게 된다. 따라서, 기존의 데이터를 사용하여 얻어진 최적해와 새롭게 추가된 데이터만을 이용하여 다음의 최적해를 얻어가는 점증적 서포트벡터 학습방법(incremental support vector learning)이 필요하게 된다.

본 논문에서는 비선형 함수 근사에 사용되어지는 SVR(support vector regression)에 대하여, 점증적인 방법의 알고리즘을 제시하고자 한다.

II. 점증적 서포트 벡터 근사 학습방법

2.1 서포트 벡터 함수 근사(SVR)

다음과 같이 m개의 주어진 데이터 $\{(x_i, y_i)\}_{i=1}^{m}$에 대해서 SVR문제는 다음의 최적화 문제를 푸는 것이 된다[2, 3]:

$$\begin{aligned} \min\ & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}(\xi_i + \xi_i^*) \\ s.t.\ & y_i - (\langle w, \phi(x_i)\rangle + b) \le \varepsilon + \xi_i \\ & (\langle w, \phi(x_i)\rangle + b) - y_i \le \varepsilon + \xi_i^* \\ & \xi_i^{(*)} \ge 0\ ,\ \forall i. \end{aligned} \tag{1}$$

여기서, w, b는 예측함수 $f(x) = (w \cdot \phi(x)) + b$의 값이다. 이러한 제약조건이 있는 최적화 문제를 풀기 위해 라그랑제 승수를 도입하여 라그랑제 함수를 다음과 같이 정의한다:

$$L = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}(\xi_i+\xi_i^*) - \sum_{i=1}^{m}\eta_i\xi_i - \sum_{i=1}^{m}\eta_i^*\xi_i^* + \sum_{i=1}^{m}\alpha_i[y_i - \langle w, \phi(x_i)\rangle - b - \varepsilon - \xi_i] + \sum_{i=1}^{m}\alpha_i^*[\langle w, \phi(x_i)\rangle + b - y_i - \varepsilon - \xi_i^*]. \quad (2)$$

여기서, $\alpha_i, \alpha_i^*, \eta_i, \eta_i^* \geq 0$가 라그랑제 승수에 해당한다. 이 라그랑제 함수는 쌍대 변수의 도입으로 풀이가 되는데 이때 구해지는 쌍대문제는 다음과 같다:

$$\min\ W = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)K(x_i,x_j) + \varepsilon\sum_{i=1}^{m}(\alpha_i+\alpha_i^*) - \sum_{i=1}^{m}(\alpha_i-\alpha_i^*)y_i + b\sum_{i=1}^{m}(\alpha_i-\alpha_i^*) \quad (3)$$

$$s.t.\ \ 0 \leq \alpha_i^{(*)} \leq C\ ,\quad \sum_{i=1}^{m}(\alpha_i-\alpha_i^*) = 0\ ,\ \forall i.$$

여기서, $K(x_i, x_j)$는 커널함수를 의미하고, Mercer의 조건에 따른 함수를 사용한다. 커널 함수를 사용함으로써 선형 문제뿐만 아니라 비선형 문제도 풀이할 수 있게 된다[2]. 그리고, $w = \sum_{i=1}^{m}(\alpha_i-\alpha_i^*)\phi(x_i)$가 되므로 예측 함수는 $f(x) = \sum_{i=1}^{m}(\alpha_i-\alpha_i^*)K(x_i,x) + b$로 바뀌게 된다. 이것으로부터 알 수 있듯이 처음 구하고자 하는 w, b의 값들은 $(\alpha_i-\alpha_i^*)$의 값으로부터 구할 수 있게 되었다. 그러므로 이 쌍대문제를 풀게 되면 예측 함수 $f(x)$를 구할 수 있게 된다.

