

**המחלקה למתמטיקה ולמדעי המחשב**

סמינר בבינה מלאכותית

20372

מימוש למידה חישובית בסדר גודל גדול

**העבודה הוכנה על-ידי:** רן טיבי ת.ז. 303072078

**בהדרכתה של** ד"ר מיריי אביגל

יוני 2014

תוכן עניינים

[2 מבוא 4](#_Toc389421101)

[3 רשתות חברתית מקוונת 5](#_Toc389421102)

[3.1 מהי רשת חברתית מקוונת? 5](#_Toc389421103)

[3.2 הרשת החברתית טוויטר 5](#_Toc389421104)

[3.3 למידה חישובית ברשתות חברתיות מקוונות 6](#_Toc389421105)

[4 רקע ללמידה חישובית 7](#_Toc389421106)

[4.1 מהי למידה חישובית? 7](#_Toc389421107)

[4.2 מודלי למידה 7](#_Toc389421108)

[4.3 למידת אצווה 8](#_Toc389421109)

[4.3.1 רגרסיה ליניארית 8](#_Toc389421110)

[4.3.2 Gradient Descent 9](#_Toc389421111)

[4.4 למידה מקוונת 10](#_Toc389421112)

[5 בעיית הסיווג 11](#_Toc389421113)

[5.1 מהי בעיית הסיווג? 11](#_Toc389421114)

[5.2 רגרסיה לוגיסטית 11](#_Toc389421115)

[6 למידה חישובית בסדר גודל גדול 13](#_Toc389421116)

[6.1 למידת חישובית תוך שימוש ב-MapReduce 13](#_Toc389421117)

[6.2 Stochastic Gradient Descent 15](#_Toc389421118)

[6.3 שיטות הרכב (Ensemble Methods) 16](#_Toc389421119)

[7 מימוש למידת מכונה בסדר גודל גדול באמצעות Hadoop ו-Pig 18](#_Toc389421120)

[7.1 אודות Hadoop 18](#_Toc389421121)

[7.2 אודות Pig 18](#_Toc389421122)

[7.3 הארכיטקטורה של Pig 18](#_Toc389421123)

[7.4 איך לממש למידה חישובית ב-Pig 19](#_Toc389421124)

[7.4.1 חבילת למידה בשפת Java 19](#_Toc389421125)

[7.4.2 אימון המודל 20](#_Toc389421126)

[7.4.3 השימוש במודל הנלמד 21](#_Toc389421127)

[8 ניסוי ניתוח רגשי על כמות נתונים גדולה 22](#_Toc389421128)

[8.1 תיאור הבעיה 22](#_Toc389421129)

[8.2 אופן מימוש הפתרון 22](#_Toc389421130)

[8.3 תוצאות הניסוי 22](#_Toc389421131)

[9 מימוש לבעיית זיהוי משתמשים דומים 24](#_Toc389421132)

[9.1 תיאור הבעיה 24](#_Toc389421133)

[9.2 מימוש הפתרון 24](#_Toc389421134)

[9.3 תוצאות הניסוי 24](#_Toc389421135)

[10 סיכום 26](#_Toc389421136)

[11 ביבליוגרפיה 27](#_Toc389421137)

# מבוא

הרשתות החברתיות הפכו בשנים האחרונות לתחום המושך אליו יותר ויותר משתמשים. ייחודם של הרשתות החברתיות הוא בכך שהן מאפשרות לאדם לחלוק קשרים אמיתיים או וירטואליים עם אנשים שיש לו בסיס משותף איתם. התשתית אותה מספקות הרשתות החברתיות מאפשרת למפעיליהן לאגור מידע רב אודות המשתמשים, החל ממי הם חבריהם הקרובים, דרך מהם תחומי העניין שלהם, ועד לאופי השימוש שלהם ברשת האינטרנט ויותר.

כלל המידע מעורר עניין עסקי רב, ומעלה צורך בקרב המפעילות לייצר שיטות מתוחכמות כדי להפיק ממנו את המרב ולעודד את המשתמשים להוסיף ולעשות שימוש ברשתות. כתוצאה מכך, מתגבר גם העניין בשילוב טכניקות למידה חישובית בפתרונות השונים. לצורך כך יש לבצע התאמה של האלגוריתמים הקיימים היום בכדי שיוכלו לתת מענה לניתוח על סמך כמות גדולה כל כך של נתונים.

על אף שהתיאוריה מציגה פתרונות ללמידה חישובית בסדר גודל גדול, מעט מאוד ידוע על אופן המימוש ובפרט לא ידוע על best-practice בהקשר זה. מסמך זה סוקר מקרה מבחן שבוצע ברשת החברתית "טוויטר", אשר מציג דרכים ליישם למידה חישובית בסדר גדול תוך שימוש בחבילת הניתוח Pig המבוססת Hadoop.

המסמך נפתח בהסבר על מהי למידה חישובית, מציג רקע תיאורטי ומספר מודלים ידועים ללמידה חישובית. סקירה זו תאפשר גם לקורא חסר רקע לקבל הבנה כללית על מהי למידה חישובית, מה הרעיון שעומד מאחוריה, וכיצד ניתן ליישם אותה. מובן שאין מסמך זה מתיימר ללמד את הקורא את כל היסודות, אולם סבורני כי באמצעות השלמות קלות שהפניות אליהן מופיעות בגוף הטקסט יוכל קורא חסר רקע להבין את הנאמר.

לאחר מכן, יוצגו שיטות ופתרונות ללמידה חישובית בסדר גדול. ולאחריו יוצג מקרה מבחן של יישום למידה חישובית בסדר גודל גדול כפי שבוצע בטוויטר, וכן ניסוי שבוצע כדי לבחון מקרה מבחן זה.

לסיום, יוצג מימוש יישום קונקרטי שעושה שימוש בתשתית שפותחה במסגרת מקרה המבחן ונמצא כיום בשימוש ברשת טוויטר.

