המאמר היומי של מייק - 13.01.25  
Improve Mathematical Reasoning in Language Models by Automated Process Supervision

מזמן רציתי לכתוב סקירה על MCTS שזה Markov Chain Tree Search ולגמרי במקרה נתקלתי במאמר הזה המציע ליישם את השיטה המגניבה הזו עבור אימון LLMs. הפעם המטרה לאמן מודל שפה לפתור בעיות מתמטיות (לוגיות) מורכבת שפתרונם מכיל שלבים רבים.

קודם כל הסבר קצר מה זה בעצם MCTS. חיפוש עץ מונטה קרלו (MCTS) הוא אלגוריתם לאופטימיזציה של פוליסי עבור תהליכי החלטה מרקוביים (Markov Decision Process) בעלי אופק סופי וגודל סופי, המבוסס על דגימת אפיזודות אקראיות המאורגנות באמצעות עץ החלטה.

י. הוא עובד 4 שלבים:

בחירה: בוחרים מסלול מהשורש לעלה לפי פוליסי חקירה/ניצול (exploration/exploitation)

הרחבה: מוסיפים מצב חדש לעץ

סימולציה: מריצים סימולציה אקראית מהמצב החדש עד סוף המשחק

עדכון לאחור: מעדכנים את הערכים בכל הצמתים במסלול שנבחר

אנו משתמשים ב-MCTS כדי לשפר את המדיניות (policy) על ידי בחירת פעולות טובות יותר. המודל מספק הערכות למצבים במקום סימולציות אקראיות ו-MCTS משתמש בהערכות אלו כדי לבנות עץ חיפוש יעיל יותר. לדוגמה, AlphaGo משתמש ב-MCTS בשילוב עם רשתות עמוקות כדי לבחור מהלכים. היתרון העיקרי של MCTS הוא בין חקירת מצבים חדשים (exploration) לבין ניצול ידע קיים(exploitation), ומשפר את קבלת ההחלטות לאורך זמן.

המאמר שנסקור היום מציע להשתמש בגישת MCTS כדי לאמן מודל שפה לבנות תשובות בעלות שלבים רבים וכמו שאתם יכולים לנחש הצמתים בגרף הזה יהיו השלבים בפתרון. המאמר מציין פתרונות SOTA לאימון מודלי שפה לפתור בעיות אלו מתחלקים לשני סוגים. הראשון מסמלץ את כל שלבי הפתרון כך שהמודל מאומן (עם טכניקות RLHF לבחירתכם) למקסם את הפרס שהמודל מקבל בסוף (בד״כ בינארי, כלומר האם הפתרון נכון/לא נכון) עם איזשהו איבר רגולריזציה (קירבה למודל המקורי).

השיטה השנייה PRM עושה דבר דומה אבל למסלולים חלקיים (=כמה שלבי פתרון בהתחלה). ניתן לראות שהגישה הראשונה תעבוד פחות טוב עבור בעיות עם הרבה שלבים כי ה-reward מאוד דליל (sparse) וקשה לאופטימיזציה. המקרה השני צריך הרבה דאטה מתויג איכותי וזה מאוד יקר.

המאמר כאמור מציע להשתמש ב-MCTS למטרה זו. כמו שמקובל ב-MDP אנו צריכים להגדיר מה זה המצב, פעולה ותגמול. המצב s מוגדר בתור שלאה q, כל שלבי הפתרון עד עכשיו (לא חייב לכלול את הפתרון) והפעולה a היא בחירת הצומת הבאה שבמקרה הזה הוא שלב הבא של פתרון שאלה q. לאחר שהפעולה a נבחרת היא מתווספת ל-s כלומר המצב החדש הוא (s\_old, a). הפעולה a נבחרת על ידי פוליסי (p(a|s כאשר עבור MCTS הוא מורכב משני מחוברים: הראשונה (exploitation) נוטה לבחור צמתים בעלי תגמול גבוה והאיבר השני (exploration) מעדיף צמתים שלא ביקרנו בהם הרבה.

עכשיו הגיע הזמן לדבר עם התגמול (reward). עבור צומת נותן v התגמול שלו הוא אחוז ה-rollouts הנכונים(המסומן בתור c) שהתחילו משלב v (אחוז המסלולים בגרף שהגיע לפתרון הנכון החל מ v). דרך אגב יש שיטה מאוד אינטואיטיבית לזיהוי של הטעות הראשונה בפתרון לא נכון (שכמה מעבודות קודמות מצאו כמידע יעיל לאימון מודל) שמאפשרת לזהות צמתים ״לא נכונים בהחלט״ (שמהם לא ניתן להגיע לפתרון הנכון) בפתרון שנקראת ״חיפוש בינארי.

השיטה כל פעם מחלקת את מסלול הפתרון לשניים ובודקת היום c עבור הצומת שנמצא בחצי המסלול גדול או קטן מ-0. אם הוא שווה לאפס אז הטעות כנראה בחצי הראשון ואם הוא גדול מ-0 אז הטעות כנראה בחצי השני. אז שוב מחלקים לחצי את החצי שבו אנו חושדים שיש טעות וממשיכים לצמצם את החיפוש עד שמגיע ל״צומת המטעה״.

כדי להגדיל את מספר הדוגמאות המחברים מציעים לאחסן rollouts של הפתרון ולבצע חיפוש בינארי של הצומת שבו (ככל הנראה) קרתה טעות ולהתחיל ממנה חיפוש חדש. זה מאפשר לבנות דוגמאות עם אותם השלבים ההתחלתיים והמשך שונה. אזכיר שעם גישת PRM (שעליה המאמר בונה את הפתרון) כל דוגמא היא השלישיה של שאלה, פתרון חלקי, וציון האם זה נכון. כל אלו אנו מקבלים בתהליך המתואר כאן.

לבסוף המאמר משתמש ב-MCTS עם פוליסי Q כאשר המצב של כל צומת בגרף הפתרון מתואר על ידי שלישיה (אחרת) שהיא מספר הפעמים שהפתרון ביקר בצומת הזה, אחוז הפתרונות הנכונים c מהצומת הזו (כלומר שערוך מונטה קרלו שלו) וגם ערך של פוליסי Q שהוא מקבל ערך גבוה עבור ערך של C קרוב ל 1(צומת מוביל לרוב לפתרון הנכון) ויש לו איבר רגולריזציה (כפלי) הקונס אותו על פתרונות ארוכים יותר. בחירה של מסלול rollout נבחר על ידי דגימה שנבנית בהתבסס על הסטטיסטיקה של העץ עם האלגוריתם שנקרא PUCT (נוסחה 3 במאמר). כמובן Q, c וסטטיסטיקה של העץ מתעדכנות במהלך MCTS.

זהו זה - סקירה מאוד ארוכה, מקווה שהצלחתי להסביר אותו, מאמר לא טריוויאלי…

https://arxiv.org/abs/2406.06592