SI 251 Project Report: Designing Statistical Estimators That Balance Sample Size, Risk, and Computational Cost

Yedi Zhang
School of Information Science and Technology
ShanghaiTech University

Abstract

This project is based on the J. J. Bruer's work in [1]. After reading the paper and some other related articles, I implemented their experiments for a better understanding. The experiment was performed under MATLAB 2016b.

1 Introduction

1.1 Background

Massive data presents an obvious challenge to statistical algorithms. While it seems natural that larger problems require more effort to solve, some researchers have found that for some convex optimization problem, there exists a trade-off between time and data. Shalev-Shwartz and Srebro [2] showed that their algorithm for learning a support vector classifier actually becomes faster as the amount of training data increases. This and more recent works support an emerging viewpoint that treats data as a computational resource. In other words, we can exploit additional data to improve the performance of statistical algorithms. What's more, some other researchers have also identified related tradeoffs. Bottou and Bousquet [3] show that approximate optimization algorithms exhibit a tradeoff between small- and large-scale problems. Agarwal et al. [4] address a tradeoff between error and computational effort in statistical model selection problems.

This paper's work bears most similarity to that of Chandrasekaran and Jordan [5]. The difference is, Chandrasekaran et.al. use an algebraic hierarchy of convex relaxations to achieve a time-data tradeoff for a class of denoising problems. while this work use a continuous sequence of relaxations based on smoothing. Finally, this paper gives a trade-off between data size, risk and computation cost in regularized linear regression problems.

1.2 Objective

After understanding the whole work, here comes two schemes in my minds for my project:

- Theoretically: According to [6], demixing with a random model should be another problem that can illustrate the authors' smoothing approach. Under this assumption, prove it theoretically.
- Experimentally: Implement the experiment in [1].

Due to the limitation of time, I choose the second scheme to complete my project. However, for the first scheme, I think it should be a worthy work, and maybe I'll realize it in the future.

1.3 Organization of the report

The rest of the report is organized as follows: In Section 2, I briefly introduce some theoretical basis and this paper's idea. In Section 3, I propose the method I used to generate the model I used in the following experiment. In Section 4, I point out some clerical errors existing in this paper and fix them. And then show the experimental results. Finally, in section 5, I analysis the result and conclude the report.

2 Theoretical Basis

2.1 Data Model

The linear regression problem can be represented via following equation:

$$b = Ax^\natural + v, \quad A \in \mathbb{R}^{m\times d}$$

where the $x^\natural$ is the given vector of parameters. This paper consider the case where $m < d$. The goal of the regression problem is to infer the underlying parameters $x^\natural$ from the data.

2.2 Phase Transition

Phase Transition in [6] gives us such conclusion: When the number m of samples is smaller than this quantity, the worst-case statistical risk is simply the noise power σ^2, and the regularized linear regression problem has no robustness to noise. That is, as the number of samples increases towards the phase transition, the statistical accuracy of the solution does not improve. After crossing the phase transition, however, additional samples decrease the worst-case risk at the rate $1/m$.

i.e., whenever $m > \delta$:

$$|\max_{\sigma>0} \frac{\mathbb{E}_v[R(x^*)|A]}{\sigma^2} - \frac{\delta}{m}| \leq tm^{-1}\sqrt{d}$$

2.3 Relaxed Regularizer

Given a regularizer f for regression problem, the authors introduce a family $\{f_\mu : \mu > 0\}$ of strongly convex majorants:

$$f_\mu(x) := f(x) + \frac{\mu}{2}||x||^2$$

In this case, with the larger value of μ, the majorants also have larger sublevel sets, and their descent cones have larger statistical dimension.

2.4 The Authors' idea

By enlarging the sublevel sets of the regularizer f, we increase the statistical dimension of the descent cone of f at $x^\natural$. *Phase Transition* tells us that the solution to the regression problem with the relaxed regularizer f_μ will have higher risk. If, however, the relaxed regularizer results in a problem that is easier to solve computationally, then we have a tradeoff between sample size, computational time, and statistical accuracy. In fact, with the relaxed regularizer mentioned above, as the convexity (i.e. μ) of the function f_μ increases, the authors have proved that the computation cost is decreasing. Then, we get the trade-off. What's more, the authors also give 3 smoothing schemes about choosing μ: Constant Smoothing, Constant Risk and a Tunable Balance method.