# רשתות חברתית מקוונת

## מהי רשת חברתית מקוונת?

אנו מגדירים רשת חברתית מקוונת כשירות מבוסס אינטרנט המאפשר למשתמשים ליצור עמוד פרופיל ציבורי או ציבורי למחצה במערכת, ליצור רשימה של אנשים שאיתם הם חולקים קשר, ולצפות בפרופיל של האנשים איתם הם מקושרים [[12](#Ell07)]. המקור לקשר, אופי הפרופיל, המידע שהוא מכיל, וכן קהל היעד של הרשת החברתית יכולים להשתנות מאתר לאתר. לדוגמא, הרשת החברתית [[1]](#footnote-1)MySpace הינה רשת חברתית המאפשרת לכל גולש לפתוח חשבון משתמש, בעוד שחשבונות המשתמש של זמרים שונים מהחשבונות הרגילים בכך שהאמנים יכולים להעלות הקלטות של השירים שלהם לעמוד הפרופיל. הרשת החברתית [[2]](#footnote-2)LinkedIn לעומת זאת, היא רשת חברתית שנועדה ליצור קשרים מקצועיים ועסקיים בין חבריה, ועמודי הפרופיל של המשתמשים בה מכילים את הניסיון התעסוקתי והידע המקצועי שלהם.

הרשת החברתית מאפשרת למשתמשים בה להיות בקשר עם המעגל החברתי הרחב שלהם, לעיתים היא אף מאפשרת להם להכיר אנשים חדשים, לחפש משתמשים בעלי תחומי עניין משותפים, או ליצור קהילה סביב נושא משותף. שירותים חברתיים מבוססי רשת מאפשרים לקשר בין אנשים החולקים תחומי עניין ופעילות ללא מגבלות של מרחק גיאוגרפית, זמן או מחיר כלכלי. אתרים רבים מאפשרים למשתמש לפרסם בעמוד הפרופיל שלו מאמרים או דעות ובעיקר מיקרו-בלוגים, ובמקרים רבים מאפשרים לחברים ברשת להגיב או להוסיף הערות על פרסומים של חבריהם לרשת.

הרשת החברתית SixDegrees.com הוצגה בשנת 1997 והייתה לראשונה ברשתות החברתיות, לצידה התפתחו רשתות חברתיות רבות, ואלו הפכו לתופעה גלובלית. בשנת 2005 נרכשה הרשת החברתית MySpace בסכום של 580 מיליון דולר והייתה לרשת החברתית שמשכה את עיקר תשומת הלב של האמריקאיים [[12](#Ell07)], וברבות השנים איבדה ממעמדה לטובת פייסבוק שנחשבת לרשת החברתית המקוונת המובילה בעולם כיום ומונה כ-1.19 מיליארד משתמשים פעילים על בסיס חודשי, וכ-727 משתמשים פעילים מדי יום נכון לחודש ספטמבר 2013 [[13](#Fac)].

## הרשת החברתית טוויטר

טוויטר היא רשת חברתית חינמית שנוסדה בשנת 2006 ע"י ג'ק דורסי, אוון ויליאמס וביז סטון. ייחודה של הרשת היא בכך שהיא מבוססת על פרסום וקריאה של מסרים קצרים שאורכם מוגבל ל-140 תווים, אלו מכונים "ציוצים" (tweets). כל משתמש באתר יכול לפרסם ציוצים, וכן להגדיר רשימת משתמשים אחריהם הוא רוצה לעקוב, וכך הוא מקבל עדכונים כאשר המשתמש אחריו הוא עוקב פרסם ציוץ חדש. מעקב אחרי משתמש כלשהו אינו דורש את אישורו, אלא אם הגדיר את המשתמש שלו כפרטי. בין משתמשי טוויטר ניתן למנות ידוענים רבים העושים שימוש תדיר בטוויטר ומפרסמים ציוצים על כל תנועה שלהם, ובכך הם שומרים על קשר עם קהל המעריצים; מומחים לנושאים שונים וחברות מסחריות. המשתמש הוא זה שבוחר אחרי מי לעקוב ובאיזה תוכן לצפות.

פרסום הציוצים וכן המעקב מתאפשר באמצעות אתר האינטרנט של הרשת, האפליקציה הסלולארית, שליחה וקבלה של מסרונים וכן באמצעות תוכנות מסרים אשר יצרו פלטפורמה לצפייה ופרסום באמצעות שימוש ב-API של טוויטר.

נכון לאוגוסט 2013 מדווחת החברה [[24](#Twi)] על כ-500 מיליון ציוצים המפורסמים בכל יום, ע"י למעלה מ-230 מיליון משתמשים הפעילים על בסיס חודשי באתר שמוצע ביותר מ-35 שפות, ושמתוכם 75% פעילים באמצעות מכשירי טלפון ניידים. החברה מעסיקה כ-2000 עובדים ברחבי העולם, ש-50% מתוכם הם מהנדסים. "המשימה שלנו" כך מגדירה טוויטר היא "לתת לכולם את הכוח ליצור ולשתף רעיונות ומידע באופן מיידי, ללא מחסומים".

נכון לינואר 2014 מדווחת חברת דירוג האתרים אלסקה כי טוויטר הוא האתר ה-12 במספר, הנצפה ביותר ברשת האינטרנט [[23](#Tto)].

## למידה חישובית ברשתות חברתיות מקוונות

כפי שניתן לראות במקטעים הקודמים, כמות המידע הנצברת בכל יום ברשתות החברתיות היא עצומה ואף הולכת וגדלה. הרשתות החברתיות מעוניינות לנצל מידע זה כדי להציג יכולות חדשות שישפרו את חווית המשתמש, ובנוסף יפיקו תועלת עסקית. כתוצאה מכך גוברת ההתעניינות בשילוב פתרונות למידה חישובית אשר יענו על צורך זה. פתרונות מסוג אלה מסוגלים לקחת את הנתונים אותם מפרסמים המשתמשים בנוסף לפרטים אותם אוגרת החברה, ולהציע להם תוכן שעשוי לעניין אותם, להציג פרסומות המתקשרות לנושאים המדוברים, ואף להמליץ על חברים שהם עשויים להכיר או קבוצות שאליהן יתכן שירצו להצטרף.

התקווה המרכזית של הרשתות החברתיות היא למצוא חוקיויות סטטיסטיות, אותן ניתן יהיה לזקק לתוך מודל שיניב תשובה לשאלות כמו האם פרסום זה של המשתמש הוא ספאם או לא? האם המשתמש שנרשם הוא אדם אמיתי או נועד לצורך קידום מכירות? מהו הדירוג שעשוי המשתמש לתת לסרט מסוים? האם שני משתמשים מכירים זה את זה? ומה הסבירות שהמשתמש ילחץ על פרסום מסוים?

