**המאמר היומי של מייק: 17.09.25**

**Bootstrapping Task Spaces for Self-Improvementֿ**

מצאתי עוד כמה סקירות שפספסתי לפני אז היום זה סקירה 511…

איך טכניקת למידה עם חיזוק חדשה (RL) מאמנת LLMs להשתפר בחשיבה, על ידי הפיכת ניסיונות העבר שלהם לשיעור הבא? כולנו היינו שם. בוהים בבעיה מורכבת כמו הוכחה מתמטית מסובכת, קוד מסובך, פסקה שקשה לנסח. הניסיון הראשון כמעט אף פעם אינו מושלם. ההתקדמות האמיתית מגיעה מהמעגל של כתיבת טיוטה, לקיחת צעד אחורה, איתור הפגמים ותיקון. התהליך האיטרטיבי הזה, היכולת ל*שיפור עצמי*, הוא סימן היכר של פתרון בעיות אינטליגנטי.

במשך שנים, מטרה מרכזית בבינה המלאכותית הייתה להטמיע במכונות את אותה היכולת. באמצעות LLMs, אנו יכולים כעת לבקש מהם "לאמת את עבודתם" או "לנסות שוב", אך כיצד אנו מאמנים אותם להפוך ל*טובים יותר באופן שיטתי* בתהליך זה? הגישה הנאיבית היא שימוש בכוח גס: לתת למודל לייצר פתרון בן 10 שלבים, לבדוק אם הוא נכון, ולתת לו תגמול. גישה זו אינה כה יעילה. זה כמו להגיד לסטודנט לכתוב חיבור שלם, לתת ציון רק על התוצר הסופי, ולא להציע שום משוב על הפסקאות הבודדות. זה יקר, איטי, וחלק גדול מאות הלמידה הולך לאיבוד.

מאמר חדש מבית Meta Superintelligence, מציג פרדיגמה אלגנטית וחזקה יותר בשם (Exploratory Iteration - EXIT). זוהי שיטת RL הממסגרת מחדש את הבעיה. במקום ללמד LLM לבצע שרשראות שיפור ארוכות ומרובות שלבים, היא מאמנת את המודל על האיטרציות (שלבים בשרשרת החשיבה) האינפורמטיביות ביותר, ובכך יוצרת ",תכנית לימוד״ דינמית שבמסגרתה המודל משפר את יכולותיו. זהו מפגש יפהפה בין למידת curriculum, חקירה, והיכולות הייחודיות של LLMs.

בואו נהיה קצת יותר טכניים. הדרך הסטנדרטית לאמן סוכן באמצעות RL היא לגרום לו להשלים משימה שלמה ("אפיזודה") ואז לעדכן את המדיניות (policy) שלו בהתבסס על התגמול הסופי. אם המשימה היא "שפר את הפתרון הזה ב-K שלבים", סוכן RL סטנדרטי יצטרך לבצע את כל K השלבים לפני שיקבל אות משמעותי.

דבר זה מציג 3 אתגרים מרכזיים:

1. עומק שרירותי: מי מחליט ש-K הוא 5, 10 או 20? המספר האופטימלי של שלבי שיפור תלוי במשימה ואינו ידוע מראש. קביעת ערך קבוע היא שרירותית ומגבילה.
2. קושי בשיוך קרדיט (Vanishing Credit): ככל שהחלטה מכרעת התקבלה בשלב מוקדם יותר בשרשרת השיפורים, כך קשה יותר לאלגוריתם הלמידה לשייך לה קרדיט או אשמה בצורה נכונה. מהלך שגוי אחד בשלב 2 עלול להרוס את כל התהליך בן 10 השלבים, אך האות מתעמעם לאורך המסלול המלא.
3. עלות חישובית: יצירת K גרסאות של פתרון יקרה פי K מיצירת גרסה אחת. הדבר מאט באופן דרמטי את לולאת האימון.

EXIT עוקפץ את כל הבעיות הללו על ידי פירוק הבעיה הארוכה והמורכבת של שיפור ב-K שלבים לסדרה של משימות שיפור פשוטות בנות שלב אחד. הרעיון הבסיסי מאחורי EXIT הוא די פשוט: כל פתרון ביניים שנוצר על ידי המודל יכול להיחשב כמכונה של משימה חדשה וייחודית. דמיינו את דף הטיוטה של תלמיד הפותר בעיה במתמטיקה. הבעיה המקורית נמצאת בראש הדף. מתחתיה נמצא הניסיון הראשון שלו. ומתחתיו, גרסה מתוקנת. EXIT מתייחס לכל שורה בדף הטיוטה הזה כנקודת התחלה פוטנציאלית לבעיה חדשה. הבעיה החדשה היא: "בהינתן *הניסיון הספציפי הזה*, האם תוכל לבצע שיפור יחיד ובעל ערך?"

