ข้อเสนอโครงงานวิศวกรรมไฟฟ้า วิชา 2102499 ปีการศึกษา 2555
การเปรียบเทียบขั้นตอนวิธีแบบเร็ว สำหรับปัญหาการประมาณที่ใช้นอร์ม-1
A Comparison of Fast Algorithms for ℓ_1-Type Penalized Estimation Problems

ชื่อนิสิต นายพุฒิไชย เลิศกุลทานนท์ รหัสนิสิต 5230367921 SRA: Advanced Control and Optimization

อาจารย์ที่ปรึกษา อ.ดร.จิตโกมุท ส่งศิริ ห้องปฏิบัติการวิจัย Control System Research Laboratory

1 บทนำ

1.1 หลักการและความเป็นมา

ในการแก้ปัญหาการประมาณแบบจำลองระบบจากข้อมูลที่มี เราสามารถเขียนปัญหาดังกล่าว เป็นปัญหาการหาค่าเหมาะสุด ในรูป

$$\text{minimize} \quad f(x) + \lambda \|x\|_1 \tag{1}$$

โดยที่ $x \in \mathbf{R}^n$ เป็นตัวแปรสำหรับปัญหาการหาค่าเหมาะสุด แสดงถึงค่าพารามิเตอร์ของแบบจำลองที่ต้องการประมาณ, $f(x)$ เป็นฟังก์ชันการสูญเสีย เช่น ฟังก์ชันการสูญเสียแบบกำลังสอง (quadratic loss function) หรือ ฟังก์ชันความเป็นไปได้ (negative of log-likelihood function), $\|x\|_1$ เรียกว่าเป็นฟังก์ชันลงโทษแบบนอร์ม-1 ทำหน้าที่ควบคุมขนาดของตัวแปร x, และ จำนวนจริง $\lambda > 0$ จะเรียกว่าเป็น พารามิเตอร์ลงโทษ (penalty parameter)

โดยปรกติ การหาค่าต่ำสุดของ $f(x)$ เพียงอย่างเดียว จะเป็นการหาค่าพารามิเตอร์ของแบบจำลอง ที่สามารถอธิบายข้อมูลที่มีได้ดีที่สุด (อย่างเช่น พารามิเตอร์ที่ทำให้ผลต่างของข้อมูลจริง กับแบบจำลอง มีขนาดน้อยที่สุด เป็นต้น) ส่วนการเพิ่มฟังก์ชันลงโทษรูปแบบใดๆก็ตาม ลงในฟังก์ชันจุดประสงค์ จะเป็นการปรับปรุงคุณสมบัติของผลเฉลยปัญหา (1) ให้มีรูปแบบตามต้องการขึ้นกับรูปแบบของฟังก์ชันลงโทษที่เพิ่มลงไป แต่ทั้งนี้ ก็จะต้องแลกกับการยอมให้พารามิเตอร์ที่หาได้ อธิบายข้อมูลจริงได้แย่ลง โดยทั้งหมดนี้ จะมีพารามิเตอร์ λ เป็นตัวกำกับว่า ในการหาผลเฉลยของปัญหาการหาค่าเหมาะสุด เราจะให้ความสำคัญกับ ความสามารถในการอธิบายข้อมูลจริง ของพารามิเตอร์ที่หาได้ หรือ รูปแบบของผลเฉลยที่ตรงตามเงื่อนไขที่ต้องการมากกว่ากัน นั่นคือ หากพารามิเตอร์ λ มีขนาดเล็ก แสดงว่าในปัญหานั้น เราให้ความสำคัญกับความสามารถในการอธิบายข้อมูลจริง ของพารามิเตอร์มากกว่า

สำหรับปัญหา (1) ฟังก์ชันลงโทษแบบนอร์ม-1 ถูกเลือกมาใช้ เมื่อเราต้องการให้พารามิเตอร์ของแบบจำลองบางค่า มีค่าเป็นศูนย์ [1] และ เราจะเรียก ลักษณะของผลเฉลย x ที่มีสมาชิกเป็นศูนย์จำนวนมาก ว่าเป็น ผลเฉลยเบาบาง (sparse solution) ซึ่งผลเฉลย x ในลักษณะนี้ จะหมายความว่า พารามิเตอร์ที่ใช้อธิบายแบบจำลองลดจำนวนลง นั่นคือ ความซับซ้อนของแบบจำลองจะลดลง

ดังนั้นรูปแบบปัญหา (1) จึงได้รับความนิยมเป็นอย่างมาก ในการแทนระบบที่ซับซ้อน ด้วยแบบจำลองที่ประหยัด (parsimonous model) เช่น ในปัญหาการหาแบบจำลองข้อมูลอนุกรมเวลา fMRI (Functional Magnetic Resonance Imaging) ซึ่งมีจำนวนตัวแปร คือ ระดับออกซิเจนในเลือดที่เลี้ยงสมอง เป็นจำนวนมากในระดับหลักหมื่น เป็นต้น

ในโครงงานนี้ เราจะศึกษา และ เปรียบเทียบขั้นตอนวิธีที่จะนำมาใช้แก้ปัญหาการประมาณที่ใช้ฟังก์ชันลงโทษแบบนอร์ม-1 โดยขั้นตอนวิธีที่จะเลือกมาทดสอบ เช่น ADMM (Alternating Direction Method of Multipliers) [2], FISTA (Fast Iterative Shrinkage-Thresholding Algorithm) [3], และ ขั้นตอนวิธีอื่นๆ สำหรับแก้ปัญหา Lasso [4] โดย 2 วิธีแรกเป็นขั้นตอนวิธี ที่ได้รับความนิยมสูง ในการแก้ปัญหาคอนเวกซ์ หลังจากการทดสอบ จะนำขั้นตอนวิธีที่มีประสิทธิภาพ มาแก้ปัญหาการหาแบบจำลองข้อมูลอนุกรมเวลา fMRI