3 Model Generating

In my experiment, since the computer cannot deal with such a huge scale model, I reduce the primal models author used in their paper by 10,100 times respectively.

3.1 Sparse Vector Regression

The data for the sparse vector regression experiment were generated as follows. Fix the ambient dimension $d = 400$ (resp. $d = 4000$). For each of the smoothing schemes described in 2.4 and

Tabelle 1: Model scale and parameter adjustment

problem	scale	$\bar{\delta}$	$\bar{\mu}$	$\bar{m}$	σ	sample
vec	40000	98000	0.1	10000	0.01	10000:2000:38000
mat	200×200	98000	0.1	10000	0.01	10000:2000:37500
vec$_{adj}$	400	90	0.1	110	0.01	110:10:390
vec$_{adj}$	4000	894	0.1	957	0.01	1000:100:3900
mat$_{adj}$	20×20	87	0.1	107	0.01	110:10:390
mat$_{adj}$	40×40	350	0.1	390	0.01	400:100:1500

each value of the sample size $m = 110 : 20 : 390$ (resp. $m = 1000 : 100 : 3900$), perform 5 trials of *trade-off experiment*, and average the result. For more details about one single time *trade-off experiment*, please look at appendix A 5. And part of main code can be found in appendix C 5.

3.2 Low-rank Matrix Regression

The data for the low-rank matrix regression experiment were generated as follows. Fix the ambient dimensions $d = 20 \times 20$ (resp. 40×40). For each of the smoothing schemes described in 2.4 and each value of the sample size $m = 110 : 20 : 390$ (resp. $400 : 100 : 1500$), perform 10 trials of *trade-off experiment*, and average the results. For more details about one single time *trade-off experiment*, please look at appendix B 5. And part of main code can be found in appendix C 5.

Note that, after re-scale the model size, we need recomputer the corresponding parameters needed in the experiment. And all the data about scale and parameters can be found in Table 1. σ^2 is the variance of noise, and the top two problems are the primal models used in this paper.

4 Implementation

4.1 Fix Some Clerical Errors

There are two Clerical errors in this paper about Auslender-Teboulle Algorithm, and here shows how to fix them, according to [7]:

- $x_k \leftarrow \mu\cdot$ SoftTresh$(A^T y_k, 1) \Rightarrow x_k \leftarrow \mu^{-1}\cdot$SoftTresh$(A^T z_k, 1)$
- $\bar{z_k} \leftarrow$Shrink$(\bar{z_k} - (b - Ax_k)/(L_\mu \cdot \theta_k), \epsilon/(L_\mu \cdot \theta)) \Rightarrow \bar{z_k} \leftarrow$Shrink$(\bar{z_k} + (b - Ax_k)/(L_\mu \cdot \theta_k), \epsilon/(L_\mu \cdot \theta))$

4.2 Experimental Results

Fig1 and Fig3 show the experiment result under two different regularized linear regression problems: sparse vector regression, low-rank matrix regression. In each figure, the blue, red and yellow curves represents *constant smoothing, constant risk and tunable balance scheme* respectively. We can see that, the tunable balance scheme has both good computation cost and risk properties.

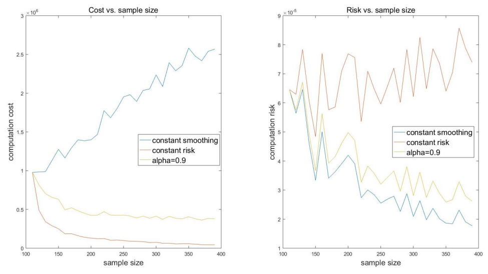
Fig2 and Fig4 show relationship between statistical dimension δ, smoothing parameter μ and sample size m. We can see that the two linear regression problems have the quite similar curves.

Finally, fig5 show the trade-off result when there is no noise.

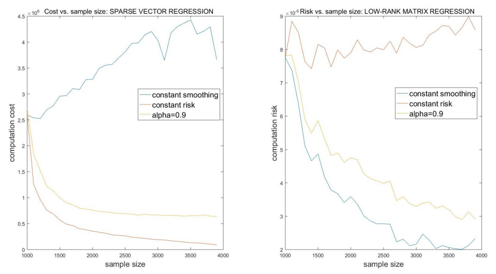
5 Conclusion

To sum up, here gives the following conclusions of this work:

- When we have excess samples in the data set, we can exploit them to decrease the statistical risk of estimator, or to lower the computational cost through additional smoothing.
- When there is no noise, we can recover the unknown signal faster with more relaxation on regularized function (using “constant riskßcheme) without losing any accuracy.