נציג כדוגמא את התרחיש הבא: שני משתמשים עוקבים אחר זמר מפורסם, לשניהם יש חבר משותף ברשימת החברים וזה מופיע איתם בתמונות רבות שצולמו במקומות בילוי. שימוש בטכניקות למידה חישובית העושות שימוש במידע זה יכול לאפשר לרשת להציע לחבר הנוסף לעקוב גם הוא אחרי אותו זמר, שייתכן שמעניין גם אותו, ובנוסף יוכל האתר להציג למשתמשים אלו פרסומות למופעים בסגנון המוזיקלי של אותו זמר. וזהו רק קצה המזלג.

# רקע ללמידה חישובית

## מהי למידה חישובית?

למידה חישובית היא תחום המחקר של אלגוריתמים המסוגלים ללמוד מתוך דוגמאות. ארתור סמואל תיאר בצורה לא פורמאלית בשנת 1959 את הלמידה החישובית כ"תחום המחקר שנותן למחשבים יכולת ללמוד מבלי שתוכנתו באופן פרוש לכך" [[20](#Sim13)]. מאוחר יותר ניסח תום מיטשל מודרנית יותר: "תוכנת מחשב מוגדרת כלומדת מניסיון E ביחס לקבוצת המשימות T עם מדידת ביצועים P, אם הביצועים שלה במשימות T כפי שהם נמדדים ע"י P משתפרים עם הניסיון E" [[16](#Mit97)].

ניקח כדוגמא משחק שח-מט, ונגדיר עבורו את הפרמטרים:

E - ניסיון שנצבר בעקבות המון משחקי שח-מט.

T - המשימה לשחק שח-מט.

P - ההסתברות שהתוכנה תנצח במשחק הבא.

נהוג לחלק את אלגוריתמי הלמידה החישובית למספר סוגים, ביניהם:

1. למידה מונחית - האלגוריתם מקבל סט של דוגמאות המהווה את הניסיון E כאשר לכל דוגמא מצורפת תווית סיווג. על סמך דוגמאות אלו צריך האלגוריתם לחזות את הסיווג של דוגמאות חדשות אשר לא נמצאות ב-E. למידה זו מאפשר לפתור בעיות כמו רגרסיה וסיווג. לדוגמא, בהינתן נתונים על מכירת בתים באזור מסוים, נרצה להפיק נוסחה שתחזה את מחיר המכירה של בית נוסף באותו איזור.
2. למידה בלתי מונחית - למידה זו מאפשרת לגשת לבעיות בלי רעיון איך התוצאה אמורה להיראות. באפשרותנו להפיק מבנים מהנתונים הידועים לנו, מבלי לדעת בהכרח מה ההשפעה של המשתנים בנתונים. נוכל להפיק את המבנה ע"י יצירת אשכולות נתונים המבוססים על הקשרים בין המשתנים בנתונים. לדוגמא, האלגוריתם יקבל קבוצה של 5,000 מאמרים, ובאופן אוטומטי ידע לצמצם אותם לקבוצות קטנות של מאמרים דומים על סמך משתנים שונים כמו מילים תדירות, מספר עמודים, תאריך הוצאה וכד'.

מאמר זה יתמקד בעיקר בלמידה מונחית, אם כי הרעיונות המוצגים בו נכונים גם ללמידה בלתי מונחית.

## מודלי למידה

מטרתו העיקרית של אלגוריתם לומד היא להכליל מהניסיון שלו. הכללה בהקשר זה היא היכולת של מכונה לומדת להתבסס על ניסיון מדוגמאות קודמות, כדי לחזות דבר על קלט שלא נראה קודם. דוגמאות אלה מכונות לרוב דוגמאות אימון או קבוצת אימון. קבוצת האימון תכיל סט של דוגמאות אימון שהתוצאה שלהם ידועה והן מהצורה , אנו נקרא לפרמטרים המאפיינים את הדוגמא וקטור תכונות ולתגית (התוצאה) שלהם . וקטור התכונה מורכב ממזהה התכונה וערכה, למשל עבור בעיה בה אנו מבקשים לבדוק האם גידול כלשהו הוא ממאיר, וקטור התכונות יכול להיראות כך <גיל החולה, גודל הגידול, האם החולה מעשן> והתגית תהיה 1 עבור גידול ממאיר ו-1- עבור גידול לא ממאיר, מכאן שדוגמת אימון תראה למשל כך כאן גיל החולה הוא 45, גודל הגידול הוא 60 סמ"ק והוא מעשן, כמו כן ידוע שגידול זה ממאיר.

דוגמאות האימון מהוות קלט למודל למידה שלעיתים יקרא גם אלגוריתם למידה או מודל אימון, תפקידו של מודל זה הוא לעבד את הנתונים באופן כלשהו ולהוציא כפלט מודל מאומן שנקרא פעמים רבות גם פונקצית היפותזה או מודל החיזוי.

לאחר שיש ברשותנו מודל מאומן, נוכל להזין לתוכו וקטור תכונות חדש ולחזות את ערכו באמצעות המודל המאומן. תרשים 1 מציג את תהליך זרימת המידע בלמידה חישובית מונחית.



תרשים 1 - זרימת המידע בלמידה מונחית

## למידת אצווה

למידת אצווה דורשת לקבל את כל דוגמאות האימון מראש, מחזיקה אותם בזיכרון, ובאמצעות דוגמאות אימון אלו היא מאמנת את המודל. במקטעים הבאים נציג דוגמה ללמידת אצווה.

### רגרסיה ליניארית

על פי המתואר ב- [[8](#Cou)], בהינתן קבוצה של דוגמאות אימון שכל אחת מהן היא מהצורה *כאשר נרצה להפיק פונקצית היפותזה* כך שמתקיים , וכן היא מהצורה

כעת בהינתן שמצאנו פונקציה כזאת, נוכל להזין לפונקציה וקטור חדש שלא הופיע בדוגמאות האימון ולחזות את ערכו.

*נוכל גם לקבוע כי כל דוגמת אימון היא מהצורה ולסמן*וכן ואז נקבל ייצוג קצר יותר לפונקציה כך:

*בכדי לקבל פונקציה עם רמת דיוק גבוהה נוכל להשתמש בפונקצית העלות (cost function). זו מחשבת ממוצע (או למעשה גירסה מהודרת יותר של ממוצע) של ההפרש בין הקלטים לבין התוצאה שהם הניבו. כלומר עבור וקטור נרצה לחשב את המינימום של הנוסחה הבא:*

נביא כעת דוגמא פשוטה שתדגים את הרעיון הנעוץ בשיטה:

נניח כי נתונה לנו קבוצת האימון הבאה: כאשר נשבץ את הדוגמאות על גבי גרף דו-מימדי וננסה להעביר ישר בין הנקודות כך שהנקודות בגרף קרובות ככל האפשר לישר. משהעברנו את הישר נוכל לבחור ערך על ציר ה- ובאמצעות הישר לחזות את ערכו. כפי שניתן לראות בתרשים 2, נוכל להעביר ישרים רבים שייתכן שיראו לנו מדויקים מספיק, לכן כדי למצוא את הישר המדויק ביותר נצטרך להשתמש בפונקצית העלות.