1.2 งานวิจัยที่เกี่ยวข้อง

ที่ผ่านมา มีงานวิจัยจำนวนมาก ให้ความสนใจเกี่ยวกับปัญหาการประมาณที่ใช้ฟังก์ชันลงโทษแบบนอร์ม-1 เนื่องจากปัญหาในรูปแบบดังกล่าว มีที่ประยุกต์ใช้งานในหลายด้าน ทั้งในปัญหาการจำแนกหมวดหมู่ (classification problems) เช่น การวิเคราะห์ผู้ป่วยโรคมะเร็งจากข้อมูลโปรตีนในเลือด [5, บทที่ 18], การจำแนกผู้ป่วยเป็นเนื้องอกจากการวิเคราะห์ลำดับยีน เป็นต้น หรือในปัญหาการหาแบบจำลองของระบบ

สำหรับในปัญหาการหาแบบจำลองของระบบ ฟังก์ชันสูญเสีย $f(x)$ ในปัญหา (1) มักเป็นพจน์ที่แสดงถึงความคลาดเคลื่อนระหว่าง ข้อมูลจริง กับ ผลตอบจาก

แบบจำลองที่หาได้ โดยที่ ตัวแปร x และ พารามิเตอร์อื่นๆในฟังก์ชันสูญเสีย ก็จะมีความหมายแตกต่างกันออกไปในปัญหาแต่ละด้าน อาทิ

ปัญหาการหาแบบจำลองข้อมูลคลื่นไฟฟ้าหัวใจ [6] มีฟังก์ชันการสูญเสีย คือ

$$f(V_E) = \|AV_E - V_T\|_2$$

โดยที่ V_E คือ เมทริกซ์ศักย์ไฟฟ้าบริเวณพื้นผิวของหัวใจ ซึ่งเป็นตัวแปรในปัญหานี้, V_T คือ เมทริกซ์ศักย์ไฟฟ้าที่วัดได้บริเวณผิวหนัง และ A เป็นเมทริกซ์ถ่ายโอนจาก V_E ไปยัง V_T ซึ่งได้จากการแก้ปัญหาค่าขอบจากสมการลาปลาซ

ปัญหาการแก้ไขภาพเบลอ [3] ฟังก์ชันการสูญเสียนิยามโดย

$$f(x) = \|Ax - b\|_2^2$$

โดยที่ b และ x เป็นเวกเตอร์แทนความเข้มของแต่ละจุดภาพของภาพเบลอ และ ของภาพจริงที่ต้องการกู้คืน ตามลำดับ, A เป็น เมทริกซ์ตัวดำเนินการเบลอ

นอกจากนั้น ยังมีงานวิจัยอีกหลายชิ้น ที่ให้ความสนใจศึกษา เปรียบเทียบ การใช้ฟังก์ชันลงโทษแบบต่างๆ หรือ ศึกษาขั้นตอนวิธีต่างๆ ที่สามารถนำมาใช้แก้ปัญหาการประมาณ เช่น ใน [7] ได้ศึกษา และ เปรียบเทียบการใช้ฟังก์ชันลงโทษแบบนอร์ม-1 และ แบบ ℓ_2 ในการแก้ปัญหาการเลือกตัวแปร (feature selection) พบว่า การใช้ ℓ_1-regularization มีข้อได้เปรียบคือในกรณีที่ปัญหาการหาแบบจำลอง มีข้อมูลตัวแปรเป็นจำนวนมาก แต่ตัวแปรที่สามารถใช้บรรยายระบบได้ดีมีอยู่จำนวนน้อย หรือ ใน [8] ได้ศึกษาขั้นตอนวิธีต่างๆที่จะนำมาใช้ในการแก้ปัญหาการประมาณที่ใช้ฟังก์ชันลงโทษแบบนอร์ม-1 เช่น วิธีจุดภายใน (interior-point method) และ วิธี Gauss-Seidel นอกจากนั้น ยังได้เสนอเทคนิคการแก้ปัญหาดังกล่าวเพิ่มเติม คือ วิธีการประมาณเป็น smooth ℓ_1 คือ ประมาณฟังก์ชันลงโทษแบบนอร์ม-1 ให้อยู่ในรูปฟังก์ชันลอการิทึม เพื่อให้สามารถหาอนุพันธ์ได้ ทำให้แก้ปัญหาการประมาณได้ง่ายขึ้น เป็นต้น

2 ทฤษฎีพื้นฐาน

ในการปัญหาการหาแบบจำลองของข้อมูลต่างๆ มักจะเขียนปัญหาให้อยู่ในรูปแบบปัญหาการหาค่าเหมาะสุด ซึ่งก็มีขั้นตอนวิธีมากมายรองรับในการแก้ปัญหา [9] แต่ถ้าหากสามารถระบุได้ว่าปัญหาดังกล่าว เป็นปัญหาคอนเวกซ์ด้วยแล้ว จะมีข้อได้เปรียบคือ ค่าต่ำสุดที่หาได้ จะเป็นค่าต่ำสุดสัมบูรณ์ (global optimum) และจะมีขั้นตอนวิธีอีกหลายแบบที่เจาะจงสำหรับแก้ปัญหาคอนเวกซ์โดยตรง [1] ทำให้การแก้ปัญหาเป็นไปอย่างมีประสิทธิภาพยิ่งขึ้น

ปัญหาคอนเวกซ์

เราจะเรียกปัญหาการหาค่าเหมาะสุด ที่มีฟังก์ชันจุดประสงค์เป็นฟังก์ชันคอนเวกซ์ และ เซตเงื่อนไขบังคับ เป็นเซตคอนเวกซ์ ว่าเป็นปัญหาคอนเวกซ์