2.2 점증적 SVR 학습 방법

본 논문에서는, [4]에서 SVM 문제를 점증적으로 푸는 알고리즘을 구하기 위해 사용한 전략을 수정함으로써, 점증적 SVR 학습 방법을 제시하기로 한다. 앞절에서 소개한 쌍대문제(3)를 KT조건[3]의 형태로 바꾸기 위해서 W를 α_i, α_i^*, b로 미분하면,

$$g_i = \frac{\partial W}{\partial \alpha_i} = \sum_{j=1}^{m}(\alpha_j-\alpha_j^*)Q_{ij} + b - y_i + \varepsilon = f(x_i) - y_i + \varepsilon \begin{cases} \geq 0; & \alpha_i = 0 \quad (R) \\ = 0; & 0 < \alpha_i < C \quad (S) \\ \leq 0; & \alpha_i = C \quad (E) \end{cases} \quad (4)$$

$$g_i^* = \frac{\partial W}{\partial \alpha_i^*} = y_i - \left(\sum_{j=1}^{m}(\alpha_j-\alpha_j^*)Q_{ij} + b\right) + \varepsilon = y_i - f(x_i) + \varepsilon \begin{cases} \geq 0; & \alpha_i^* = 0 \quad (R) \\ = 0; & 0 < \alpha_i^* < C \quad (S^*) \\ \leq 0; & \alpha_i^* = C \quad (E^*) \end{cases} \quad (5)$$

$$\frac{\partial W}{\partial b} = \sum_{i=1}^{m}(\alpha_i-\alpha_i^*) = 0 \quad (6)$$

를 구하게 된다[4]. 여기서, S와 S^*는 서포트 벡터의 집합을, E와 E^*는 에러 벡터의 집합을, 그리고, R은 일반 벡터의 집합을 의미한다. KT조건으로부터 $\alpha_i\alpha_i^* = 0$이 되므로, g_i, g_i^* 역시 둘 중 하나는 항상 0이 되어야 한다. 위의 식(4), (5)와 (6)은 α_i, α_i^*의 범위로부터 이러한 벡터 집합을 구분하게 되는 조건을 제공하는데 이는 KT조건과 일치한다.

새로운 데이터의 벡터가 추가되어도 기존의 다른 벡터(S, S^*, E, E^*, R에 있는 벡터)들은 다른 집합으로의 이동이 없다고 가정하고 위의 식으로부터 점증적인 방법을 도입하면,

$$\Delta g_i = Q_{ic}\Delta(\alpha_c-\alpha_c^*) + \sum_{j\in SS}Q_{ij}\Delta(\alpha_j-\alpha_j^*) + \Delta b \quad (7)$$

$$\Delta g_i^* = -Q_{ic}\Delta(\alpha_c-\alpha_c^*) - \sum_{j\in SS}Q_{ij}\Delta(\alpha_j-\alpha_j^*) - \Delta b \quad (8)$$

$$,\ \forall i \in D\cup\{c\}$$

$$0 = \Delta(\alpha_c-\alpha_c^*) + \sum_{j\in SS}\Delta(\alpha_j-\alpha_j^*) \quad (9)$$

로 표현되어지는데, 여기서, Q_{ij}는 커널 함수를 편의상 표현한 것이고, D는 기존의 벡터들이 존재하는 집합이고 $\{c\}$가 이번 스텝에서 추가되어진 데이터이다. 그리고, $SS = S\cup S^*$를 의미한다.

식(7)과 (9)가 $SS = \{s_1, \cdots, s_n\}$의 영역에서 움직일 경우를 생각해 보자. 그러면, $g_i \equiv 0$ 이 되므로,

$$Q\cdot\begin{bmatrix}\Delta b\\ \Delta(\alpha_{s_1}-\alpha_{s_1}^*)\\ \vdots\\ \Delta(\alpha_{s_n}-\alpha_{s_n}^*)\end{bmatrix} = -\begin{bmatrix}1\\ Q_{s_1c}\\ \vdots\\ Q_{s_nc}\end{bmatrix}\Delta(\alpha_c-\alpha_c^*) \quad (10)$$

을 얻는다. Q는 다음과 같이 정의되는 대칭행렬이다.

$$Q \triangleq \begin{bmatrix}0 & 1 & \cdots & 1\\ 1 & Q_{s_1s_1} & \cdots & Q_{s_1s_n}\\ \vdots & \vdots & \ddots & \vdots\\ 1 & Q_{s_ns_1} & \cdots & Q_{s_ns_n}\end{bmatrix} \quad (11)$$