במקרה זה נקבל את פונקצית העלות הבאה:

תרשים 2

כעת נמצא את המינימום של פונקציה זו ונקבל ו- ומכאן שפונקצית ההיפותזה שלנו היא . ולמשל עבור הקלט 1 הפלט של הפונקציה שלנו הוא 1.3143.

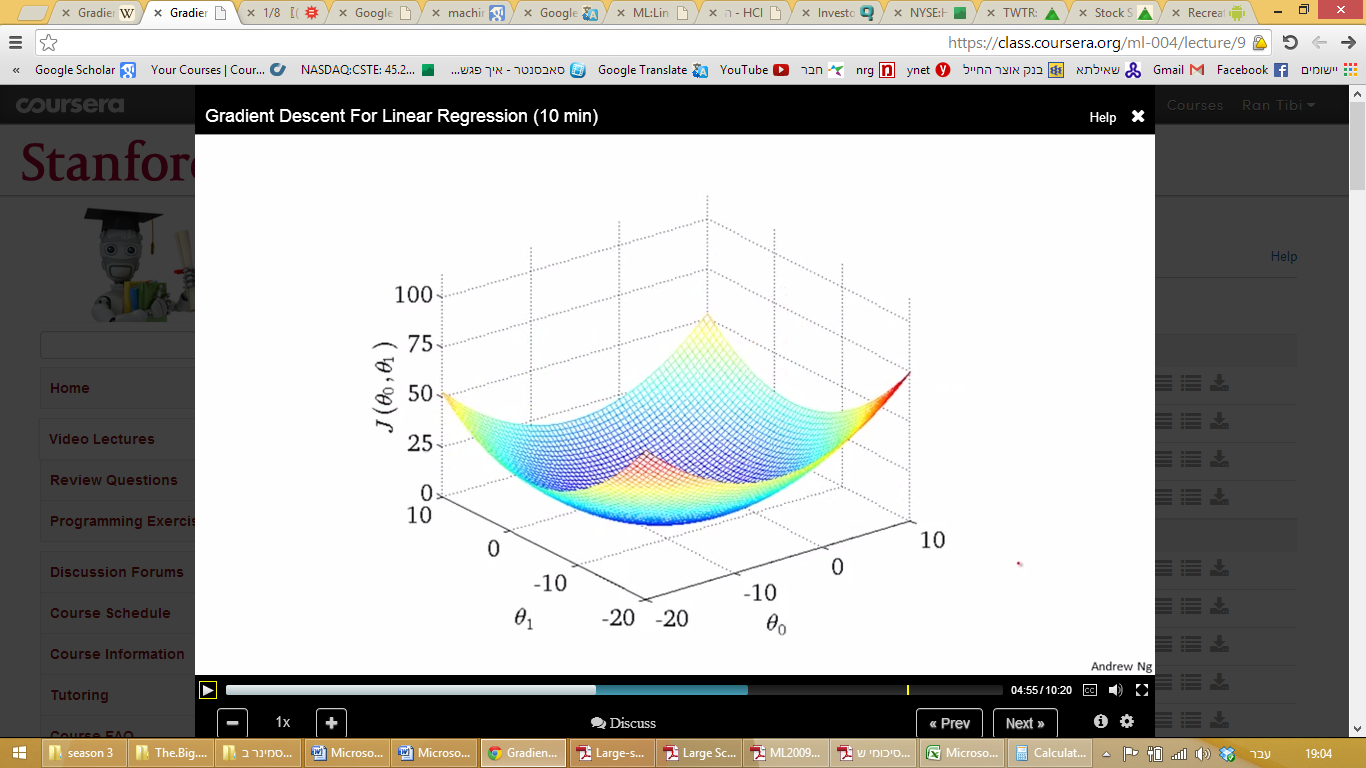
### Gradient Descent

במקטע הקודם הראנו כי כדי לקבל פונקציה היפותזה מדויקת עלינו למצוא וקטור עבורו ערכה של פונקצית העלות הוא מינימאלי. Gradient Descent הוא אלגוריתם המאפשר לנו למצוא ערך זה. למעשה הוא שיטה כללית הנמצאת בשימוש בכל תחום הלמידה החישובית ולא רק ברגרסיה לינארית. האלגוריתם נראה כך:

|  |
| --- |
| choose = (  repeat until convergence{  (for each  } |

הפרמטר באלגוריתם נקרא "קצב הלמידה" (learning rate), והוא מייצג את גודל השינוי שאנו מבצעים בכל איטרציה של הלולאה. כלומר, נבחר וקטור כלשהו, ונשנה את ערכיו בקצב באופן שבו הוא משפר את  *עד שנגיע למינימום מקומי.*

פרט מעניין שבזכותו אלגוריתם זה יכול לשמש אותנו למציאת פונקצית העלות המינימאלית הוא שברגרסיה ליניארית פונקצית העלות היא תמיד פונקציה קמורה, ומכאן שהמינימום המקומי שלה שווה תמיד למינימום הגלובאלי. למשל עבור פונקצית עלות שהקלט שלה הוא וקטור עם שני משתנים נקבל פונקציה מהצורה המוצגת בתרשים 3.



תרשים 3

עבור רגרסיה ליניארית נקבל כי כאשר ועל כן נקבל את האלגוריתם הבא:

|  |
| --- |
| choose = (  repeat until convergence{  (for each  } |

## למידה מקוונת

בלמידה מקוונת בשונה מלמידת אצווה, דוגמאות האימון אינן ניתנות לאלגוריתם מראש, אלא הוא מקבל בכל פעם דוגמא בודדת, ובהתאם אליה משנה את המודל הנלמד.