ปัญหากำลังสองต่ำสุด

ปัญหากำลังสองต่ำสุด (Least-squares problems) เป็นปัญหาการหาค่าเหมาะสุดแบบไม่มีเงื่อนไขบังคับ และ มีฟังก์ชันจุดประสงค์อยู่ในรูปแบบผลบวกของพจน์กำลังสอง นั่นคือปัญหาในรูปแบบ

$$\text{minimize} \quad \|Ax - b\|_2^2 \tag{2}$$

โดยที่ $A \in \mathbf{R}^{m\times n}(m \geq n)$ และ เวกเตอร์ $x \in \mathbf{R}^n$ เป็นตัวแปร

ปัญหาดังกล่าว สามารถลดรูปเป็นการแก้ระบบสมการเชิงเส้น $(A^TA)x = A^Tb$ จากการใช้เงื่อนไขเกรเดียนต์เป็นศูนย์ ที่จุดต่ำสุดของฟังก์ชันจุดประสงค์ หาก A เป็นเมทริกซ์ที่มีค่าลำดับชั้นเต็ม ปัญหากำลังสองต่ำสุดจะมีผลเฉลยในรูปแบบปิดคือ

$$x = (A^TA)^{-1}A^Tb$$

เนื่องจากรูปแบบปัญหาที่ง่าย และ การมีผลเฉลยของปัญหาในรูปแบบปิดด้วย ทำให้ปัญหา (2) ถูกนำมาใช้อย่างแพร่หลาย ในปัญหาการประมาณ หรือ บ่อยครั้ง ถูกนำมาใช้เป็นวิธีเปรียบเทียบ (based-line method) กับปัญหาการประมาณในรูปแบบอื่นๆ

เมื่อแก้ปัญหากำลังสองต่ำสุดแล้ว โดยทั่วไป พบว่าผลเฉลย x จะมีลักษณะเป็นเวกเตอร์หนาแน่น (dense vector) ดังนั้น จึงมีการปรับปรุงรูปแบบปัญหากำลังสองต่ำสุด เพื่อให้ผลเฉลย x มีคุณสมบัติอื่นๆ

ในที่นี้ จะกล่าวถึงรูปแบบปัญหาที่เรียกว่า Lasso [4] ซึ่งเขียนอยู่ในรูป

$$\text{minimize} \quad \frac{1}{2}\|Ax - b\|_2^2 + \lambda\|x\|_1 \tag{3}$$

พจน์ $\|x\|_1$ ที่เพิ่มเข้าไปในฟังก์ชันจุดประสงค์ จะเป็นเหมือนการเพิ่มจุดประสงค์อีกข้อในการแก้ปัญหาค่าเหมาะสุด คือ ต้องการให้ $\|x\|_1$ มีขนาดเล็กด้วย

จากลักษณะธรรมชาติของ นอร์ม-1 เป็นที่ทราบกันดีว่า หาก $\|x\|_1$ มีขนาดเล็ก หมายความว่าสมาชิกบางตัวของ $\|x\|_1$ มีค่าเป็นศูนย์ [1, บทที่ 7] การเลือกใช้พารามิเตอร์ลงโทษ (penalty parameter) λ ที่มีค่ามาก จะทำให้ผลเฉลยของ (3) มีลักษณะเบาบาง (sparse solution) แต่จะทำให้ความผิดพลาดจากการประมาณ (พจน์ $\|Ax - b\|_2$) สูงขึ้น ในขณะที่การเลือกใช้ λ ค่าน้อยๆ ทำให้ผลเฉลยมีลักษณะหนาแน่น แต่ก็จะทำให้ความผิดพลาดจากการประมาณต่ำลง

นอกจากนี้แล้ว เนื่องจากฟังก์ชันนอร์ม เป็นฟังก์ชันคอนเวกซ์ , ฟังก์ชันประกอบ ของฟังก์ชันคอนเวกซ์ และ การส่งสัมพรรค (affine transformation) เป็นฟังก์ชันคอนเวกซ์ ดังนั้น จะเห็นว่า ฟังก์ชันจุดประสงค์ของปัญหากำลังสองต่ำสุดเป็นฟังก์ชันคอนเวกซ์ และเนื่องจากฟังก์ชันลงโทษแบบนอร์ม-1 เป็นฟังก์ชันคอนเวกซ์ และ ผลบวกของฟังก์ชันคอนเวกซ์เป็นฟังก์ชันคอนเวกซ์ ดังนั้นปัญหาแบบ Lasso (3) จึงเป็นปัญหาคอนเวกซ์ ด้วยเหตุนี้ จึงสามารถใช้ขั้นตอนวิธีต่างๆ เช่น ขั้นตอนวิธีจุดภายใน (interior-point method) [1] มาใช้แก้ปัญหาคอนเวกซ์ดังกล่าว นอกจากนี้ เรายังสามารถใช้โปรแกรม CVX [13] ที่สามารถแปลงปัญหาคอนเวกซ์ที่มีขนาดเล็กถึงกลาง ให้กับตัวแก้ปัญหา (solver) ที่ใช้ขั้นตอนวิธีจุดภายในได้ วิธีดังกล่าว ในแต่ละรอบ เราจะเป็นต้องหาอนุพันธ์อันดับสองของฟังก์ชันจุดประสงค์ ซึ่งเป็นที่ทราบกันดีว่ามีความสิ้นเปลืองในการคำนวณมาก

ด้วยเหตุนี้ ในการแก้ปัญหาคอนเวกซ์ที่มีขนาดใหญ่ (จำนวนตัวแปรอาจอยู่ในหลักหมื่นหรือมากกว่านั้น) จึงมักใช้ขั้นตอนเกรเดียนต์ เนื่องจากเป็นวิธีที่มีค่าสิ้นเปลืองในการคำนวณค่อนข้างต่ำ ดังนั้นต่อไป เราจะกล่าวถึงขั้นตอนวิธีเกรเดียนต์แบบต่างๆ ที่จะนำมาใช้ในโครงงานนี้