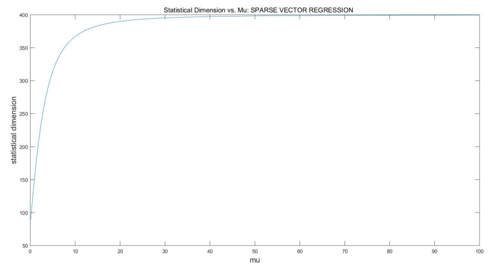


(a) $d = 400$

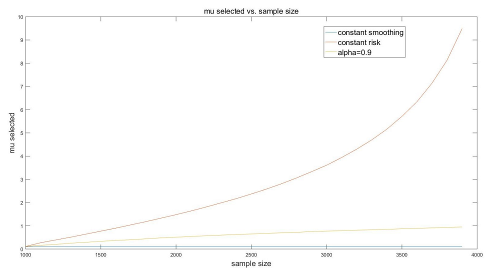


(b) $d = 4000$

Abbildung 1: Sparse Vector regression experiment

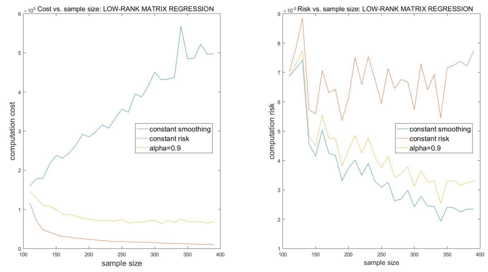


(a) δ vs. μ

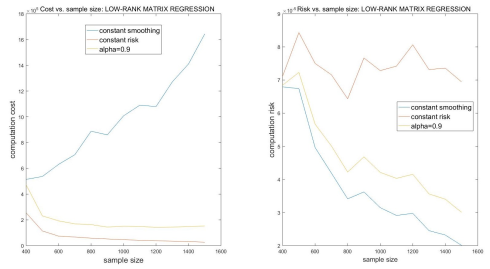


(b) μ vs. m

Abbildung 2: Relation between δ, μ and m in sparse vector regression

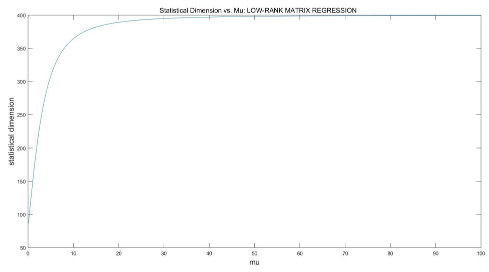


(a) $d = 20 \times 20$

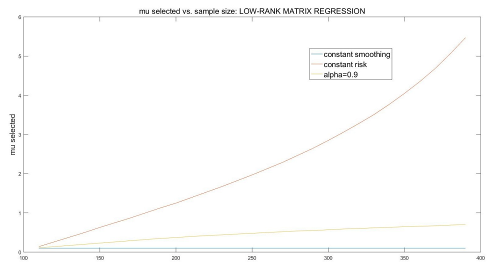


(b) $d = 40 \times 40$

Abbildung 3: Low-rank Matrix regression experiment

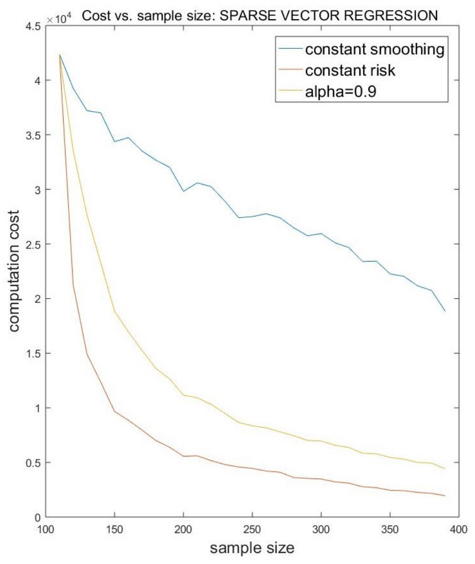


(a) δ vs. μ

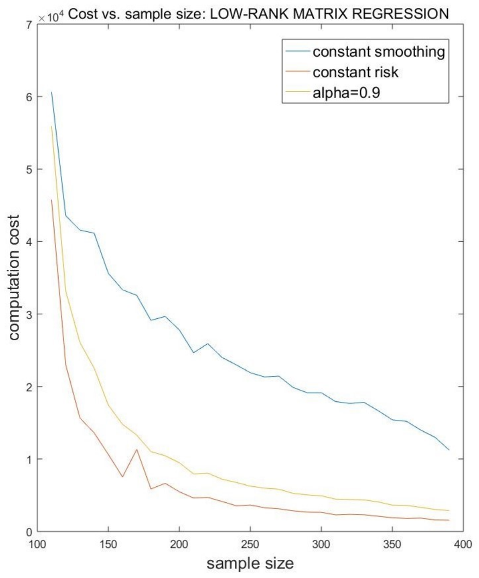


(b) μ vs. m

Abbildung 4: Relation between δ, μ and m in Low-rank Matrix regression



(a) Sparse Vector with $d = 400$



(b) Low-rank Matrix with $d = 20 \times 20$

Abbildung 5: No noise trade-off experiment

Appendix

Appendix A

Once we get the scale of problem, the one time *trade-off experiment* is as follows:

- Generate a sparse vector $x^\natural$ with 20 (resp. 2000) nonzero entries placed uniformly at random, each taking the value either -1 or +1 independently with equal probability.
- Choose a measurement matrix $A \in \mathbb{R}^{m\times 400}$ (resp. $A \in \mathbb{R}^{m\times 4000}$) uniformly at random from the ensemble of $m \times 400$ (resp. $m \times 400$) matrices with orthonormal rows (see [8] for the numerical details, as some care must be taken to ensure the appropriate distribution).
- Calculate the smoothing parameter μ according to the current scheme and the resulting statistical dimension $\delta = \delta(\mathcal{D}(f_\mu; x^\natural))$.
