Journal of the Korean Data & Information Science Society
2013, **24**(5), 989–998

http://dx.doi.org/10.7465/jkdi.2013.24.5.989
한국데이터정보과학회지

빅 데이터 분석을 위한 지지벡터기계[†]

최호식[1] · 박혜원[2] · 박창이[3]

[1]호서대학교 정보통계학과 · [23]서울시립대학교 통계학과

접수 2013년 6월 19일, 수정 2013년 7월 24일, 게재확정 2013년 8월 5일

요 약

최근 산/학계에서 주목받고 있는 빅 데이터는 정의상 한꺼번에 자료를 메모리에 올려 분석할 수 없기 때문에 기존의 데이터마이닝 시대에 개발된 일괄처리 (batch processing) 방식의 알고리즘을 적용할 수 없게 된다. 따라서 가장 시급히 해결해야 하는 문제는 기존의 여러 가지 기계학습방법을 빅 데이터에 적용할 수 있도록 분산처리 (distributed processing)를 수행하는 적절한 알고리즘을 개발하는 것이라 볼 수 있다. 본 논문에서는 분류문제에서 각광받는 지지벡터기계 (support vector machines)의 여러 알고리즘을 살펴보고자 한다. 특히 빅 데이터 분류문제에 유용할 것으로 예상되는 온라인 타입 알고리즘과 병렬처리 알고리즘에 대하여 소개하고, 이러한 알고리즘들의 성능 및 장단점을 선형분류에 대한 모의실험을 통해서 살펴본다.

주요용어: 분산처리, 온라인 알고리즘, 일괄처리, 합의.

1. 서론

최근 빅 데이터 (big data)에 대한 관심이 고조되고 있는데, 통계적 관점에서 빅 데이터와 관련된 주요 사안들로는 빅 데이터에 적용될 수 있는 분석 알고리즘의 개발과 사회망 (social network) 등의 새로운 형태의 자료의 분석 방법론의 개발 등을 들 수 있다. 특히 빅 데이터는 이름에서 내포하듯이 자료를 한번에 메모리에 올려 분석할 수 없기 때문에 기존의 데이터마이닝 시대에 개발된 일괄처리 (batch processing) 방식의 알고리즘을 적용할 수 없게 된다. 따라서 가장 시급히 해결해야 하는 문제는 기존의 여러 가지 기계학습방법을 빅 데이터에 적용할 수 있도록 분산처리 (distributed processing)를 수행하는 적절한 알고리즘을 개발하는 것이라 볼 수 있다.

빅 데이터 분석을 위한 주요 접근법은 훈련자료 (train data)의 일부를 표본 추출한 후 일괄처리 알고리즘을 적용하는 방법, 일괄처리 알고리즘을 병렬로 구현하는 방법, 그리고 훈련자료를 하나씩 (혹은 몇 개씩 묶음으로) 학습하는 온라인 알고리즘 (online algorithm) 등으로 구분할 수 있다. 이러한 여러 접근법들중 병렬처리 (parallel processing) 혹은 온라인 학습 알고리즘 등이 실질적으로 빅 데이터 분석에 의미있는 접근법으로 여겨진다. 최근 지지벡터기계 (support vector machines)는 비정상적인 신호의 탐지 (Park, 2011), 만족도 비교 (Pi 등, 2011), 기업 도산 예측 (Park 등, 2012) 등 다양한 분류문제에서 각광받고 있다. 본 논문에서는 지지벡터기계의 여러 가지 구현 알고리즘을 살펴보고, 빅 데이터

[†] 이 논문은 2012년도 정부 (교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No. 2012R1A1A2004901).
[1] (336-795) 충남 아산시 배방읍 호서로 79번길 20번지, 호서대학교 정보통계학과, 조교수.
[2] (130-743) 서울시 동대문구 전농동 90번지, 서울시립대학교 통계학과, 석사과정.
[3] 교신저자: (130-743) 서울시 동대문구 전농동 90번지, 서울시립대학교 통계학과, 부교수.
E-mail: park463@uos.ac.kr

분류문제에 유용할 것으로 예상되는 Shalev-Shwartz 등 (2011)의 온라인 타입 알고리즘인 PEGASOS (Primal Estimated sub-GrAdient SOlver for Support vector machines) 와 Forero 등 (2010)에서 제안된 병렬 알고리즘을 소개하고, 모의실험을 통하여 알고리즘들의 장단점을 살펴보고자 한다.

본격적인 논의에 앞서 병렬/분산 처리에 쓰이는 용어를 간단히 소개하자. 관측값들을 여러 집합으로 쪼개서 병렬처리하는 경우에는 합의 (consensus)라고 하며 입력변수들을 여러 집합으로 쪼개서 병렬처리하는 경우에는 공유 (sharing)라고 한다. Forero 등 (2010)에서 활용한 ADMM (alternating direction method of multipliers) 방법은 볼록최적화에 대하여 합의와 공유 방식의 병렬처리 알고리즘을 포괄하는데, Forero 등 (2010)은 합의 방식을 ADMM에 적용한 알고리즘을 개발하였다. 본 논문에서는 편의상 Forero 등 (2010)의 알고리즘을 ADMM으로 부르기로 한다. 일반적인 ADMM에 대한 자세한 사항은 Boyd 등 (2010)을 참고하기 바란다.

