המאמר היומי של מייק: 24.05.25  
rStar-Math: Small LLMs Can Master Math Reasoning with Self-Evolved Deep Thinking

כמה ימים לא סקרתי מאמר אבל ביום הולדתי לא יכולתי לא לכתוב סקירה למרות העומס המטורף. היום אסקור מאמר די מעניין שיצא לפני 4 חודשים והוא משלב פיין טיון של מודל שפה למשימות מתמטיות באמצעות MCTS שזה קיצור של Monte Carlo Tree Search. רובכם כנראה מכירים את MCTS מהפרויקטים המפורסמים AlphaGo ו- AlphaZero של דיפמיינד של אימנו מודלים המשחק Go. אציין AlphaZero למד לשחק רק דרך המשחקים עם עצמו ללא שום ידע מוקדם על Go. המודלים שפותחו היו כה חזקים שאלוף העולם ב-Go פרש בעקבות אחד מהם (לא זוכר איזה). הרעיון המתמטי מאחורי פיתוח מודלים אלו היה MCTS.

אלגוריתם MCTS הוא אלגוריתם חיפוש המשמש בעיקר במשחקים לקבלת החלטות אופטימליות. הוא בונה עץ החלטות על ידי הרצת דגימות אקראיות (סימולציות) רבות של מהלכים אפשריים מהמצב הנוכחי, ומעריך את איכותם. לאחר מכן, הוא בוחר את המהלך שמניב את התוצאות הטובות ביותר בממוצע לאורך הסימולציות. האלגוריתם מאזן בצורה חכמה בין חקירת מהלכים חדשים (exploration) העשויים להתגלות כיעילים, לבין ניצול מהלכים שכבר נמצאו כמוצלחים (exploitation) בסימולציות קודמות(כלומר מובילים לרוב לניצחון במשחק).

תהליך זה חוזר על עצמו, כאשר בכל איטרציה העץ מורחב והערכות איכות המהלכים מתעדכנות, עד שמתקבלת החלטה סופית. ארבעת השלבים המרכזיים בכל איטרציה הם: בחירת צומת הבא (selection), הרחבת העץ (expansion), סימולציה של המשחק (simulation), ועדכון ערכי הצמתים עד שורש העץ (backpropagation). הצלחת האלגוריתם נובעת מיכולתו להתמקד באזורים מבטיחים יותר בעץ החיפוש, גם במרחבי חיפוש עצומים. בסוף המודל, בהינתן מהלכי משחק נתונים, (מסלול בעץ) בוחר צומת בעל הסיכוי הגבוה ביותר לניצחון.

אבל איך זה קשור למודל שפה. התשובה היא אוטורגרסיביות. גם במודלי שפה אנחנו כרגע חוזים טוקן לאחר טוקן כמו במשחק גו. בעצם הרעיון הגדול בשימוש ב-MCTS לאימון של מודל שפה היא בניית דאטהסטים באיכות גבוה באמצעות חיפושם בעץ החלטות. אבל להבדיל מעץ החלטות שבו הצמתים הם מהלכי משחק כאן כל צומת הוא שלב בתהליך reasoning (הנמקה של המודל). לאחר מכן משתמשים בדאטהסט זה, בעל איכות גבוהה, כדי לעשות SFT למודל. אז השאלה כאן איך לדגום פתרונות נכונים ומגוונים עם גישה זו?

כאמור המאמר מציע מבוססת MCTS לאימון מודל לפתרון בעיות מתמטיות כאשר יש לנו פונקציית תגמול ברורה (האם הפתרון נכון או לא) בסוף הגנרוט. לעומת זאת התגמול (reward) באמצע שרשרת ההנמקה הוא משהו ברור (ד״א יש ב-PPO את אותה הבעיה - יש לנו פונקציית reward שאימנו אולם היא נותנת ציון לכל הפתרון ולא לחלקו ואז אנו מאמנים פונקציית value המשערכת את התגמול בשלבי ביניים - דרך פתרון בעיית רגרסיה). ב-MCTS בניית פונקציה המקנה ציון לצומת (פתרון חלקי) הוא קריטי כי אחרת לא נצליח לבנות את עץ פתרונות בצורה טובה (כלומר מניבה פתרונות טובים לבעיות מתמטיות). כמובן כל צומת בעץ נבנה על ידי דגימה ממודל שפה.

בהתחלה ציון הצומת (= פתרון חלקי עד שלב מסוים) נבנה באמצעות שכיחות הופעתו בפתרונות נכונים של הבעיה. ככל הוא מופיע יותר בשרשראות הנמקה המובילות לפתרון נכון, ציון שלו גבוה יותר. בשלבים מאוחר יותר (כאשר פונקציית ציון מצייצבת) המאמר עושים משהו דומה לאימון מודל תגמול באימון RLHF של מודלי שפה. בכל עומק (שכבה) של עץ לוקחים צמתים בעלי ציונים הגבוהים והנמוכים ביותר ומאמנים מודל ציון צומת בסגנון Bradley-Terry (כמו שמקובל ב-RLHF סטנדרטי). כאמור פונקציית ציון משמשת אותנו לבחירה מאיזה צומת לדגום שלב הבא לפתרון באמצעות אלגוריתם די סטנדרטי (Upper Confidence bounds for Trees (UCT המג'נגל בין exploration vs exploitation.

כדי להגיע לפתרונות יותר איכותיים יותר מהר המודל מתבקש לממש כל שלב בשרשרת הנמקה בפייטון ואם קוד זה לא עובר טסטים, הצומת נפסל. המאמר מתחיל ממודל שפה קטן, יוצר עץ פתרונות (עם כל השלבים שתיארתי), בוחר פתרונות הכי איכותיים (בעלי ציוני הגבוהים ביותר), מצבע SFT על המודל וחוזר על זה עוד פעם. וכתוצאה מכך אנו מקבלים מודל קטן וחמוד אבל מסוגל לפתור בעיות מתמטיות די מורכבות (לכאורה).

https://arxiv.org/abs/2501.04519