- Set the parameter $\epsilon = \sigma(m-\delta)^{\frac{1}{2}}$.
- Use the Auslender–Teboulle algorithm to solve the dual-smoothed sparse vector regression problem.
- Stop the algorithm when the relative primal feasibility gap $|||Ax_k - b|| - \epsilon|/\epsilon < 0.001$.
- Store the computational cost $k \cdot m \cdot 400$ (resp. $k \cdot m \cdot 4000$) and the (average) squared prediction error $||A(\hat{x} - x^\natural)||^2/m$, where $x^\natural$ is the final value of the primal iterate.

Appendix B

Once we get the scale of problem, the one time *trade-off experiment* is as follows:

- Generate a low-rank matrix $X^\natural := Q_1 Q_2^T$, where the Q_i are chosen uniformly at random from the ensemble of 20×1 (resp. 40×2) matrices with orthonormal columns.
- Choose a measurement matrix $A \in \mathbb{R}^{m\times 400}$ (resp. $A \in \mathbb{R}^{m\times 1600}$) uniformly at random from the ensemble of $m \times 400$ (resp. $m \times 1600$) matrices with orthonormal rows.
- Calculate the smoothing parameter μ according to the current scheme and the resulting statistical dimension $\delta = \delta(\mathcal{D}(f_\mu; x^\natural))$.
- Set the parameter $\epsilon = \sigma(m-\delta)^{\frac{1}{2}}$.
- Use the Auslender–Teboulle algorithm to solve the dual-smoothed low-rank matrix regression problem.
- Stop the algorithm when the relative primal feasibility gap $|||A \cdot vec(X_k) - b|| - \epsilon|/\epsilon < 0.001$.
- Store the computational cost $k \cdot m \cdot 400$ (resp. $k \cdot m \cdot 1600$) and the (average) squared prediction error $||A \cdot vec(\hat{X} - X^\natural)||^2/m$, where $X^\natural$ is the final value of the primal iterate.