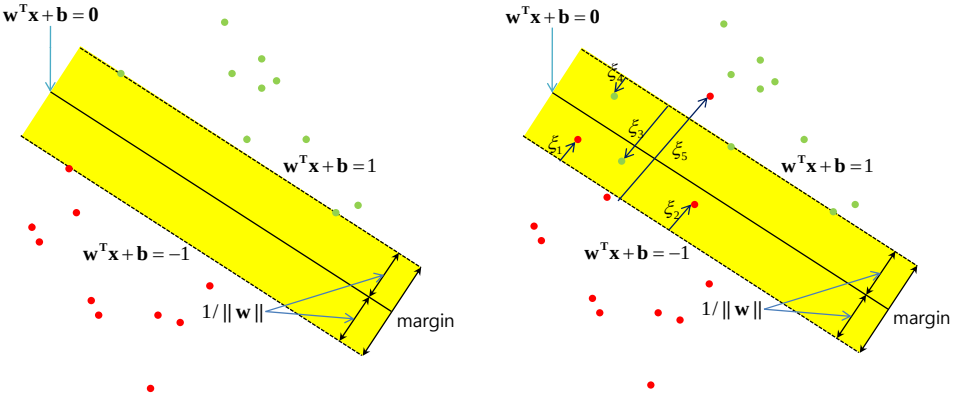
본 논문의 구성은 다음과 같다. 2절에서는 지지벡터기계와 여러 가지 구현 알고리즘을 살펴본다. 3절에서는 빅 데이터 분석에 적합한 온라인 타입의 PEGASOS와 병렬 알고리즘인 ADMM 을 소개하고, 4절에서는 선형분류에 대한 모의실험을 통하여 훈련시간과 분류 정확도 관점에서 두 알고리즘의 성능을 Fan 등 (2005)가 제안한 지지벡터기계의 대표적인 일괄처리 알고리즘의 하나인 LIB-SVM과 비교한다. 마지막으로 5절에서는 본 논문을 요약하고 향후 실제 문제에서의 적용 방안과 연구 방향 등 빅 데이터 분류의 여러 사안에 대하여 논의한다.

2. 지지벡터기계 알고리즘

2.1. 지지벡터기계

Cortes와 Vapnik (1995)에 의한 지지벡터기계는 여러 가지 분류문제에서 분류결과가 매우 정확하며 여러 형태의 자료에 대하여 적용이 가능한 매우 유연한 (flexible) 방법임이 입증되었다. 이 절에서는 입력공간과 출력공간이 각각 $\mathcal{X}\subset\mathbb{R}^p$과 $\mathcal{Y}=\{\pm 1\}$인 이진분류문제에서의 지지벡터기계에 대하여 소개한다.

선형 지지벡터기계의 분류함수 (classification function)를 $f(x) = \langle w, x\rangle + b$로 나타내고 훈련자료를 $\{(x_i, y_i)\}_{i=1}^n$라 하자. 훈련자료를 이용하여 구한 추정치를 $\hat{w}$와 $\hat{b}$로 나타내면, 추정된 분류함수는 $\hat{f}(x) = \langle \hat{w}, x\rangle + \hat{b}$로 나타낼 수 있으며 새로운 입력변수의 값 x에 대하여 $\text{sign}(\hat{f}(x))$로 예측할 수 있게 된다. 모든 훈련자료점이 $f(x) = 0$에 의해 정의되는 선형 초평면 (linear hyperplane)에 의하여 완벽히 분리되는 경우, 즉, 모든 $i = 1, \ldots, n$에 대하여 $y_i f(x_i) > 0$을 만족하는 f가 존재하는 경우를 분리가능한 경우 (separable case)라고 하며, 그렇지 않은 경우는 분리불가능한 경우 (nonseparable case)라고 한다. 각 경우들은 Figure 2.1에 도시되어 있다.



(a) Separable case (b) nonseparable case

Figure 2.1 Illustration of the linear support vector machine.

우선 선형적으로 분리가능한 경우에 대하여 살펴보자. 이 경우에는 모든 훈련자료점을 완벽히 분리하는 선형 분류함수가 무수히 많이 존재하므로 최적화 관점에서 해가 무수히 많이 존재하는 문제점이 있다. 따라서 해의 유일성을 보장하고 좋은 분류함수를 구하기 위하여, 다음과 같이 분류경계와의 거리인 소위 마진 (margin) 을 최대화하는 분류함수를 추구하게 된다.

$$\max_{w} \frac{2}{\|w\|^2} \text{ subject to } \; y_i \; f(x_i) \geq 1, \; i = 1, \ldots, n.$$

이를 강마진 (hard margin) 지지벡터기계라고 부른다. 그러나 실제 분류문제에서 선형 분리가능하다는 가정은 현실적이지 못하므로, 어느 정도의 훈련자료의 오분류를 허용하며 마진을 최대화하는 것을 고려하게 된다. 이를 약마진 (soft margin) 지지벡터기계라 부르며 다음과 같은 원시 (primal form) 최적화 문제로 구성된다.

$$\begin{aligned} &\min_{w} \left(\frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} \xi_i \right) \\ &\text{subject to } \xi_i \geq 1 - y_i f(x_i), \xi_i \geq 0, i = 1, \ldots, n. \end{aligned} \tag{2.1}$$

여기서 $\{\xi_i\}_{i=1}^n$는 여유변수 (slack variable)라 부르며 $\langle w, x\rangle + b = \pm$에 의해 정의되는 초평면들로부터의 잘못 나뉘는 훈련자료점들의 거리를 나타낸다. 그 비율은 일종의 훈련오차로 볼 수 있다. $\lambda > 0$는 기하마진 (geometric margin)의 역수의 제곱에 비례하는 $\frac{1}{2}\|w\|^2$와 훈련오차의 상대적 비율을 조절하는 조율모수이다. 분리가능하지 않은 일반적인 경우를 고려하는 약마진 지지벡터기계를 흔히 지지벡터기계라 한다.