אלגוריתם למידה מקוון עובד על עקרון של צעדי זמן, בצעד זמן מוצג לאלגוריתם וקטור תכונות ותגית (ברוב הדוגמאות ), אלו מהווים ביחד דוגמת אימון. האלגוריתם מנבא את התגית של הווקטור , ולאחר מכן משווה את התגית שהתקבלה לתגית שהוצגה בדוגמת האימון. האלגוריתם "נענש" על כל טעות שהוא עשה, כלומר מבצע שינוי כלשהו באופן שבו הוא מנבא. המטרה שלנו היא לייצר אלגוריתם המקבל מספר קטן ככל האפשר של טעויות. דוגמאות רבות לאלגוריתמים מקוונים ניתן למצוא ב- [[9](#Cra06)] [[17](#Ngu11)] [[4](#Blu98)].

# בעיית הסיווג

## מהי בעיית הסיווג?

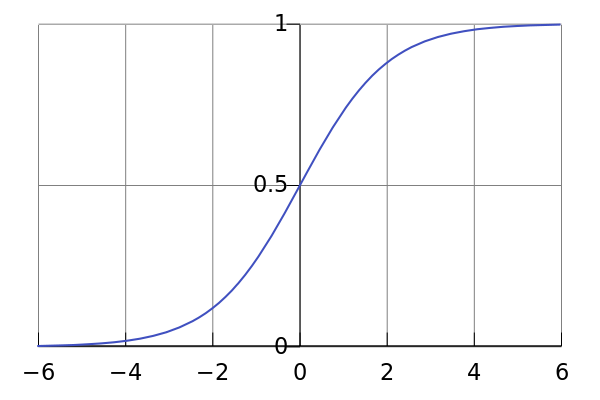
בבעיית הסיווג אנו מתבקשים לייצר מודל המקבל וקטור תכונות ומחזיר תגית . אנו נעסוק רק בבעיות מסוג שכן ניתן להפוך כל בעיית סיווג רב מחלקתית, למספר בעיות סיווג בינאריות. לרוב 0 ייצג את המחלקה השלילית ו-1 את המחלקה החיובית, אך אנו רשאים לייצג זאת כיצד שנרצה.

דוגמא פשוטה לבעיית הסיווג היא יצירת מערכת מומחה אשר בהינתן וקטור תכונות כמו (גיל המטופל, מין המטופל, גודל הגידול, כמות הכדוריות האדומות בדם,...) צריכה לקבוע האם הגידול של חולה הוא ממאיר או לא.

אחת הדרכים לבצע זאת היא באמצעות רגרסיה לינארית באופן כזה שבו אנחנו קובעים כי עבור חיזויים הגדולים מ-0.5 נסווג ועבור גידולים קטנים מ-0.5 נסווג . שיטה זו לא עובדת טוב כל כך שכן ברוב המקרים לא מדובר בפונקציה לינארית.

## רגרסיה לוגיסטית

על פי המתואר ב- [[8](#Cou)], כאשר נפתור בעיית סיווג נרצה להשתמש בפונקצית היפותזה המקיימת . בעוד שברגרסיה ליניארית פונקצית ההיפותזה היא  *. ברגרסיה לוגיסטית נעשה שימוש בפונקצית היפותזה מהצורה נסמן ונגדיר את ההיפותזה באמצעות הפונקציה הלוגיסטית . יתרונה של פונקציה זו היא שהיא אכן מפיקה טווח ערכים רצוי ומציגה גרף מהצורה המוצג ב*תרשים *4.*



תרשים *4*

*ואם כן, באמצעות ההיפותזה זו אנו מקבלים את ההסתברות ש- מסווג כ-1, כלומר* . ומכאן שנוכל לחשב גם כי .

כעת כשנרצה לסווג נצטרך להחליט מה רמת הסמך אותה אנחנו מבקשים לקבל, למשל אם מדובר בעיה שאמורה לקבוע האם גידול ממאיר או לא, ייתכן שנבחר לקבוע שכן גם במצב שההסתברות לכך היא 40%, בעוד שאם רצוננו לקבוע מה מידת ההתאמה של מועמד למשרה בטרם נזמן אותו לראיון, ייתכן שנזמן אותו רק אם מתאים בהסתברות של -70%.

משמצאנו את פונקצית ההיפותזה, נצטרך למצוא את הווקטור . לצורך כך נצטרך להגדיר את פונקצית העלות, אולם לא נוכל להשתמש בפונקצית העלות שהגדרנו עבור רגרסיה לינארית, שכן היא מכילה הרבה נקודות מינימום מקומי, כלומר היא לא פונקציה קמורה. ולכן, נעשה שימוש בפונקצית העלות הבאה:

כאשר

נוכל גם לנסח פונקציה זו כך:

ונקבל כי פונקצית העלות היא:

כל שנותר לנו כדי למצוא את הווקטור כעת הוא לעשות שימוש ב- Gradient Descentבאותה צורה שביצענו זאת ברגרסיה לינארית, וכך נקבל את פונקצית ההיפותזה שחיפשנו.

# למידה חישובית בסדר גודל גדול

טכניקות למידה חישובית רלוונטיות לתחומים כמו כריית מידע, ראייה ממוחשבת וזיהוי דיבור, שעם הזמן כמות נתונים בהן הולכת וגדלה. נתונים אלו יכולים להיות מיליארדים שורות של טקסט, טרה של תמונות או שעות דיבור, די להסתכל על מספר הציוצים המפורסמים בטוויטר ביום כדי לאמת טענה זו. לפיכך עולה צורך להתאים את האלגוריתמים קיימים בלמידה חישובית להתמודדות עם כמות נתונים כזאת, או לייצר פתרונות למידה חדשים [[7](#Col04)].

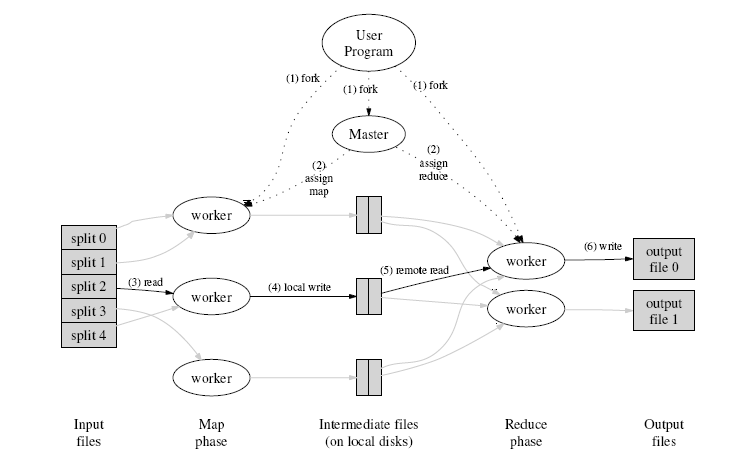
אחד הבעיות באלגוריתמי למידה חישובית, בעיקר אלגוריתמי אצווה, היא שהן תוכננו בצורה כזאת שכלל דוגמאות האימון צריכות להיות בזיכרון, דבר שכמובן בלתי אפשרי גם במשאבי החומרה של ימינו.