גישה זו משנה את תהליך הלמידה. המודל כבר לא לומד רק לפתור את קבוצת הבעיות המתמטיות המקורית. הוא לומד מיומנות כללית הרבה יותר: כיצד לשפר פתרון מ*כל* מצב נתון. תהליך זה עושה bootstrap למרחב עצום ומגוון של משימות אימון מתוך קבוצה ראשונית קטנה בהרבה.

אז איך זה עובד בפועל? EXIT היא מערכת מתוחכמת עם כמה רכיבים רעיוניים מרכזיים. היא פועלת על גבי GRPO המפורסם, שהוא חשוב מסיבה מרכזית אחת: במקום ללמוד "פונקציית ערך" מורכבת כדי להעריך תגמולים, הוא מעריך את המדיניות על ידי יצירת *קבוצה* קטנה של פתרונות שונים לאותה בעיה והשוואת התוצאות שלהם. גישה מבוססת-קבוצה זו היא המרכיב הסודי של הקוריקולום של EXIT.

המערכת מתחזקת מאגר, זיכרון של כל נקודות ההתחלה ה"מעניינות" ביותר (כלומר, פתרונות קודמים) שבהן נתקלה. אבל איך היא מגדירה "מעניין"? כאן נכנסת האינטואיציה המתמטית. עבור כל נקודת התחלה, GRPO מייצר סט של שיפורים פוטנציאליים המורכבים משלב אחד ומקבל תגמול עבור כל אחד מהם. EXIT מחשבת את השוֹנוּת של תגמולים אלה ופועלת באופן הבא

* שונות נמוכה ותגמול נמוך: המודל נכשל מנקודת התחלה זו בכל ניסיון. זה קשה מדי; אין כאן גרדיאנט למידה.
* שונות נמוכה ותגמול גבוה: המודל מצליח מנקודת התחלה זו בכל פעם. המשימה נלמדה במלואה. אין צורך להמשיך לתרגל אותה.
* שונות גבוהה: המודל לפעמים מצליח ולפעמים נכשל. זהו הגבול של יכולת המודל – ה"אזור הרופף" שבו הוא הכי פתוח ללמידה.

EXIT נותנת עדיפות לדגימת נקודות התחלה אלו בעלות השונות הגבוהה מהמאגר שלה כדי להתאמן עליהן בשלב הבא. הדבר יוצר אוטו-קוריקולום טבעי ומתהווה. המודל מתמקד בנקודות בתהליך פתרון הבעיות שבהן הוא הכי לא בטוח. ככל שהוא שולט בשלבים אלה, השונות שלהם יורדת, והם מאבדים את עדיפותם, בעוד שלבים חדשים ומורכבים יותר עולים לראש רשימת העדיפויות.

EXIT כוללת שני מנגנונים חכמים כדי להבטיח שמרחב המשימות יישאר מגוון והמודל ימשיך לחקור באופן יצירתי:

1. התבדרות עצמית (Self-Divergence): בהסתברות מסוימת, במקום להתבקש "לשפר" את הפתרון האחרון שלו, המודל מקבל הנחיה "לשפר את הפתרון, אך בדרך שונה באופן משמעותי". הדבר מאלץ את המודל במפורש לקפוץ לחלק אחר במרחב הפתרונות, וליצור ענפי חקירה חדשים שאחרת היו עלולים להתעלם מהם.
2. בונוס גיוון כפלי (Multiplicative Diversity Bonus): זוהי דחיפה מתמטית עדינה יותר. פתרונות המודל ממופים לתוך מרחב אמבדינג שבו פתרונות דומים קרובים יותר זה לזה. עבור כל קבוצת הרצות (rollouts), המערכת מחשבת את "מרכז המסה" של הפתרונות כל פתרון ש*רחוק יותר* ממרכז זה נחשב למקורי יותר. האלגוריתם מעניק בונוס קל לפתרונות מתבדרים אלה, ובכך למעשה אומר לפוליסי: "כל הכבוד על ההצלחה, ונקודות נוספות על כך שעשית זאת בדרך לא שגרתית."

שני מנגנונים אלה מבטיחים שמאגר המשימות לא רק הופך עמוק יותר אלא גם רחב יותר, ומזריק באופן מתמיד חדשנות לתהליך האימון.

המאמר הוא יותר מסתם עוד מאמר על RL. הוא מצביע על עתיד שבו הגבולות בין דאטה, אימון והסקה (inference) מתחילים להיטשטש. EXIT מספקת דרך עקרונית לסוכן להפוך למורה של עצמו, לזהות את חולשותיו וליצור את הקוריקולום המדויק שהוא צריך כדי להשתפר. זהו רעיון רב עוצמה. במקום להסתמך על מאגרי דאטה סטטיים, מערכות AI עתידיות עשויות לחקור באופן רציף, וליצור אתגרים חדשים לעצמן מתוך מרקם החוויות שלהן. ה"משימה" כבר אינה ישות קבועה, אלא מרחב דינמי וגדל תמידית, שנוצר בשיתוף עם הסוכן הלומד עצמו.

https://www.arxiv.org/abs/2509.04575