참고로 벌점화 (method of penalization) 관점에서 보면 (2.1)은 다음의 최적화 문제와 동치이다.

$$\min_{w} \left(\frac{1}{n}\sum_{i=1}^{n} l(w; x_i, y_i) + \frac{\lambda}{2} J[f] \right). \tag{2.2}$$

여기서 $l(w; x_i, y_i) = (1 - y_i f(x_i))_+ \equiv \max(1 - y_i f(x_i), 0)$은 경첩손실 (hinge loss)이며 $J[f] = \|w\|^2$이다.

알고리즘에 따라서는 원시 최적화 문제를 풀어서 해를 구하기도 하지만 대부분 다음과 같은 라그랑즈 승수법에 의해 유도되는 쌍대 (dual form) 최적화 문제를 푼다.

$$\begin{aligned} &\min_{\alpha} \left(\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \langle x_i, x_j\rangle - \sum_{i=1}^{n} \alpha_i \right) \\ &\text{subject to} \sum_{i=1}^{n} y_i \alpha_i = 0, 0 \leq \alpha_i \leq C, i = 1, \ldots, n. \end{aligned} \tag{2.3}$$

여기서 $C = 1/(n\lambda)$이다. 쌍대문제 (2.3)의 해 $\hat{\alpha}_i, i = 1, \ldots, n$를 얻으면 KKT 조건 (Karush-Kuhn-Tucker condition)으로부터 $\hat{w} = \sum_{i=1}^{n} \hat{\alpha}_i y_i x_i$와 $\hat{b}$를 계산할 수 있다. 쌍대문제의 유도과정 등 보다 자세한 사항은 Park 등 (2013)의 14장을 참고하기 바란다. 로지스틱 회귀 등의 방법에서는 모든 훈련자료점을 이용하여 분류함수를 구하는 반면, 지지벡터기계에서는 훈련자료중 $\hat{\alpha}_i > 0$에 대응되는 소위 지지벡터라는 훈련자료점들만이 분류함수에 기여한다는 특징이 있다. 지지벡터기계라는 이름 자체도 지지벡터라는 용어에 기인한다.

2.2. 구현 알고리즘

지지벡터기계의 구현은 결국 원시 또는 쌍대 최적화 문제의 해를 구하는 것으로 귀결된다. 쌍대문제 (2.3)에서는 내적을 커널로 대체하는 소위 커널트릭 (kernel trick)을 이용하면 쉽게 비선형 분류로 확장이 가능하다는 장점이 있다. 반면 원시문제 (2.1)은 근사해를 다루기 편하다는 장점이 있다. 여기서 목적함수 Q에 대한 ϵ 근사해 $\tilde{x}$는

$$Q(\tilde{x}) \leq Q(x^*) + \epsilon$$

을 만족하는 해로 정의된다. 단, $x^* = \arg\min_x Q(x)$이다.

지지벡터기계의 구현 알고리즘은 전통적인 방법, 쌍대 최적화 문제 기반의 방법, 원시 최적화 문제 기반의 방법 등으로 나눌 수 있다. 전통적인 방법으로는 쌍대문제를 이차계획법 (quadratic programming)으로 직접 푸는 방법과 Platt (1999)의 SMO (sequential minimal optimization)와 SVM$^{\text{light}}$처럼 쌍대문제를 일련의 작은 부분문제로 쪼개서 푸는 방법들이 있다. 전통적인 방법들은 계산량이 매우 많기 때문에 빅 데이터의 학습문제에는 전혀 적합하지 않다. 쌍대 최적화 문제에 기반한 방법으로는 쌍대 최적화 문제를 일련의 일변량 최적화 문제로 푸는 Hsieh 등 (2008)의 좌표별 강하 알고리즘 (coordinate descent algorithm)과 Smola 등 (2007)의 번들 (bundle)방법 등이 대표적이다. 원시 최적화 문제에 기반한 방법으로는 원시함수에 확률적 경사 강하 (stochastic gradient descent) 알고리즘 타입으로 다음절에 소개할 Shalev-Shwartz 등 (2011)의 PEGASOS, Bordes 등 (2008)의 SGD-QN (stochastic gradient descent with a quasi-Newton), Duchi와 Singer (2008)의 FOLOS (FOrward LOoking Subgradient) 등이 있다. 그 밖에 위의 세 범주에 속하지 않는 방법으로는 원시와 쌍대함수를 번갈아 가며 최적화하는 Forero 등 (2010)의 ADMM 이나 Franc와 Sonnenburg (2008)의 OCAS (Optimized Cutting plane AlgorithmS) 등이 있다.