כמו כן, בחלק מהמקרים, האלגוריתמים כוללים סקירה חוזרת של הנתונים שעלותה גדולה בסדרי גודל מכמות הנתונים שעצומה כשלעצמה, ובמקרים אלו יכולים האלגוריתמים להיות כלל לא ישימים מבחינת זמן ריצה.

במקטעים הבאים יוצגו שיטות שונות בהן ניתן להשתמש כדי ליישם למידה חישובית בסדר גודל גדול.

## למידת חישובית תוך שימוש ב-MapReduce

MapReduce הוא מודל תכנות ומימוש הקשור בעיבוד ויצירת קבוצות נתונים גדולות. עפ"י המודל על המשתמש להגדיר את הפונקציה map() המקבלת מפתח/ערך ומייצרת קבוצה של ערכי ביניים מהצורה מפתח/ערך, וכן מגדיר פונקציה reduce() הממזגת את כל ערכי הביניים הקשורים לאותו מפתח ביניים [[10](#Dea08)]. תרשים 5 מציג את זרימת המידע תוך שימוש באלגוריתם.



תרשים 5 - זרימת המידע ב-MapReduce [[10](#Dea08)]

שימוש במודל זה מאפשר לייצר תוכניות המבצעות עיבוד נתונים בצורה מקבילית ומבוזרת. חשוב מכך, המודל מאפשר למשתמש שאינו מבין בתכנות מקבילי ומבוזר לעשות שימוש בחבילות רבות המטפלות בביזור העיבוד וניצול אופטימאלי של המשאבים, כפי שנוכל לראות במקטעים הבאים. נציין כי במקרים רבים לצורך שיפור ביצועים על המשתמש להגדיר פונקציות נוספות, בהן פונקצית חלוקה, איחוד ומיון.

נדגים כעת מימוש פשוט ל-MapReduce עבור בעיית ספירת המילים. בבעיה זו אנו מבקשים לקבל מפה מצורה (מילה/מספר מופעים) עבור טקסט מסוים. מובן שיכולנו לעבור על כל הטקסט ולמנות את המילים, אולם אם מדובר בטקסט ארוך מאוד, נעדיף לבצע פיצול של פעולת הספירה על מעבדים שונים.

כדי לבצע זאת, ניקח את מסמך הקלט ונחלק אותו ל- חלקים שווים בגודלם. כעת בצורה מקבילית נקראת לפונקציה map עם כל אחד מהחלקים.

נראה אם כן פסאדו-קוד לפונקציה map שבמקרה זה מייצרת מפה של (מילה/ערך 1):

|  |
| --- |
| map(String key, String value):  // key: never mind  // value: document part contents  for each word w in value:  EmitIntermediate(w, 1) |

בשלב זה, לרוב יופעל רכיב מאחד (combiner) שתפקידו לאחד את כלל התוצאות עבור אותו מפתח במכונה בודדת, פעולה זו דומה לפעולת ה-reduce רק שהיא מתבצעת על מחשב בודד.

לאחר מכן, כלל הפלטים של הפונקציה map (או ה-combiner) יהוו קלט לפונקציה reduce אשר תבצע סכימה של כל התדירויות למילה בודדת (תפקידה של החבילה המממשת יהיה להזין לפונקציה רשימה של כל הערכים המתאימים למילה (המפתח) כפי שהתקבלו מהפונקציה map). להלן פסאדו-קוד של הפונקציה reduce:

|  |
| --- |
| reduce(String word, Iterator partialCounts):  // word: a word  // partialCounts: a list of aggregated partial counts  sum = 0  for each pc in partialCounts:  sum += pc  emit (word, sum) |

תרשים 6 מציג דוגמא של זרימת הנתונים עבור קלט מסוים.



תרשים 6 - פתרון בעיית ספירת המילים הממומשת באמצעות MapReduce. בירוק פונקציות שעל המשתמש לממש, בחום פונקציות שיממשו ע"י החבילה בה משתמשים (נציין כי לעיתים פונקצית החלוקה תמומש גם ע"י המשתמש)

ואם כן, באמצעות שיטה זו ביזרנו ומקבלנו את הבעיה וקיבלנו פתרון בזמן קצר יותר.

MapReduce נמצא בשימוש נרחב במימוש מערכות למידה חישובית אשר מטפלות בכמות נתונים בסדר גודל גדול. ברגרסיה ליניארית למשל אנו מנסים למצוא מינימום לפונקצית העלות , ולכן תוך שימוש ב-gradient descent אנו נדרשים עבור כל לחשב פעמים רבות את הסכום  
 , כאשר כזכור הן דוגמאות האימון, במימוש MapReduce של בעיה זו, נחלק את דוגמאות האימון ל- קבוצות, המימוש ל-map יהיה לחשב את הסכום של הפעולה combine תמזג את הסכום עבור האיברים באותה מכונה, מימוש של ה-reduce יהיה לחשב סכום כולל של כל הסכומים הזמניים לסכום אחד.



תרשים 7 - דוגמא לשימוש MapReduce ברגרסיה ליניארית לצורך חישוב כאשר

דוגמאות נוספות לשילוב MapReduce באלגוריתמים ידועים בלמידה חישובית ניתן למצוא ב- [[6](#Chu07)].

## Stochastic Gradient Descent

כזכור, האלגוריתם Gradient Descent מחשב את הערך בכל איטרציה של הלולאה, ובפועל סורק את כל דוגמאות האימון פעם אחר פעם. כאשר עושים שימוש בקבוצת דוגמאות אימון בסדר גודל גדול, משך הריצה של האלגוריתם יכול להיות ארוך מאוד, ופעמים רבות חישוב זה כלל לא מעשי מבחינת זמן הריצה.