Appendix C

Auslender-Teboulle Algorithm of Sparse Vector:

```matlab
function [time,risk] = Auslender_Teboulle(A,true_x,b,mu,epislon)
[m,d] = size(A);
z0 = zeros(m,1);
zp0 = z0;
theta0 = 1;
gap=1;

% this is for scheme-1
L_mu = 1/mu * (norm(A,2)^2) ;
time = 0;
while(gap>0.01)
    x0 = 1/mu * soft(A'*z0,1);
    zp1 = Shrink(z0+(b-A*x0)/(L_mu*theta0),epislon/(L_mu*theta0));
    z1 = (1-theta0)*z0+theta0*zp1;
    theta1 = 2/(1+(1+4/(theta0^2))^0.5);
    %gap = norm(true_x-x0,2)/norm(true_x,2) % without noise
    gap = abs(norm((A*x0-b),2)-epislon)/epislon
    theta0 = theta1;
    z0 = z1;
    zp0 = zp1;
    time = time + 1;
end
risk = (norm((A*x0-A*true_x),2)^2)/m;
end
```

One time trade-off experiment of Sparse Vector:

```matlab
function [cost1,risk1,cost2,risk2,cost3,risk3] = solver(A,true_x,b,sigma)
    [m,d] = size(A);
    x_inf_norm = 1;
    mu_p = 0.10;
    delta_p = delta_upper(0.05,x_inf_norm,mu_p);
    m_p = delta_p+sqrt(d);
    epislon = sigma * ((m-delta_p)^0.5);

    best_mu_table1 = 0.1; % mu always be 0.1
    %d=400;
    best_mu_table2 = [0.10,0.21,0.31,0.42,0.53,0.64,0.75,0.87,0.99,1.11,1.23,1.36,
    1.50,1.64,1.79,1.94,2.10,2.27,2.45,2.64,2.84,3.06,3.29,3.54,3.82,
    4.11,4.45,4.81,5.23];
    best_mu_table3 = [0.10,0.12,0.14,0.17,0.20,0.23,0.25,0.28,0.30,0.32,0.34,0.36,
    0.38,0.40,0.42,0.44,0.46,0.47,0.49,0.50,0.52,0.53,0.55,0.56,0.57,
    0.59,0.60,0.61,0.62];
    %d=4000;
    %best_mu_table2 = [0.11,0.27,0.39,0.51,0.64,0.77,0.90,1.04,1.33,1.48,1.64,1.81,
    1.99,2.17,2.37,2.58,2.81,3.06,3.33,3.61,3.94,4.30,4.70,5.16,5.17,
    6.34,7.14,8.13,9.49];
    %best_mu_table3 = [0.11,0.16,0.20,0.25,0.29,0.33,0.37,0.40,0.44,0.48,0.51,0.54,
    0.57,0.60,0.62,0.65,0.67,0.70,0.72,0.75,0.77,0.79,0.81,0.83,0.85,
    0.87,0.89,0.91,0.93];

    % for scheme-1
    [cost1,risk1] = Auslender_Teboulle(A,true_x,b,mu_p,epislon);

    %for scheme-2
    index = (m-10)/10 + 1;
    mu2 = best_mu_table2(index);
    delta2 = delta_p*m/m_p;
    delta2 = delta_upper(0.05,x_inf_norm,mu2);
    epislon2 = sigma * ((m-delta2)^0.5)
    [cost2 , risk2] = Auslender_Teboulle(A,true_x,b,mu2,epislon2);

    %for scheme-3
    mu3 = best_mu_table3(index);
    delta3 = delta_p*m/(m_p+(m-m_p)^0.9);
    epislon3 = sigma * ((m-delta3)^0.5);
    [cost3, risk3] = Auslender_Teboulle(A,true_x,b,mu3,epislon3);

end
```

Auslender-Teboulle Algorithm of Low-rank Matrix:

```
function [time,risk] = Auslender_Teboulle(A,true_X,b,mu,epislon)
[m,d] = size(A);
z0 = zeros(m,1);
zp0 = z0;
theta0 = 1;
gap=1;

% this is for scheme-1
L_mu = 1/mu * (norm(A,2)^2) ;
time = 0;
while(gap>0.01)
    y0 = (1-theta0) * z0 + theta0 * zp0;
    mm = my_mat(A'*z0);
    [U,S,V]=svd(mm,'econ');

    x0 = 1/mu * U * diag((soft(diag(S),1)))*V';

    zp1 = Shrink(z0+(b-A*vec(x0))/(L_mu*theta0),epislon/(L_mu*theta0));
        % 20*20
    z1 = (1-theta0)*z0+theta0*zp1;
    theta1 = 2/(1+(1+4/(theta0^2))^0.5);
    gap = abs(norm((A*vec(x0)-b),2)-epislon)/epislon
    %gap = norm(true_X-x0,2)/norm(true_X,2);
    theta0 = theta1;
    z0 = z1;
    zp0 = zp1;
    time = time + 1;
end
risk = (norm(A*vec(x0-true_X),2)^2)/m;
end
```