3. 빅 데이터 분석을 위한 지지벡터기계 알고리즘

3.1. PEGASOS

Shalev-Shwartz 등 (2011)의 PEGASOS는 확률적 경사 강하 알고리즘으로 볼 수 있다. Bottou와 Bousquet (2008)에서 지적한 바와 같이 대규모 학습문제에서는 근사해로도 충분히 좋은 예측오차 (prediction error)를 얻을 수 있기 때문에 최적화에서 매우 정확한 해를 찾기 위해 많은 노력을 기울이는 것이 큰 의미가 없을 수 있다. 그 이유는 다음과 같이 설명할 수 있다. 학습문제에서 최적화에 의해 해를 구할 때 예측오차는 근사오차 (approximation error), 추정오차 (estimation error), 최적화오차 (optimization error)로 분해된다. 근사오차는 해를 찾는 함수공간과 관련된 오차를 나타내며, 추정오차는 이론상 훈련자료의 크기가 커지면 영으로 수렴하는 오차이며, 최적화오차는 최적화 알고리즘에 의하여 찾는 해에 의한 오차이다. 훈련자료의 크기가 매우 크면 최적화 오차를 약간 줄이기 위해 매우 복잡한 알고리즘을 사용하는것보다는 최적화 성능은 좀 떨어지지만 덜 복잡한 알고리즘을 적용함으로써 더 많은 훈련자료를 학습하여 추정오차를 줄이는 것이 더 효율적일 수 있다는 것이다. 비록 확률적 경사 강하 알고리즘은 좋은 해를 주는 최적화는 아니지만 속도가 매우 빠르기 때문에 더 많은 자료를 처리할 수 있고 따라서 더 좋은 해를 주는 최적화 방법에 비하여 추정오차를 유의하게 줄일 수 있다. 또 다른 장점은 온라인 타입의 학습에 적합하다는 것이다.

자료를 J개씩 학습한다고 가정할 때 T는 전체 n개의 자료를 학습하는데 필요한 최소한의 반복수라고 하자. 특별한 경우로서 $J = 1$이면 온라인 타입 알고리즘으로 $T = n$이다. 그러면 J개씩 학습하는 PEGASOS 알고리즘은 다음과 같다.

1. 초기화: $w^1 = 0, b^1 = 0$.

2. $t = 1, \ldots, T$에 대하여 다음을 반복한다.

   (a) $|I_t| = J$를 만족하는 첨자집합 $I_t \subset \{1, \ldots, n\}$를 랜덤하게 선택한다.

   (b) 식 (2.2)를

   $$\frac{\lambda}{2}\|w\|^2 + \sum_{i \in I_t} l(w, b; x_i, y_i) \tag{3.1}$$

   로 근사한다.

   (c) 근사 목적함수 (3.1)의 서브그래디언트 (sub-gradient)

   $$\nabla_t = \lambda \begin{pmatrix} w^t \\ 0 \end{pmatrix} - \sum_{i \in I_t} I(y_i(\langle w^t, x_i \rangle + b^t) < 1) \begin{pmatrix} y_i x_i \\ y_i \end{pmatrix}$$

   를 계산한다.

   (d) 해의 이동폭을 $\eta_t = 1/(\lambda t)$로 하여 해를

   $$\begin{pmatrix} w^{t+1} \\ b^{t+1} \end{pmatrix} \leftarrow \begin{pmatrix} w^t \\ b^t \end{pmatrix} - \eta_t \nabla_t$$

   로 갱신한다.

   (e) 사영단계: $w^{t+1} \leftarrow \min\left\{1, \frac{1/\sqrt{\lambda}}{\|w^{t+1}\|}\right\} w^{t+1}$.

3. 최종해: w^{T+1}, b^{T+1}.

사영단계는 허용가능한 해의 범위를 반지름 $1/\sqrt{\lambda}$인 구로 제한한다. 참고로 선형 커널의 경우 PEGASOS의 실행시간은 $O(d/(\lambda\epsilon))$으로 알려져 있다. 단 d는 0이 아닌 설명력이 있는 변수들의 갯수를 나타낸다. 흥미롭게도 실행시간이 훈련자료의 크기에 의존하지 않음을 알 수 있다. 비선형 커널과 다범주 분류로의 확장 및 수렴성에 대한 자세한 사항은 Shalev-Schwartz 등 (2011)을 참조하기 바란다.

3.2. ADMM

주어진 훈련자료를 J개의 블럭으로 나누는 경우를 생각해 보자. $j = 1, \ldots, J$에 대하여 $I_j \subset \{1, \ldots, n\}$는 서로 배반인 j번째 블럭의 첨자집합이라 하자. $\rho > 0$는 주어진 상수를 나타낸다. Forero 등 (2010)에서는 본래 (2.1)에서 출발하여 알고리즘을 유도하는데, 본 논문에서는 논의의 편의상 동치인 벌점화된 최적화 (2.2)를 기반으로 알고리즘을 설명하기로 한다.

$$\begin{aligned} &\text{minimize} \left(\frac{1}{2}\|z\|^2 + C\sum_{j=1}^{J}\sum_{i \in I_j} l(w_j, b_j; x_i, y_i) + \frac{\rho}{2}\sum_{j=1}^{J}\|w_j - z\|^2 \right) \\ &\text{subject to } w_j - z = 0, b_j - b = 0, j = 1, \ldots, J. \end{aligned} \tag{3.2}$$