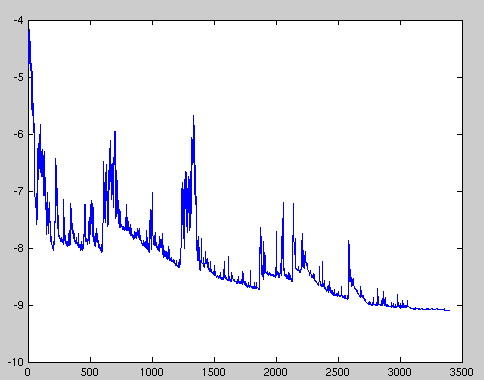
האלגוריתם Stochastic Gradient Descent (SGD) הוא פישוט דרסטי של Stochastic Gradient. במקום לחשב את הסכום פעם אחר פעם, בכל איטרציה *נבחר דוגמת אימון אחת באופן רנדומאלי מסך דוגמאות האימון ונשפר*  כך *. פישוט זה משפר את ריצת האלגוריתם בסדר גודל.*

*בפועל נקבל את האלגוריתם הבא:*

|  |
| --- |
| choose = (  Randomly 'shuffle' the dataset  Repeat until an approximate minimum is obtained{  for from 1 to  (for each  } |

*לא נדון כאן בשאלת ההתכנסות, אך מן הראוי לציין כי כאשר כמות דוגמאות האימון גדולה מספיק, וכן בחרנו פרמטר מתאים, ובכפוף להנחות מתונות יחסית SGD מתכנס כמעט בוודאות למינימום גלובאלי, זו תוצאה נוספת של משפט רובינס-זיגמונד עליו ניתן לקרוא בהרחבה ב-* [[19](#Rob85)]*.*

*עניין נוסף אותו חשוב להדגיש, הוא אופן ההתקדמות אל המינימום הגלובלי. בעוד ש-*Gradient Descent *מקדם עצמו בכל איטרציה אל המינימום הגלובלי, SGD יכול במהלך האלגוריתם להתקרב ולהתרחק ממנו בכל שלב. אולם ככל שיותר דוגמאות אימון מחושבות, כך השינוי במרחק מהמינימום הגלובלי הולך וקטן. ניתן לקבל תחושה לכך בניתוח של* תרשים *8.*



תרשים *8 -* [[21](#Sto)] *ציר ה-x מציג את מספר דוגמאות האימון (האיטרציות) והציר ה-y מייצג את ערך אליו פונקצית העלות מתכנסת*

*נציין גם כי ניתן לעשות שימוש ב-SGD כאלגוריתם מקוון. דוגמאות רבות ל-SGD ניתן למצוא ב* [[5](#Bot10)]*.*

## שיטות הרכב (Ensemble Methods)

המוטיבציה לעסוק בשיטות הרכב אינה דווקא בלמידה חישובית בסדר גדול, אך כפי שניווכח בהמשך ניתן לעשות שימוש בה גם כדי לתת מענה ללמידה בתחום זה.

שיטות הרכב הם אלגוריתמי למידה אשר בונים קבוצה של מסווגים ואז מסווגים נתונים חדשים ע"י הצבעה (בעלת משקל) עבור החיזוי שלהם [[11](#Die01)]. בבואנו לפתור בעיית סיווג סטנדרטית, יש ברשותנו קבוצה של דוגמאות אימון , קבוצה זו מהווה קלט לאלגוריתם למידה שהפלט שלו הוא מסווג. המסווג הוא בפועל פונקצית היפותזה אשר בהינתן חדש חוזה את ערכו של המתאים לו.

מובן שקיימים אלגוריתמי למידה שונים אשר הפלט שלהם הוא מסווגים שונים, נסמן מסווגים אלו כ-. בהינתן חדש, שיטות הרכב מציעות לקחת את כל החיזויים שהניבו המסווגים ולשלב אותם בדרך כלשהי.

אחת הדרכים הנפוצות לשלב בין החיזויים השונים היא שיטת הכרעת הרוב. בשיטה זו אנו מפעילים את הנוסחה הבא:

כאשר היא ההסתברות שהמסווג ינבא בהינתן . המקדם הוא המשקל אותו מבקשים לתת למסווג . כאשר אנו רואים את כל המסווגים כשווים אז ערכו יהיה 1.

המטרה בשיטות הרכב היא להגיע למצב שבו הרכב מסווגים הוא מדויק יותר מאשר מסווג בודד. ההגדרה למסווג מדויק הוא כזה שבהינתן הסיכוי שלו לטעות בסיווג קטן הסיכוי לטעות בניחוש רנדומאלי. *ומובן שככל הסיכוי של המסווג לטעות קטן יותר, כך הוא נחשב מדויק יותר. אנו גם נגדיר שני מסווגים כשונים אם עבור* קלטים שונים הם יניבו שגיאות שונות. כדי להיווכח למה יש חשיבות בכך נסתכל על הדוגמה הבא: נניח כי בהינתן המסווגים מניבים כולם שגיאה, אזי אין כל ערך לשימוש במספר מסווגים, ואולם אם הניב שגיאה ואילו ו- הניבו חיזוי נכון, אז די בשימוש בבחירה עפ"י הסיווג השכיח כדי לספק סיווג נכון. ניתן אף להציג בצורה פורמאלית את טענה זו, אם ההסתברות לשגיאה במסווג בודד מקיימת וההסתברות לשגיאה בין המסווגים היא בלתי תלויה, אז ההסתברות שהצבעת הרוב תניב חיזוי שגוי תהיה הסתברות שיותר מ- מסווגים הניבו שגיאה [[11](#Die01)].

[[22](#Str01)] מציע אלגוריתם המאפשר לנצל את שיטות הרכב לצורך סיווג על בסיס נתונים בסדר גודל גדול. להלן פסאדו-קוד של האלגוריתם:

|  |
| --- |
| while there is more training examples in  select d training examples is into  create a classifier using  calculate by evaluate using  calculate for each classifier on by evaluate using  if is not full  add  else if > for some  replace with in |

בהינתן קבוצת אימון , נרצה לייצר הרכב של מסווגים שנסמנה . לצורך כך אנו מאמנים את המסווג ע"י תת קבוצה של בגודל . בנוסף אנו מחשבים את איכות המסווג באמצעות הפעלתו על תת קבוצת האימון , וכן את איכות כל אחד מהמסווגים ב-. אם הקבוצה אינה מלאה, אז נוסיף אליה את , אחרת נבדוק אם קיים מסווג ב- שאיכותו נמוכה מ- ונחליף אותו ב-. וכך נמשיך בהתהליך עד שנסרוק את כל דוגמאות האימון. בצורה זו אימנו כל מסווג על מספר דוגמאות אימון קטן יותר, אך בפועל אימנו את ההרכב על כלל דוגמאות האימון, ואף שיפרנו את איכותו. בנקודה זו ראוי להדגיש כי למרות שהאלגוריתם המוצג הוא איטרטיבי נוכל בקלות רבה יחסית לבזר את פעילות האלגוריתם על מספר מכונות, ובכך לשפר עוד יותר את ביצועיו.