One time trade-off experiment of Low-rank Matrix:

```
function [cost1,risk1,cost2,risk2,cost3,risk3] = solver(A,true_X,b,sigma
    )
    [m,d] = size(A);
    mu_p = 0.1;
    delta_p = delta_upper(true_X,mu_p);
    m_p = delta_p+sqrt(d);
    epislon = sigma * ((m-delta_p)^0.5) ;


    %best_mu_table1 = 0.1;
    %20*20
    %best_mu_table2 =
        [0.14,0.26,0.38,0.50,0.63,0.75,0.87,1.00,1.13,1.25,1.39,1.53,
    1.67,1.82,1.97,2.13,2.29,2.47,2.65,2.85,3.06,3.28,3.51,3.77,4.05,
    4.35,4.68,5.06,5.47];
    %best_mu_table3 =
        [0.11,0.14,0.17,0.20,0.23,0.26,0.29,0.32,0.35,0.37,0.40,0.42,
    0.44,0.46,0.48,0.50,0.52,0.54,0.55,0.57,0.59,0.60,0.62,0.63,0.65,
    0.66,0.67,0.69,0.70];

    %40*40
    best_mu_table2 =
        [0.20,0.50,0.90,1.20,1.60,2.00,2.40,3.00,3.60,4.30,5.30,6.80];
    best_mu_table3 =
        [0.11,0.23,0.33,0.42,0.50,0.58,0.64,0.70,0.75,0.80,0.85,0.89];
    % for scheme-1
    [cost1,risk1] = Auslender_Teboulle(A,true_X,b,mu_p,epislon);

    %for scheme-2
    index = (m-400)/100 + 1;
    mu2 = best_mu_table2(index);
    delta2 = delta_p*m/m_p;
    epislon2 = sigma * ((m-delta2)^0.5);

    [cost2 , risk2] = Auslender_Teboulle(A,true_X,b,mu2,epislon2);

    %for scheme-3
    mu3 = best_mu_table3(index);
    delta3 = delta_p*m/(m_p+(m-m_p)^0.9);
    epislon3 = sigma * ((m-delta3)^0.5);
    [cost3, risk3] = Auslender_Teboulle(A,true_X,b,mu3,epislon3);
end
```

References

[1] J. J. Bruer, J. A. Tropp, V. Cevher & S. R. Becker. Designing statistical estimators that balance sample size, risk, and computational cost. *IEEE Journal of Selected Topics in Signal Processing*, 9(4):612–624, June 2015.

[2] S. Shalev-Shwartz & N. Srebro. SVM optimization: inverse dependence on training set size. In *Proceeding of the 25th international conference on Machine Learning*, pp. 928–935, 2008.

[3] L. Bottou & O. Bousquet. The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems 20*, pp. 161–168, 2008.

[4] A. Agarwal, P. L. Bartlett, & J. C. Duchi. Oracle inequalities for computationally adaptive model selection. In *arXiv*, 2012, 1208.0129v1.

[5] V. Chandrasekaran & M. I. Jordan. Computational and statistical tradeoffs via convex relaxation. In *Proceedings of the National Academy of Sciences*, vol. 110, 2013.

[6] D. Amelunxen, M. Lotz, M. B. McCoy & J. A. Tropp. Living on the edge: A geometric theory of phase transitions in convex optimization. *Information and Inference*, 2014.

[7] J. J. Bruer, J. A. Tropp, V. Cevher & S. R. Becker. Time-Data Tradeoffs by Aggressive Smoothing. In *Advances* in *Neural Information Processing Systems 27 (NIPS 2014)*, pp. 1664-1672, 2014.

[8] F. Mezzadri. How to generate random matrices from the classical compact groups. *Notices Amer. Math. Soc.*, vol. 54, no. 5, pp. 592–604, 2007.