여기서 제약조건은 각 블럭에서의 추정값이 서로 일치하도록 하기 위한 것으로 소위 전역 합의 (global consensus) 조건이라고 볼 수 있다. 만일 블럭간의 근방의 구조가 사전정보로 주어진 경우에는 사전정보를 제약조건에 반영할 수도 있다. 실제로 위의 최적화를 직접 푸는 것이 아니라 자료의 블럭으로 문제를 분할하여 반복하게 되며 반복이 진행될수록 각 블럭에서의 추정값은 점점 더 유사한 값으로 수렴하게 된다. 또한 (2.2)를 블럭으로 나누고 이차항 $\rho/2\sum_{j=1}^{J}\|w_j - z\|^2$을 추가한 소위 증대된 라그랑즈 함수 (augmented Lagrangian function) (3.2)를 최적화한다는 점에 주목할 필요가 있다. 제약조건으로 인하

여 증대된 최적화는 원래의 최적화와 동치임은 쉽게 알 수 있다. 이차항을 추가하는 이유는 해의 수치적 안정성을 위한 것으로 회귀문제에서 능형회귀나 Zou와 Hastie (2005)의 일래스틱 네트 (elastic net)에서 l_2 벌점항을 추가하는 것과 유사하다. 다소 차이가 있는 점은 ρ가 조율의 역할을 하기 보다 고정된 적당히 큰 양의 값에 대하여 w_j들 전역 합의 변수 z와 동일함을 보장할 수 있게 하는 역할을 한다. 그 밖의 ADMM 알고리즘에 대한 이론적인 고찰은 Boyd 등 (2010) 을 참고하기 바란다.

지지벡터기계의 합의 문제에 대한 ADMM 알고리즘은 다음과 같이 요약된다. 자세한 유도과정은 Forero 등 (2010)을 참고하기 바란다.

1. 초기화: w_j^1, b_j^1, u_j^1은 랜덤한 값, $z_j^1 = 0$ for $j = 1, \ldots, J$.
2. $t = 1, 2, \ldots$에 대하여 다음을 수렴할 때까지 반복한다.

   (a) $j = 1, \ldots, J$에 대하여 다음의 갱신식을 푼다.

$$(w_j^{t+1}, b_j^{t+1}) = \arg\min_{w,b} \left(C \sum_{i \in I_j} l(w, b; x_i, y_i) + \frac{\rho}{2} \| w - z^t + u_j^t \|^2 \right)$$

   (b) $j = 1, \ldots, J$에 대하여 z^{t+1}와 u_j^{t+1}를

$$\begin{aligned} z^{t+1} &= \frac{1}{J + \rho^{-1}} \sum_{j=1}^{J} (w_j^{t+1} + u_j^t) \\ u_j^{t+1} &= u_j^t + w_j^{t+1} - z^{t+1} \end{aligned}$$

   로 갱신한다.

Forero 등 (2010)에서는 앞서 언급한 블럭에 대해 wireless sensor networks와 같은 사전정보가 있는 경우를 고려하였으며, 아울러 온라인과 비선형 분류문제에 대한 알고리즘을 제시하였다. ADMM과 관련된 주요 참고문헌인 Boyd 등 (2010)에서는 회귀와 분류 등 여러 가지 학습문제에 대한 ADMM 알고리즘에 대한 적용방안을 제시하며, 입력변수들을 여러 블럭으로 나누어 처리하는 공유 문제에 대하여도 다루고 있다.

4. 모의실험

이 절에서는 지지벡터기계의 대표적인 일괄처리 알고리즘인 LIB-SVM, 온라인 방식의 PEGASOS, 그리고 병렬처리 방식의 ADMM의 예측력과 훈련시간을 모의실험을 통하여 비교한다. 자료분석에는 32개의 코어를 갖는 멀티코어 (multi-core) 시스템인 Dell R720을 사용하였다. 시스템의 CPU는 Intel Xeon CPU E5-2690 2.90GHz이고 메모리는 192GB이며 운영체제는 64비트 우분투 12.04 서버이다. LIB-SVM은 R의 `e1071` 패키지에 구현된 `svm` 함수를 이용하였고 PEGASOS와 ADMM은 R로 직접 구현하였다. ADMM의 경우 `snow` 패키지를 이용하여 여러 코어로 분산처리한 결과 코어를 하나를 사용한 것보다 실행속도가 더 느려지는 문제가 있어서 하나의 코어만을 이용하였다. 아마도 자료와 사용 패키지 등을 각 코어에 전달하고 초기화하고 다시 처리 결과를 가져오는 통신상의 비용이 큰 것으로 보인다. 실제 병렬처리에서는 이러한 비용을 줄이는 것이 현실적으로 중요한 문제로서 작업을 효율적으로 분배하고 처리하는 문제와 관련이 있다. 선형 지지벡터기계의 경우 조율모수 λ를 선택해야 하는데 자료가 큰 경우를 상정하여 훈련자료를 임의로 100개 추출하여 $\{2^{-5}, 2^{-4}, \cdots, 2^5\}$상에서 5-묶음 교차확인법 (5-fold cross validation)으로 구한 후 자료의 훈련시 고정된 값으로 사용하였다. ADMM의 경우 추가적인 모수인 ρ는 적절히 큰 값으로 고정시킬수 있으므로 $\rho = 10^4$로 설정하였다.