วิธีเกรเดียนต์

วิธีเกรเดียนต์ เป็นวิธีทำซ้ำ (iterative method) แบบพื้นฐานที่สุดวิธีหนึ่งในการแก้ปัญหาการหาค่าเหมาะสุด ซึ่งมีรูปแบบ

$$\text{minimize} \quad f(x)$$

โดยที่ $x \in \mathbf{R}^n$ เป็นตัวแปร และ $f : \mathbf{R}^n \to \mathbf{R}$ เป็นฟังก์ชันซึ่งหาอนุพันธ์อันดับที่หนึ่งได้

หากกำหนดให้ k คือ ดัชนีการวนซ้ำ, x_k คือ จุดประมาณของจุดต่ำสุดของฟังก์ชัน f ในการวนซ้ำรอบที่ k, $\nabla f(x_k)$ คือ อนุพันธ์อันดับที่หนึ่ง ของ f ที่จุด x_k, และ t_k คือ ระยะขั้น (step length, step size) จะได้ว่าขั้นตอนวิธีของวิธีเกรเดียนต์เป็นดังต่อไปนี้

<u>กำหนดให้</u> จุด $x_0 \in \mathbf{R}^n$ เป็นจุดเริ่มต้นของการวนซ้ำ
<u>ทำซ้ำ</u>

1. คำนวณ $\nabla f(x_k)$
2. กำหนดค่า t_k
3. คำนวณจุด x ในรอบถัดไป จากสูตร

$$x_{k+1} = x_k - t_k \nabla f(x_k) \tag{4}$$

<u>จนกระทั่ง</u> จุดประมาณของจุดต่ำสุด หรือ ค่าประมาณของค่าต่ำสุด ของฟังก์ชัน f สอดคล้องกับเกณฑ์การหยุด (stopping criterion) ที่กำหนดไว้

ในการกำหนดค่า t_k เราสามารถทำได้หลายวิธี เช่น กำหนดค่า t_k ให้คงที่ตลอดการวนซ้ำ หรือ คำนวณจากการแก้ปัญหาค่าเหมาะสุดใน 1 มิติ เพื่อให้ได้ค่า t_k ซึ่งทำให้ค่าของฟังก์ชันในการวนซ้ำรอบถัดไป มีค่าลดลงมากที่สุด เป็นต้น

ข้อดีของวิธีเกรเดียนต์คือ ในแต่ละรอบของการวนซ้ำ จะมีค่าสิ้นเปลืองในการคำนวณค่อนข้างต่ำ อีกทั้งในการคำนวณทั้งหมด จะใช้เพียงอนุพันธ์อันดับหนึ่งของฟังก์ชันเท่านั้น แต่อย่างไรก็ตาม วิธีเกรเดียนต์ มีสมบัติการลู่เข้าซึ่งถือได้ว่าค่อนข้างช้า เนื่องจาก หากต้องการให้ค่าประมาณค่าต่ำสุดของฟังก์ชัน คลาดเคลื่อนจากค่าจริงไม่เกิน ε เราจะต้องการจำนวนรอบของการวนซ้ำอยู่ในอันดับของ $1/\varepsilon$ ($\mathcal{O}(1/\varepsilon)$) สำหรับฟังก์ชันที่มีอนุพันธ์ต่อเนื่องแบบลิพชิทซ์ [10] นั่นหมายความว่า หากเราต้องการให้ความแม่นยำในการประมาณค่าต่ำสุดของฟังก์ชัน อยู่ในอันดับ 10^{-4} เราจะต้องทำการวนซ้ำมากเป็นหลักหมื่นรอบ เพื่อให้ได้ความแม่นยำในระดับดังกล่าว หรืออีกนัยหนึ่ง สามารถกล่าวได้ว่า ขั้นตอนวิธีเกรเดียนต์มีการลู่เข้าอยู่ในอันดับของ $\mathcal{O}(1/k)$

นอกจากนี้ ข้อจำกัดที่สำคัญของวิธีเกรเดียนต์คือ สามารถใช้ได้กับเพียงปัญหาการหาค่าเหมาะสุด ซึ่งมีฟังก์ชันจุดประสงค์ที่สามารถหาอนุพันธ์ได้เท่านั้น และเนื่องจากปัญหาการประมาณที่ใช้ฟังก์ชันลงโทษแบบนอร์ม-1 มีฟังก์ชันจุดประสงค์ซึ่งไม่หาสามารถหาอนุพันธ์ได้ เราจึงไม่สามารถนำวิธีเกรเดียนต์มาใช้แก้ปัญหาดังกล่าวได้ ดังนั้น ในการแก้ปัญหาการประมาณที่ใช้ฟังก์ชันลงโทษแบบนอร์ม-1 หรือในการแก้ปัญหาคอนเวกซ์อื่นๆ ซึ่งมีฟังก์ชันจุดประสงค์ที่ไม่สามารถหาอนุพันธ์ได้ เราจึงต้องมีวิธีอื่นๆ ที่นำมาใช้แก้ปัญหาเหล่านั้น ดังจะได้กล่าวต่อไป

เกรเดียนต์ย่อย (subgradient)

เกรเดียนต์ย่อย เป็นแนวคิดหนึ่งที่ใช้รับมือกับปัญหาคอนเวกซ์ ซึ่งฟังก์ชันจุดประสงค์หาอนุพันธ์ไม่ได้ โดยเราจะกำหนดเวกเตอร์บางตัวขึ้นมาเพื่อใช้แทนเวกเตอร์เกรเดียนต์ ณ จุดซึ่งหาอนุพันธ์ไม่ได้ของฟังก์ชันคอนเวกซ์