이러한 결과는 식(8)과 (9)를 통해서도 같은 결과를 얻게 된다. 위의 식을 간략하게 표현하기 위하여,

$$\Delta b = \beta\Delta(\alpha_c-\alpha_c^*) \quad (12)$$

$$\Delta(\alpha_j-\alpha_j^*) = \beta_j\Delta(\alpha_c-\alpha_c^*),\ \forall j \in SS \quad (13)$$

를 정의하여 식(10)에 대입하여 정리하면 다음을 얻게 된다.

$$\begin{bmatrix}\beta\\ \beta_{s_1}\\ \vdots\\ \beta_{s_n}\end{bmatrix} = -R\cdot\begin{bmatrix}1\\ Q_{s_1c}\\ \vdots\\ Q_{s_nc}\end{bmatrix} \quad (14)$$

여기서, $R = Q^{-1}$을 의미하고, $\forall j \notin SS$에 대해서 β_j는 0이다. 식(12)와 (13)등을 사용하여 식(7), (8)을 표현하면,

$$\Delta g_i = \gamma_i \Delta(\alpha_c - \alpha_c^*), \quad \forall i \in D \cup \{c\} \tag{15}$$

$$\Delta g_i^* = -\gamma_i \Delta(\alpha_c - \alpha_c^*), \quad \forall i \in D \cup \{c\} \tag{16}$$

을 구하게 되고, γ_i는 다음을 의미한다:

$$\gamma_i = Q_{ic} + \sum_{j \in SS} Q_{ij}\beta_j + \beta, \quad \forall i \notin SS. \tag{17}$$

위의 식(15)와 (16)은 α_c혹은 α_c^*의 변화($\Delta(\alpha_c - \alpha_c^*)$)로부터 g_i혹은 g_i^*의 변화를 계산하는 식이 된다. 즉, 새롭게 추가된 $\{c\}$영역의 데이터에 의해서 기존의 데이터의 g_i(g_i^*)의 값의 변화를 얻게된다. 또한, 식(12)와(13)도 같은 경우이다. 그러므로, 새롭게 추가된 $\{c\}$영역의 데이터가 기존의 데이터의 영역에 영향을 미치게 된다. 이상의 내용을 요약 정리하면 다음과 같다:

[요약] $\Delta(\alpha_i - \alpha_i^*)$의 변화가 충분히 작아서, 이 변화에 따라 각 데이터의 영역(S, S^*, E, E^*, R)에서 이동하지 않는다고 가정하고, 최대로 증가되어질 α_c , α_c^*의 값을 찾는다.

1. $g_c \le 0$ 와 $g_c^* \ge 0$ 일때, $g_c = 0$ 이면 $c \in S$ 혹은,
   $g_c \ge 0$ 와 $g_c^* \le 0$ 일때, $g_c^* = 0$ 이면 $c \in S^*$
2. $\alpha_c \le C$ 일때, $\alpha_c = C$가 되면 $c \in E$ 혹은,
   $\alpha_c^* \le C$ 일때, $\alpha_c^* = C$가 되면 $c \in E^*$
3. $0 \le \alpha_i \le C$, $\forall i \in S$ 일때, $\begin{cases} \alpha_i = C\text{가 되면}, & i \in S \to E \\ \alpha_i = 0\text{이 되면}, & i \in S \to R \end{cases}$
4. $0 \le \alpha_i^* \le C$, $\forall i \in S^*$ 일때, $\begin{cases} \alpha_i^* = 0\text{이 되면} & i \in S^* \to R \\ \alpha_i^* = C\text{가 되면} & i \in S^* \to E^* \end{cases}$
5. $g_i \le 0$와 $g_i^* \ge 0$, $\forall i \in E$. 일때, $g_i = 0$이면 $i \in E \to S$
6. $g_i \ge 0$ 와 $g_i^* \le 0$, $\forall i \in E^*$. 일때, $g_i^* = 0$이면 $i \in E^* \to S^*$
7. $g_i \ge 0$와 $g_i^* \ge 0$, $\forall i \in R$. 일때 $\begin{cases} g_i = 0\text{ 이 되면 } i \in R \to S \\ g_i^* = 0\text{ 이 되면 } i \in R \to S^*. \end{cases}$