모의실험의 자료의 생성법은 다음과 같다. 우선 등확률로 $Y=\pm 1$를 생성한 후, $X_1, \cdots, X_r \sim N(\sqrt{2/r}Y, 1)$과 $X_r, \cdots, X_p \sim N(0,1)$를 독립적으로 생성한다. 그러면 앞의 r개의 입력변수들은 설명력이 있는 변수이고 나머지 $p-r$개의 입력변수들은 잡음변수가 된다. 이 경우 p와 r의 값에 관계 없이 항상 베이즈 오분류율은 0.0787이다. 각 방법의 특성을 비교하기 위해 설명력이 있는 입력변수들의 비율 r/p을 $0.05, 0.10, 0.15, 0.20$의 네 가지 수준에서 실험하였다. 또한 훈련자료의 크기는 $5000, 10000, 15000, 20000$의 네 가지 수준을 고려하였고 시험자료는 100000로 고정시켰다. 또한 모의실험의 변동성을 파악하기 위하여 자료를 생성하고 훈련 및 시험오차를 구하는 과정을 100회 반복하였다.

Table 4.1 Average training times of PEGASOS, LIB-SVM and ADMM over 100 replicates

		average of training times (s.e.)		
r/p	n	PEGASOS	LIB-SVM	ADMM
0.05	5000	0.1719 (0.0001)	2.6960 (0.0012)	1.9110 (0.0017)
	10000	0.3459 (0.0017)	11.9400 (0.0267)	3.9230 (0.0041)
	15000	0.5195 (0.0009)	32.3990 (0.0415)	5.9690 (0.0032)
	20000	0.6984 (0.0029)	60.5570 (0.0855)	8.1120 (0.0049)
0.10	5000	0.1653 (0.0001)	2.6950 (0.0011)	1.9130 (0.0012)
	10000	0.3328 (0.0017)	11.8040 (0.0239)	3.9210 (0.0030)
	15000	0.4973 (0.0007)	32.2300 (0.0345)	5.9800 (0.0030)
	20000	0.6692 (0.0017)	61.5080 (0.0569)	8.1180 (0.0050)
0.15	5000	0.1707 (0.0011)	2.6620 (0.0024)	1.9330 (0.0027)
	10000	0.3467 (0.0021)	13.3020 (0.0313)	3.9330 (0.0028)
	15000	0.5207 (0.0022)	38.0700 (0.0716)	6.0120 (0.0038)
	20000	0.6890 (0.0018)	75.0810 (0.0562)	8.1690 (0.0043)
0.20	5000	0.1698 (0.0008)	2.5950 (0.0018)	1.9210 (0.0021)
	10000	0.3459 (0.0022)	11.8010 (0.0208)	3.9280 (0.0027)
	15000	0.5111 (0.0014)	32.2560 (0.0427)	5.9940 (0.0031)
	20000	0.6875 (0.0021)	61.4860 (0.0638)	8.1590 (0.0056)

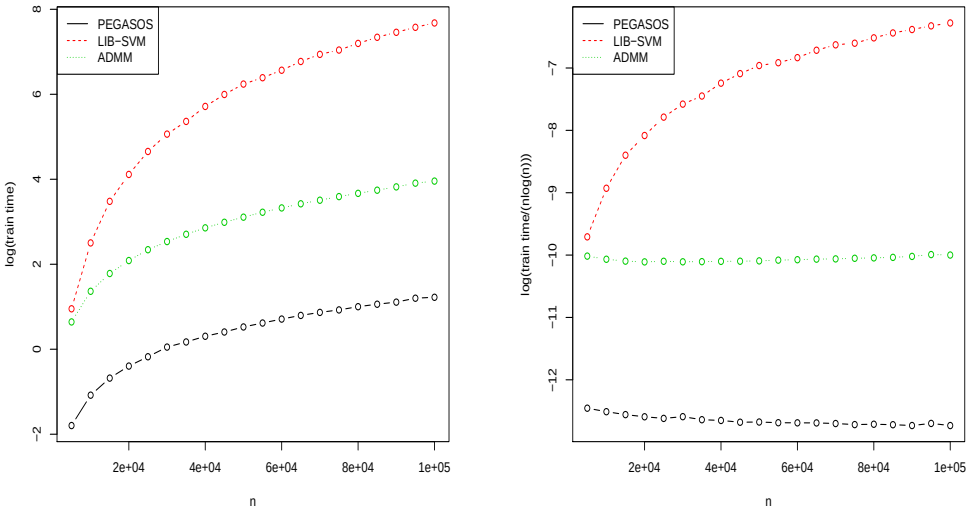
Table 4.2 Average test error rates of PEGASOS, LIB-SVM and ADMM over 100 replicates