พิจารณาฟังก์ชัน f กำหนดให้ $\mathbf{dom} f$ แทนโดเมนของฟังก์ชัน f จากเงื่อนไขอันดับที่หนึ่งของฟังก์ชันคอนเวกซ์ เราจะได้ว่า f เป็นฟังก์ชันคอนเวกซ์ ก็ต่อเมื่อ $\mathbf{dom} f$ เป็นเซตคอนเวกซ์ และ

$$f(y) \geq f(x) + \nabla f(x)^T (y - x)$$

สำหรับทุกจุด $x, y \in \mathbf{dom} f$ [1]

จากอสมการดังกล่าว เราจะนิยามเกรเดียนต์ย่อยในทำนองเดียวกัน โดยสำหรับเวกเตอร์ g ถ้าอสมการ

$$f(y) \geq f(x) + g^T (y - x)$$

เป็นจริงสำหรับทุกๆ $y \in \mathbf{dom} f$ เราจะกล่าวว่า g เป็นเกรเดียนต์ย่อยของฟังก์ชันคอนเวกซ์ f ณ จุด $x \in \mathbf{dom} f$

นอกจากนั้น ยังมีศัพท์อีกคำหนึ่งซึ่งเกี่ยวข้องกับเกรเดียนต์ย่อยโดยตรง คือ ผลต่างเชิงอนุพันธ์ย่อย (sub-differential) ใช้สัญลักษณ์ $\partial f(x)$ ซึ่งนิยามเป็น เซตของเกรเดียนต์ย่อยทุกตัวของฟังก์ชัน f ที่จุด x นั่นคือ

$$\partial f(x) = \{g | g^T(y-x) \leq f(y) - f(x) \quad \forall y \in \mathbf{dom} f\}$$

ยกตัวอย่างเช่น ฟังก์ชัน $f(x) = |x|$ จะมีผลต่างเชิงอนุพันธ์ย่อย คือ

$$\partial f(x) = \begin{cases} \text{sign}(x), & x \neq 0 \\ a \in [0,1], & x = 0 \end{cases}.$$

เป็นต้น

วิธีเกรเดียนต์ย่อย

วิธีเกรเดียนต์ย่อย เป็นวิธีทำซ้ำ ซึ่งสามารถใช้แก้ปัญหาคอนเวกซ์ซึ่งมีฟังก์ชันจุดประสงค์ที่ไม่สามารถหาอนุพันธ์ได้ โดยขั้นตอนวิธีเกรเดียนต์ย่อย จะแตกต่างกับขั้นตอนวิธีเกรเดียนต์เพียงการใช้เกรเดียนต์ย่อยของฟังก์ชัน f ที่จุด x_k แทนเกรเดียนต์ในสูตร (4) เพื่อคำนวณจุดประมาณจุดต่ำสุดของฟังก์ชันจุดต่อไป

สำหรับการกำหนดค่า t_k มีที่ใช้อยู่ 3 แบบ คือ

1. กำหนดค่า t_k ให้คงที่ตลอดการวนซ้ำ
2. กำหนดค่า t_k เพื่อให้ $\|t_k g_k\|_2$ มีค่าคงที่ในทุกรอบของการวนซ้ำ
3. กำหนดค่า t_k ให้ลดลงในทุกรอบของการวนซ้ำ โดยให้ $\lim_{k\to\infty} t_k = 0$ และ $\sum_{k=0}^{\infty} t_k = \infty$

ถึงแม้ว่าวิธีเกรเดียนต์ย่อย มีข้อดีที่สามารถใช้แก้ปัญหาคอนเวกซ์ ซึ่งมีฟังก์ชันจุดประสงค์ที่ไม่สามารถหาอนุพันธ์ได้ แต่ได้มีการพิสูจน์มาแล้ว ว่าวิธีเกรเดียนต์ย่อย มีสมบัติการลู่เข้าที่ช้ามากคือ $\mathcal{O}(1/\sqrt{k})$ เท่านั้น [11] นอกจากนั้น การเลือกระยะขั้น t_k มีเพียงแบบที่ 3 แบบเดียวเท่านั้นที่รับประกันว่าวิธีเกรเดียนต์ย่อยจะลู่เข้าสู่ค่าต่ำสุดอย่างแน่นอน

ตัวกระทำพร็อกซิมิตี้ (proximity operator)

ตัวกระทำพร็อกซิมิตี้สำหรับฟังก์ชันคอนเวกซ์ h นิยามโดย

$$\mathbf{prox}_h(x) = \underset{u}{\text{argmin}} \left(h(u) + \frac{1}{2}\|u - x\|_2^2 \right)$$

ยกตัวอย่างเช่น สำหรับฟังก์ชัน $h(x) = 0$ มีตัวกระทำพร็อกซิมิตี้คือ $\mathbf{prox}_h(x) = x$ และ สำหรับฟังก์ชัน $h(x) = t\|x\|_1$ มีตัวกระทำพร็อกซิมิตี้คือ

$$\mathbf{prox}_h(x)_i = \begin{cases} x_i - t & \text{if } x_i \geq t, \\ 0 & \text{if } -t \leq x_i \leq y, \\ x_i + t & \text{if } x_i \leq -t. \end{cases}$$

ซึ่งเป็นที่รู้จักกันในชื่อ ตัวกระทำ soft-thresholding [12] โดยตัวกระทำนี้ จะถูกนำไปใช้ในหลายขั้นตอนวิธี ที่เราจะศึกษาต่อไป

วิธีพร็อกซิมอลเกรเดียนต์

วิธีพร็อกซิมอลเกรเดียนต์ เป็นวิธีทำซ้ำอีกหนึ่งวิธีซึ่งสามารถนำมาใช้แก้ปัญหาคอนเวกซ์ ที่มีฟังก์ชันจุดประสงค์ซึ่งหาอนุพันธ์ไม่ได้ โดยวิธีนี้ ในแต่ละรอบของการวนซ้ำ จะคำนวณจุดประมาณจุดต่ำสุดของฟังก์ชันใหม่ โดยอาศัยตัวกระทำพร็อกซิมิตี้