학습이 되어지면서 $\{c\}$의 벡터들 중 일부는 서포트 벡터가 된다. 서포트 벡터가 되면 식(14)의 R의 차원은 다음과 같이 증가하게 된다:

$$R \leftarrow \begin{bmatrix} & & 0 \\ & R & \vdots \\ & & 0 \\ 0 & \cdots\ 0 & 0 \end{bmatrix} + \frac{1}{\gamma_c} \begin{bmatrix} \beta \\ \beta_{s_1} \\ \vdots \\ \beta_{s_n} \\ 1 \end{bmatrix} [\beta\ \beta_{s_1} \cdots \beta_{s_n} 1] \tag{19}$$

이때, γ_c, β, β_j등은 식(14)와 (17)로부터 얻는다. 반대로, 일부는 서포트 벡터에서 다른 벡터로 이동하게 되는데 이때는,

$$R_{ij} \leftarrow R_{ij} - R_{kk}^{-1} R_{ik} R_{kj} \quad \forall i, j \in SS \cup \{0\};\ i, j \ne k \tag{18}$$

와 같은 방법으로 차원을 줄이게 된다.

2.3 학습 알고리즘

데이터의 개수가 m에서 $m+1$로 증가하는 경우 ($D^{m+1} = D^m \cup \{c\}$) 추가되는 데이터의 새로운 해 $\{(\alpha_i - \alpha_i^*)^{m+1}, b^{m+1}\}_{i=1}^{m+1}$는 기존 데이터의 해 $\{(\alpha_i - \alpha_i^*)^m, b^m\}$로 표현되어진다고 할 때 다음의 알고리즘을 제안하였다.

[알고리즘]

1. α_c 및 α_c^*를 0으로 초기화
2. $g_c > 0$ & $g_c^* > 0$인 경우 iteration 종료
3. $g_c \le 0$ & $g_c^* \ge 0$ 일때: 가능한 최대의 $\Delta\alpha_c$구함.
   ① $g_c = 0 : c \to S$, update R 그리고 종료.
   ② $\alpha_c = 0 : c \to E$, 그리고 종료.
   ③ D^m의 데이터중 [요약]의 조건에 따른 이동.
4. $g_c \ge 0$ & $g_c^* \le 0$ 일때: 가능한 최대의 $\Delta\alpha_c^*$구함.
   ① $g_c^* = 0 : c \to S^*$, update R 그리고 종료.
   ② $\alpha_c^* = 0 : c \to E^*$, 그리고 종료.
   ③ D^m의 데이터중 [요약]의 조건에 따른 이동.
   iteration 종료.

필요에 따라 반복.

이러한 알고리즘을 사용하여 점증적으로 주어지는 문제를 풀이할 수 있게 되었다.

III. 예 제

본 논문에서 제시한 알고리즘을 사용하여 함수 근사 문제를 풀이하여보도록 한다. 본 논문에서는 데이터의 개수가 50개인 sinc 함수 근사 문제를 사용하여 논문에서 제시한 알고리즘과 matlab에서 제공하는 QP 문제를 푸는 함수인 quadprog() 라는 함수를 사용하여 풀이하였다. 주어진 데이터와 그 결과를 그림 1에 나타내었는데, 여기서 각 데이터의 숫자는 입력되어지는 데이터의 순서를 의미한다. 그리고, ⊗는 서포트 벡터 ($0 < \alpha_i^{(*)} < C$)를 ⊠는 에러 벡터($\alpha_i^{(*)} = C$)를 의미한