		average of test error rates (s.e.)		
r/p	n	PEGASOS	LIB-SVM	ADMM
0.05	5000	0.1544 (0.0071)	0.0811 (0.0000)	0.0790 (0.0000)
	10000	0.1527 (0.0076)	0.0816 (0.0000)	0.0798 (0.0000)
	15000	0.1433 (0.0066)	0.0807 (0.0000)	0.0797 (0.0000)
	20000	0.1527 (0.0079)	0.0798 (0.0000)	0.0797 (0.0000)
0.10	5000	0.0809 (0.0000)	0.0822 (0.0000)	0.0783 (0.0000)
	10000	0.0818 (0.0000)	0.0822 (0.0000)	0.0802 (0.0000)
	15000	0.0808 (0.0000)	0.0809 (0.0000)	0.0798 (0.0000)
	20000	0.0798 (0.0000)	0.0803 (0.0000)	0.0792 (0.0000)
0.15	5000	0.0850 (0.0007)	0.0824 (0.0000)	0.0784 (0.0000)
	10000	0.0827 (0.0004)	0.0806 (0.0000)	0.0796 (0.0000)
	15000	0.0827 (0.0003)	0.0810 (0.0000)	0.0804 (0.0000)
	20000	0.0819 (0.0004)	0.0798 (0.0000)	0.0791 (0.0000)
0.20	5000	0.0806 (0.0000)	0.0832 (0.0000)	0.0792 (0.0000)
	10000	0.0792 (0.0000)	0.0806 (0.0000)	0.0786 (0.0000)
	15000	0.0804 (0.0000)	0.0816 (0.0000)	0.0803 (0.0000)
	20000	0.0798 (0.0000)	0.0800 (0.0000)	0.0795 (0.0000)

Table 4.1과 4.2는 r/p과 n의 각 설정별로 100회의 반복에 대한 모의실험의 결과인 평균 훈련시간과 시험오차를 보여준다. PEGASOS와 ADMM은 서로 다른 철학에 기반하므로 직접적인 훈련시간의

비교는 큰 의미가 없지만 LIB-SVM를 통한 간접적인 비교는 가능하다. 입력변수들중 설명력있는 변수의 비율에 크게 관계없이 각 방법의 훈련자료의 크기에 따른 실행시간 패턴은 비슷한 것으로 보인다. PEGASOS와 ADMM의 경우에는 훈련자료의 크기에 따라 훈련시간의 차수가 대략 선형으로 증가하는 것처럼 보이며, LIB-SVM은 제곱의 차수로 증가하는 것으로 보인다. 실제로 LIB-SVM은 SMO 알고리즘에 기반한 것으로서 빅 데이터 분석에는 적합하지 않은 알고리즘으로 알려져 있다.

훈련시간기준으로는 셋 중 PEGASOS가 가장 빠르지만 예측력은 가장 떨어지는 것을 알 수 있다. 특히 설명력 있는 변수의 비율이 5%로 작은 경우에는 훈련자료가 늘어나도 다른 두 방법의 예측오차와 큰 차이가 남을 알 수 있다. 그러나 PEGASOS의 예측력은 r/p가 증가함에 따라 다른 알고리즘의 예측력과 비슷해짐을 볼 수 있는데, 이는 아마도 PEGASOS 자체가 좋은 최적화 보다는 속도가 빠른 SGD (stochastic gradient descent) 알고리즘이기 때문에 초기치에서 출발하여 좋지 않은 국소 최적해로 빨리 수렴하기 때문에 나타나는 현상으로 보인다. 결국 잡음이 심한 자료에서는 PEGASOS의 예측력은 떨어질 것으로 예상된다. 매우 정확한 해를 추구하는 알고리즘인 LIB-SVM과 ADMM의 예측력은 r/p의 비율에 큰 상관없이 비교적 안정적으로 나타났다. 특히 ADMM의 예측력은 근소한 차이지만 LIB-SVM에 비해 더 좋음을 알 수 있다. 따라서 훈련자료의 크기가 커질수록 일괄처리 알고리즘으로 해를 구하는 것보다 자료를 적절한 블럭으로 쪼개서 해를 구하는 것이 보다 효과적이라고 할 수 있다.



(a) log(average of training times) (b) $\log\left(\frac{\text{average of training times}}{n\log n}\right)$

Figure 4.1 Comparison of average training times of PEGASOS, LIB-SVM and ADMM over 10 replicates

Figure 4.1에서는 훈련자료의 크기에 따른 세 알고리즘의 훈련시간 비교하였다. 앞의 모의실험에서 $r/p = 0.20$인 경우에 대하여 $n = 5000, 10000, \cdots, 100000$에서 10회 반복실험한 후 각 알고리즘의 평균시간을 구하였다. Figure 4.1 (a)는 n에 대하여 각 알고리즘의 평균 훈련시간의 로그값을 그린것으로 훈련속도는 PEGASOS, ADMM, LIB-SVM 순이며 LIB-SVM은 다른 두 방법에 비하여 훈련시간이 훈련자료의 크기에 따라 더욱 빨리 증가하는것으로 보인다. Figure 4.1 (b)은 각 방법의 훈련시간을 $n\log n$으로 나눈 후 로그를 취하여 비교한 결과를 나타낸다. 각 알고리즘의 훈련시간의 차수를 $n\log n$를 기준으로 비교해보면, LIB-SVM의 훈련시간은 $n\log n$보다 더 빠르게 증가하며 PEGASOS나 ADMM의 경우는 $n\log n$ 혹은 더 느리게 증가하는 것으로 나타났다.