หลักการของวิธีนี้คือ จะนำไปใช้กับปัญหาคอนเวกซ์ซึ่งมีรูปแบบคือ

$$\text{minimize } f(x) = g(x) + h(x) \tag{5}$$

โดยที่ $g(x)$ เป็นฟังก์ชันคอนเวกซ์ซึ่งหาอนุพันธ์ได้, $\mathbf{dom} g = \mathbf{R}^n$ และ $h(x)$ เป็นฟังก์ชันคอนเวกซ์ ซึ่งสามารถคำนวณตัวกระทำพร็อกซิมิตี้ได้ง่าย

สำหรับขั้นตอนวิธีในการหาค่าต่ำสุดของฟังก์ชัน f จะมีลักษณะเช่นเดียวกันกับวิธีที่ได้กล่าวในข้างต้น เพียงแต่ ในแต่ละรอบของการวนซ้ำ จะคำนวณจุดประมาณจุดต่ำสุดของฟังก์ชันใหม่ จากสูตร

$$x_{k+1} = \mathbf{prox}_{t_k h}\big(x_k - t_k \nabla g(x_k)\big) \tag{6}$$

ถ้าหาก ∇g เป็นฟังก์ชันต่อเนื่องแบบลิพชิทซ์ ด้วยค่าคงที่ L นั่นคือ

$$\|\nabla g(x) - \nabla g(y)\|_2 \leq L\|x - y\|_2$$

สำหรับทุกจุด x, y แล้ว วิธีพร็อกซิมอลเกรเดียนต์ซึ่งใช้ $t_k = 1/L$ จะลู่เข้าด้วยอัตรา $\mathcal{O}(1/k)$

ขั้นตอนวิธี FISTA (Fast Iterative Shrinkage-Thresholding Algorithm)

ขั้นตอนวิธี FISTA [3] เป็นอีกหนึ่งขั้นตอนวิธีสำหรับแก้ปัญหา (5) โดยขั้นตอนวิธีนี้ ปรับปรุงมาจากขั้นตอนวิธีพร็อกซิมอลเกรเดียนต์ เพื่อให้มีสมบัติการลู่เข้าที่ดีขึ้น ในขณะเดียวกันก็ยังสามารถคงความง่าย แบบขั้นตอนวิธีพร็อกซิมอลเกรเดียนต์ไว้

แนวคิดสำหรับขั้นตอนวิธี FISTA คือ ในการคำนวณจุดประมาณจุดต่ำสุดของฟังก์ชันใหม่ จากสมการ (6) จะเปลี่ยนจากการใช้ฟังก์ชันของ x_k เป็นการใช้ฟังก์ชันของ y_k แทน โดยที่ y_k คือ จุดซึ่งเป็นผลรวมเชิงเส้นของ x_{k-1} และ x_{k-2} ซึ่งมีสูตรคำนวณได้หลายแบบ ตัวอย่างเช่น

$$y_k = x_{k-1} + \frac{k-2}{k+1}(x_{k-1} - x_{k-2})$$

เป็นต้น

สำหรับขั้นตอนวิธี FISTA ซึ่งใช้ $t_k = 1/L$ คงที่ โดยที่ L เป็น ค่าคงที่ลิพชิทซ์ของ ∇g จะมีอัตราการลู่เข้าเป็น $\mathcal{O}(1/k^2)$ [3] ซึ่งเร็วกว่าขั้นตอนวิธีพร็อกซิมอลเกรเดียนต์ ซึ่งมีอัตราการลู่เข้าอยู่ที่ $\mathcal{O}(1/k)$

3 รายละเอียดของข้อเสนอโครงงาน

3.1 วัตถุประสงค์

ศึกษา เปรียบเทียบขั้นตอนวิธีต่างๆ สำหรับการแก้ปัญหาการประมาณที่ใช้ฟังก์ชันลงโทษแบบนอร์ม-1 คือ ขั้นตอนวิธี ADMM, ขั้นตอนวิธี FISTA, ขั้นตอนวิธีสำหรับปัญหา Group Lasso และ เขียนชุดคำสั่ง MATLAB เพื่อนำขั้นตอนวิธีที่มีประสิทธิภาพที่สุด ตามเกณฑ์ที่กำหนด ไปแก้ปัญหาการหาแบบจำลองอนุกรมเวลา fMRI

3.2 ขั้นตอนการทำงาน

1. ศึกษารูปแบบของปัญหาการประมาณที่ใช้ฟังก์ชันลงโทษแบบนอร์ม-1
2. ศึกษาขั้นตอนวิธีต่างๆ ที่จะนำมาใช้แก้ปัญหาการประมาณที่ใช้ฟังก์ชันลงโทษแบบนอร์ม-1 ได้แก่ ขั้นตอนวิธี ADMM, ขั้นตอนวิธี FISTA, ขั้นตอนวิธีสำหรับปัญหา Group Lasso
3. เขียนชุดคำสั่ง MATLAB ของขั้นตอนวิธีต่างๆที่ได้ศึกษามา
4. ตั้งเกณฑ์สำหรับการเปรียบเทียบประสิทธิภาพในการแก้ปัญหา ของขั้นตอนวิธีที่ได้ศึกษามา
5. เปรียบเทียบประสิทธิภาพของชุดคำสั่งของขั้นตอนวิธีต่างๆ
6. ประยุกต์ใช้ชุดคำสั่งของขั้นตอนวิธีที่เหมาะสม ในการหาแบบจำลองอนุกรมเวลา fMRI ซึ่งเป็นปัญหาที่มีขนาดใหญ่