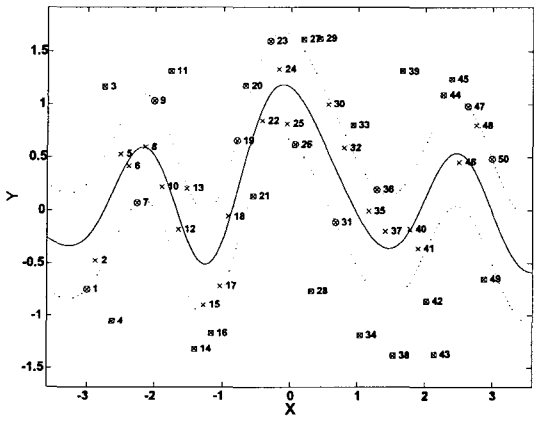

그림 1. 데이터 예제의 결과

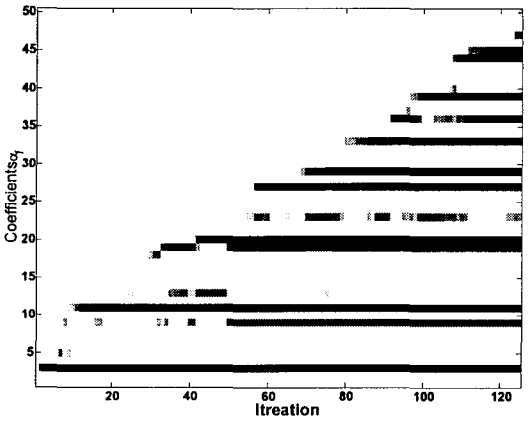

그림 4. 학습시간동안의 변수 α_i의 변화

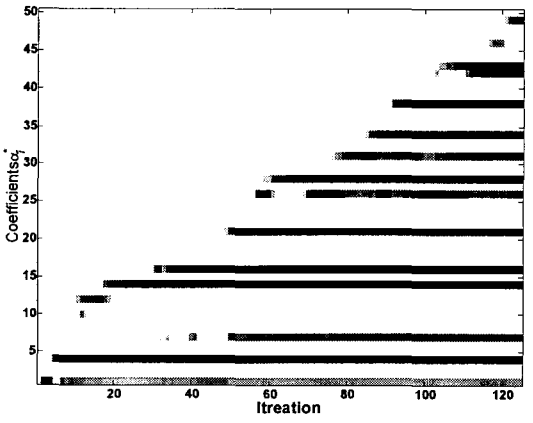

그림 5. 학습시간 동안의 변수 α_i^*의 변화

다. 예를 들어, 그림 1의 좌측 하단에 보면 "⊗1"이라고 표시되어 있는 것은 첫 번째 입력되어진 데이터이고, 결과가 서포트 벡터로 나타났다는 것을 의미한다. 그림 1의 결과는 논문에서 제시한 알고리즘을 사용한 것과 matlab의 quadprog()를 사용한 것이 같은 결과를 얻었다.

그림 2와 그림 3은 학습이 반복적으로 이루어짐에 따라 구하고자 하는 변수인 α_i혹은 α_i^*의 값의 변화를 나타낸 것이다. 색이 밝아질수록 0에 가까워지고, 검은색이 될수록 C에 가까워지게 된다.

IV. 결 론

본 논문에서는, 기존의 QP문제로 풀이되는 서포트 벡터의 학습 알고리즘을 좀더 빠른 연산속도와 점증적 데이터에 대해서도 연산이 가능하도록 하는 점증적 서포트 벡터 학습방법을 제안하였다. 제안된 알고리즘을 사용하여 예제에서 보여진 것과 같이 QP문제와 동일한 결과를 얻을 수 있었으며, 점증적인 특징을 갖게 되었다. 향후 연구과제로는 대규모의 실용적 데이터에 대한 적용과, 강화학습 등과 같이 함수 근사를 사용하는 분야에 대한 응용 등을 들수 있다.

* 감사의 글 : 본 연구는 고려대학교 특별연구비에 의하여 수행되었음.

참고 문헌

[1] V. N. Vapnik, The Nature of Statistical Learning Theory(Statistics for Engineering and Information Science), Springre-Verlag, 1999.

[2] A. J. Smola and B. Scholköpf, "A Tutorial on Support Vector Regression," NeuroCOLT Technical Report NC-TR-98-030, Royal Holloway College, University of London, UK, 1998.

[3] N. Cristianini and J. Shawe-Taylor, An Introduction to Support Vector Machines, Cambridge Press, 2000.

[4] G. Cauwenberghs and T. Poggio, "Incremental and Decremental Support Vector Machine Learning," Advanced Neural Information Processing Systems (NIPS 2000), MIT Press, Vol. 13, pp. 409-425, 2001.