5. 결론

본 논문에서는 분류문제의 대표적인 학습법인 지지벡터기계의 여러 알고리즘을 살펴보고, 빅 데이터 분류문제에 적합할 것으로 예상되는 온라인 타입의 PEGASOS 알고리즘과 병렬처리 알고리즘인 ADMM을 소개하였다. 또한 이 두 알고리즘을 일괄처리 방식의 구현 알고리즘인 LIB-SVM과 모의실험을 통해 훈련시간과 시험오차를 비교하였다. 빅 데이터의 분류문제에서는 LIB-SVM과 같은 일괄처리 알고리즘은 적용하기 어려운 반면 PEGASOS나 ADMM은 하드웨어 및 소프트웨어 환경, 자료처리 방식 등을 잘 고려하면 적용이 가능할 것으로 예상된다. PEGASOS는 최적화를 간단히 하여 보다 많은 자료를 학습할 수 있는 반면 ADMM은 큰 문제를 작은 부분문제로 쪼개어 학습에 필요한 계산을 줄이면서도 정확한 해를 구할 수 있다. 자료에 대한 온라인 분류가 필요한 경우에는 PEGASOS가 효과적이나 모의실험의 결과에서 확인할 수 있듯이 자료에 노이즈 변수가 많으면 분류정확도가 떨어지는 단점이 있다. ADMM은 PEGASOS에 비해 계산량이 많은 반면 자료가 여러 곳에서 수집, 관리, 처리되는 경우에 유용할 것으로 예상된다.

References

Bordes, A., Bottou, L. and Gallinari, P. (2008). SGD-QN: Careful quasi-Newton stochastic gradient descent. *Journal of Machine Learning Research*, **10**, 1737-1754.

Bottou, L. and Bousquet, O. (2008). The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems*, **20**, 161-168.

Boyd, S., Parikh, N., Chu, E., Peleato, B. and Eckstein, J. (2010). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, **3**, 1-122.

Cortes, C. and Vapnik, V. (1995). Support vector networks. *Machine Learning*, **20**, 273-297.

Duchi, J. and Singer, Y. (2009). Efficient online and batch learning using forward-backward splitting. *Journal of Machine Learning Research*, **10**, 2873-2898.

Fan, R.-E., Chen, P.-H. and Lin, C.-J. (2005). Working set selection using second order information for training SVM. *Journal of Machine Learning Research*, **6**, 1889-1918.

Forero, P. A., Cano, A. and Giannakis, G. B. (2010). Consensus-based distributed support vector machines. *Journal of Machine Learning Research*, **11**, 1663-1707.

Franc, V. and Sonnenburg, S. (2008). Optimized cutting plane algorithm for support vector machines. In *Proceedings of the 25th International Conference on Machine Learning*, ACM, 320-327.

Hsieh, C.-J., and Chang, K.-W., Lin, C.-J., Keerthi, S. S and Sundararajan, S. (2008). A dual coordinate descent method for large-scale linear SVM. In *Proceedings of the 25th International Conference on Machine Learning*, ACM, 408-415.

Park, C., Kim, Y., Kim, J., Song, J. and Choi, H. (2013). *Data mining using R*, 2nd Edition, Kyowoo Publisher, Seoul.

Park, D.-J., Yun, Y.-B. and Yoon, M. (2012). Prediction of bankruptcy data using machine learning techniques. *Journal of the Korean Data & Information Science Society*, **23**, 569-577.

Park, H.-J. (2011). Online abnormal events detection with online support vector machine. *Journal of the Korean Data & Information Science Society*, **22**, 197-206.

Pi, S.-Y., Park, H.-J. and Ryu, K.-H. (2011) An analysis of satisfaction index on computer education of university using kernel machine. *Journal of the Korean Data & Information Science Society*, **22**, 921-929.

Platt, J. C. (1999). Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernal Methods - Support Vector Learning*, MIT Press, 185-208.

Shalev-Shwartz, S., Singer, Y., Srebro, N. and Cotter, A. (2011). Pegasos: Primal estimated sub-gradient solver for SVM. *Mathematical Programming B*, **127**, 3-30.

Smola, A. J., Vishwanathan, S. V. N. and Le, Q. V. (2007). Bundle methods for machine learning. In *Advances in Neural Information Processing Systems*, **20**, MIT Press, 1377-1384.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society B*, **67**, 301-320.

Journal of the Korean Data & Information Science Society
2013, **24**(5), 989–998

http://dx.doi.org/10.7465/jkdi.2013.24.5.989
한국데이터정보과학회지

Support vector machines for big data analysis[†]

Hosik Choi[1] · Hye Won Park[2] · Changyi Park[3]

[1]Department of Informational Statistics, Hoseo University
[23]Department of Statistics, University of Seoul

Received 19 June 2013, revised 24 July 2013, accepted 5 August 2013

Abstract

We cannot analyze big data, which attracts recent attentions in industry and academy, by batch processing algorithms developed in data mining because big data, by definition, cannot be uploaded and processed in the memory of a single system. So an imminent issue is to develop various leaning algorithms so that they can be applied to big data. In this paper, we review various algorithms for support vector machines in the literature. Particularly, we introduce online type and parallel processing algorithms that are expected to be useful in big data classifications and compare the strengths, the weaknesses and the performances of those algorithms through simulations for linear classification.

Keywords: Batch processing, consensus, distributed processing, online algorithm.

[†] This research was supported by the Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology (No.2012R1A1A2004901).
[1] Assistant professor, Department of Informational Statistics, Hoseo University, Chungnam 336-795, Korea.
[2] Graduate student, Department of Statistics, University of Seoul, Seoul 130-743, Korea.
[3] Corresponding author: Associate professor, Department of Statistics, University of Seoul, Seoul 130-743, Korea. E-mail: park463@uos.ac.kr