3.3 ผลลัพธ์ที่คาดหวังจากโครงงาน

ได้ชุดคำสั่ง MATLAB ที่มีประสิทธิภาพในการนำไปแก้ปัญหาการประมาณซึ่งมีขนาดใหญ่ได้ ด้วยทรัพยากรที่เหมาะสม

3.4 ภาพรวมของโครงงาน

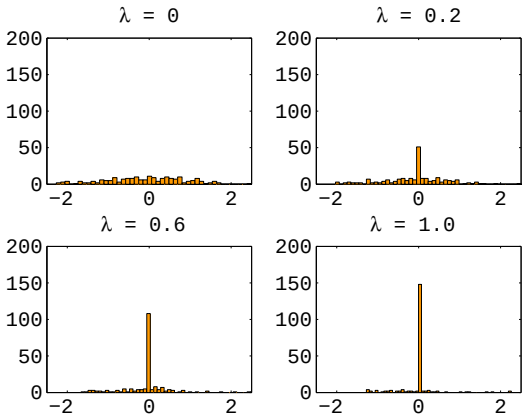
โครงงานนี้ จะศึกษาเกี่ยวกับขั้นตอนวิธีสำหรับการแก้ปัญหาการประมาณที่มีขนาดใหญ่ โดยเน้นปัญหาที่ใช้ฟังก์ชันลงโทษแบบนอร์ม-1 เนื่องจากการใช้ฟังก์ชันลงโทษแบบนอร์ม-1 ดังกล่าวในปัญหาการหาแบบจำลองจะทำให้ตัวแปรของแบบจำลองจำนวนมาก มีค่าเป็นศูนย์ ซึ่งนับว่าเป็นข้อดีในแง่ที่สามารถลดความซับซ้อนของแบบจำลองลงได้ โดยเฉพาะกับระบบที่มีขนาดใหญ่

จากนั้นจะเขียนชุดคำสั่งของขั้นตอนวิธีต่างๆที่ศึกษา และ นำมาเปรียบเทียบประสิทธิภาพในแง่ต่างๆ แล้วจึงนำขั้นตอนวิธีที่เห็นว่ามีประสิทธิภาพที่สุด ไปประยุกต์ใช้ในการหาแบบจำลองระบบ จากข้อมูลจริง

3.5 สรุปเนื้อหาของส่วนที่ได้ทำไปแล้วในภาคการศึกษาต้น

1. ศึกษาทฤษฎีพื้นฐานที่เกี่ยวข้องกับโครงงาน เช่น วิธีกำลังสองต่ำสุด, แบบจำลองถดถอยตัวเอง, ฟังก์ชันคอนเวกซ์, การหาค่าเหมาะสุดสำหรับปัญหาคอนเวกซ์ เป็นต้น
2. ศึกษาหลักการ สมบัติของขั้นตอนวิธีต่างๆที่นิยมใช้แก้ปัญหาการหาค่าเหมาะสุด เช่น ADMM (Alternating Direction Method of Multipliers), FISTA (Fast Iterative Shrinkage-Thresholding Algorithm) เป็นต้น จากงานวิจัยต่างๆที่ผ่านมา
3. ศึกษาเพิ่มเติม เกี่ยวกับ ปัญหาทางด้าน Machine Learning จากหลักสูตรบนเว็บไซต์ http://www.coursera.org
4. ศึกษาการใช้โปแกรม CVX [13] ในการแก้ปัญหาคอนเวกซ์ขนาดเล็ก และ ขนาดกลาง
5. จำลองผลจากการใช้ฟังก์ชันลงโทษแบบนอร์ม-1 ในปัญหาแบบ Lasso (3) โดยเปรียบเทียบทั้งในกรณีที่ไม่ใช้ฟังก์ชันลงโทษ และ กรณีที่ใช้ฟังก์ชันลงโทษโดยที่พารามิเตอร์ลงโทษ λ มีค่าต่างๆกัน

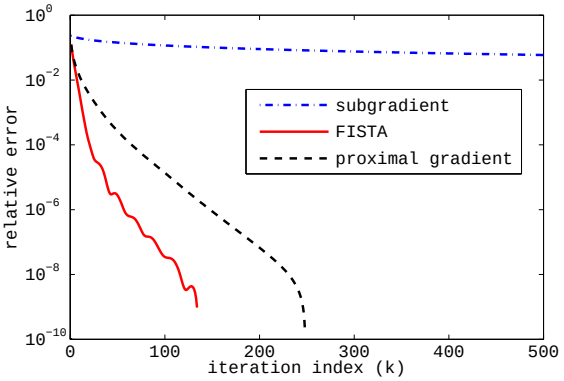
ในการจำลองปัญหา ใช้พารามิเตอร์ต่างๆ คือ $A \in \mathbf{R}^{500\times 200}$ เป็นเมทริกซ์ที่สุ่มขึ้นมา จากนั้นทำให้แต่ละหลักของ A เป็นปรกติ (normalization) คือ นอร์ม-2 ของแต่ละหลักของ A จะมีค่าเป็น 1 และ สุ่ม $x \in \mathbf{R}^{200}$ เป็นสัญญาณเข้าของระบบ จากนั้นคำนวณ $b \in \mathbf{R}^{500}$ โดยสมมติให้สัญญาณออก คือ Ax ถูกรบกวนด้วยสัญญาณรบกวนซึ่งมี ค่าเฉลี่ยเป็น 0 และมีความแปรปรวนเป็น 0.01 สุดท้ายจะแก้ปัญหาคอนเวกซ์ดังกล่าว ผ่านการใช้โปรแกรม CVX ซึ่งได้ฮิสโทแกรมของ x ดังกราฟต่อไปนี้



จากฮิสโทแกรมด้านบน จะเห็นได้ว่า หากเราไม่ใช้ฟังก์ชันลงโทษ (ให้ $\lambda = 0$) จะทำให้ค่าแต่ละตำแหน่งของ x กระจายตัวกันอย่างสม่ำเสมอ (x หนาแน่น) ต่างกับกรณีที่มีฟังก์ชันลงโทษ และ จำนวนสมาชิกที่เป็น 0 ใน x มีค่าเพิ่มขึ้น ตามค่าของพารามิเตอร์ลงโทษที่เพิ่มขึ้นด้วย (x เบาบาง)

6. เขียนชุดคำสั่ง MATLAB สำหรับการแก้ปัญหา Lasso (3) โดยใช้ขั้นตอนวิธีเกรเดียนต์ย่อย ขั้นตอนวิธีพร้อกซิมอลเกรเดียนต์ ขั้นตอนวิธี FISTA และ เปรียบเทียบสมบัติการลู่เข้าของทั้งสามขั้นตอนวิธี

   จากการจำลองปัญหาขนาดเล็ก โดยสุ่ม $A \in \mathbb{R}^{500\times1500}$ และ $b \in \mathbb{R}^{500}$ ขึ้นมา กำหนดให้ $\lambda = 1$ จากนั้นใช้โปรแกรม CVX เพื่อคำนวณ หา f^* (ค่าต่ำสุดของปัญหา Lasso) ไว้ล่วงหน้า และ สมมติให้ค่าที่คำนวณได้ เป็นค่าที่ถูกต้อง แล้วจึงใช้ชุดคำสั่งที่เขียนขึ้นมา แก้ปัญหาดังกล่าว พบว่าได้กราฟการลู่เข้าของทั้งสองขั้นตอนวิธีดังต่อไปนี้



โดยที่

$$\text{relative error} = \frac{f_k - f^*}{f^*}$$

จะเห็นว่า ทั้งสามวิธีซึ่งมีค่าสิ้นเปลืองในการคำนวณค่อนข้างต่ำ จะมีสมบัติการลู่เข้าเป็นไปตามทฤษฎี กล่าวคือ ขั้นตอนวิธี FISTA มีอัตราการลู่เข้าเร็วที่สุด ด้วยเหตุนี้ จึงเลือกขั้นตอนวิธี FISTA เป็นขั้นตอนวิธีหนึ่งที่จะนำมาศึกษา และ เปรียบเทียบกับขั้นตอนวิธีที่มีประสิทธิภาพอื่นๆ ต่อไป

3.6 อธิบายส่วนของงานที่จะทำต่อไปในภาคการศึกษาปลาย

1. ศึกษารายละเอียดของขั้นตอนวิธีต่างๆในการแก้ปัญหาคอนเวกซ์ เพิ่มเติม
2. เขียนชุดคำสั่ง MATLAB ของขั้นตอนวิธีต่างๆที่ศึกษามา
3. ตั้งเกณฑ์ที่จะใช้ในการเปรียบเทียบประสิทธิภาพของแต่ละขั้นตอนวิธี
4. นำชุดคำสั่งที่มีประสิทธิภาพที่สุด จากการเปรียบเทียบตามเกณฑ์ ไปใช้ในการหาแบบจำลองอนุกรมเวลา fMRI ซึ่งเป็นปัญหาที่มีขนาดใหญ่

4 บทสรุป

ในภาคการศึกษานี้ ได้ศึกษาความรู้ ทฤษฎีพื้นฐานต่างๆ ที่จำเป็น เกี่ยวกับขั้นตอนวิธีต่างๆ ที่จะนำมาแก้ปัญหาการประมาณที่ใช้ฟังก์ชันลงโทษแบบนอร์ม-1 รวมทั้งศึกษารายละเอียดเบื้องต้นเกี่ยวกับขั้นตอนวิธีนั้นๆ เพื่อเตรียมตัวเขียนชุดคำสั่ง MATLAB เพื่อการแก้ปัญหาการประมาณสำหรับระบบขนาดใหญ่ ได้อย่างมีประสิทธิภาพ

5 เอกสารอ้างอิง

[1] S.P. Boyd and L.Vandenberghe. *Convex optimization*. Cambridge Univ Pr, 2004.

[2] S.Boyd, N.Parikh, E.Chu, B.Peleato, and J.Eckstein. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers, 2011.

[3] A.Beck and M.Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[4] R.Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[5] J.Friedman, T.Hastie, and R.Tibshirani. *The Elements of Statistical Learning; Data Mining, Inference, and Prediction*. Springer-Verlag, 2001.

[6] S.Ghosh and Y.Rudy. Application of l1-norm regularization to epicardial potential solution of the inverse electrocardiography problem. *Annals of biomedical engineering*, 37(5):902–912, 2009.

[7] A.Y. Ng. Feature selection, l_1 vs. l_2 regularization, and rotational invariance. In *Proceedings of the twenty-first International Conference on Machine Learning*, page78. ACM, 2004.

[8] M.Schmidt, G.Fung, and R.Rosales. Fast optimization methods for ell_1 regularization: A comparative study and two new approaches. *Machine Learning: ECML 2007*, pages 286–297, 2007.

[9] E.K.P. Chong and S.H. Żak. *An introduction to optimization*. Wiley-interscience, 2004.

[10] M.Grant and S.Boyd. CVX: Matlab software for disciplined convex programming (web page and software). `http://stanford.edu/~boyd/cvx`, August 2008.

[11] D.P. Bertsekas. *Nonlinear programming*. Athena Scientific, 1999.

[12] S.P. Boyd and L.Vandenberghe. 5. subgradient method. `http://www.ee.ucla.edu/vandenbe/236C/lectures/sgmethod.pdf`, 2011.

[13] D.L. Donoho. De-noising by soft-thresholding. *Information Theory, IEEE Transactions on*, 41(3): 613–627, 1995.