# מימוש למידת מכונה בסדר גודל גדול באמצעות Hadoop ו-Pig

## אודות Hadoop

Apache Hadoop היא פרויקט קוד פתוח לתכנות מבוזר, אמין ופתוח להרחבות [[1](#Apa)].

ספריית התוכנה Hadoop היא חבילה המאפשרת עיבוד מבוזר של מבני נתונים גדולים על אשכולות של מחשבים (clustering) תוך שימוש במודלי תכנות פשוטים. היא עוצבה באופן כזה שבו האשכולות ניתנים להרחבה, החל ממספר שרתים בודדים ועד לאלפים, אשר כל אחד מהם מציע עיבוד ואחסון מקומי. במקום להסתמך על זמינות גבוהה של החומרה, הספרייה עצמה תוכננה לזהות ולטפל בכישלונות בשכבת היישום, ולכן מספקת שירות בזמינות גבוהה בהרצה על אשכול מחשבים, כאשר כל מכונה בודדת יכולה להיות מקור לבעיות.

הפרויקט מכיל את המודולים הבאים:

* Hadoop Common - יכולות משותפות המשתמשים מודולים אחרים.
* Hadoop Distributed File System (HDFS™) - מערכת קבצים מבוזרת המספקת גישה לנתוני האפליקציה בתפוקה גבוהה.
* Hadoop YARN - חבילה לתזמון משימות וניהול מקורות מבוזר.
* Hadoop MapReduce - מערכת מבוססת YARN לעיבוד מקבילי של מבני נתונים גדולים באמצעות MapReduce.

## אודות Pig

Apache Pig היא פלטפורמה לניתוח מבני נתונים גדולים המורכבת משפה עילית לביטוי תוכניות לניתוח נתונים, בשילוב עם תשתית להרצת תוכניות אלו [[2](#Apa1)]. המאפיין המרכזי של תוכניות Pig הוא שהמבנה שלהן הוא נוח לתכנות מקבילי, דבר המאפשר להן להתמודד עם מבני נתונים גדולים מאוד.

כיום, שכבת התשתית של Pig מורכבת ממהדר אשר מיצר סידרה של תוכניות מבוססות MapReduce, שעבורן מימוש מקבילי לכמות נתונים גדולה כבר קיים (למשל Hadoop) שכבת השפה של Pig המכונה Pig Latin מאופיינת בקלות תכנות, אופטימיזציה אוטומטית המאפשר למשתמש להתמקד בסמנטיקה ולא ביעילות, ופתיחות להרחבות המאפשרת למשתמש ליצור פונקציות משלו.

## הארכיטקטורה של Pig

לא ניתן לסקור בקצרה את כלל הארכיטקטורה של Pig, אולם ישנם מספר עקרונות רעיונות שצריך להכיר כדי להבין את הנאמר במקטעים הבאים, ולכן נציג רעיונות אלה בקצרה.

כאמור, PigLatin היא שפת סקריפטים המאפשרת לייצר סידרה של תוכניות מבוססות Map Reduce, והיא עושה שימוש לרוב בתשתית של Hadoop כדי לממש זאת. כל סקריפט ב-Pig מורכב משלושה שלבים:

1. טעינה (load) - בשלב זה נטענים הנתונים אשר מבוזרים על פני קבצים ב-HDFS.
2. סדרת טרנספורמציות על הנתונים - בשלב זה למעשה מתבצע עיבוד הנתונים, ע"י תהליכים שחלקם מבוססים MapReduce.
3. אחסון (store) - שמירה של הנתונים לקבצים ב-HDFS.

כך יראה לדוגמה סקריפט Pig פשוט הממיין נתונים על ילדים לפי גיל:

|  |
| --- |
| A = load 'children' AS (name, age);  A = order A by age;  store A into '/data/examples/sorted'; |

בעוד שלקורא נראה כאילו מדובר בסקריפט סדרתי, פעולת המיון למשל עשויה להתבצע כפעולות MapReduce, וכתוצאה מכך מספר מחשבים שונים יעבדו בנפרד חלק מהנתונים שנטענו במקביל, ובכך קיבלנו קוד מקבילי, המנצל כראוי את המשאבים, מבלי שכלל היה עלינו לעסוק בכך (כדי להבין איך פעולת המיון יכולה להתבצע באופן מקבילי יש צורך להבין את זרימת הנתונים במימוש Hadoop ל-MapReduce הסבר מפורט על כך ניתן למצוא ב- [[25](#Yah)]).

תהליך הטעינה הוא זה שמחלק את הקלט לחלקים המהווים קלט לפונקציה map והפלט שלה (לאחר מספר תהליכים נוספים) יהווה קלט לפונקציה reduce. אין באפשרותנו לשלוט במספר ה-maps אך באמצעות הפקודה parallel נוכל לקבוע את מספר ה-reduces, כאשר ההגדרה ב-pig היא שלכל reduce יש קובץ output משלו. לפיכך אם נגדיר A = order A by age parallel 5, נקבל בסוף ריצת הסקריפט (כלומר לאחר שנקרא גם פונקצית האחסון) 5 קבצים שהתוכן שלהם ממוין לפי גיל, המאוחסנים ב-HDFS ואם נחבר אותם נקבל את רשימת הילדים המלאה ממוינת.

בנוסף לפעולות הסטנדרטיות שאותן מספקת Pig יש באפשרותנו לממש את פונקצית הטעינה, פונקצית האחסון וכן פעולות טרנספורמציה ע"י יצירת מחלקות בג'אווה היורשות ממחלקות אב מתאימות והתייחסות אליהן בסקריפט. פעולות אלו נקראות UDF(User Define Function) והסבר על איך לממשם ניתן למצוא ב- [[3](#Pig)].

## איך לממש למידה חישובית ב-Pig

[[15](#Lin12)] מציג מקרה מבחן למימוש למידה חישובית בסדר גודל גדול תוך שימוש ב-Pig כפי שבוצע בחברת טוויטר. Pig משמשת כפלטפורמה לעיבוד Big Data בחברה, ועשו בה שימושים רבים עוד טרם שילוב למידה חישובית בסדר גודל גדול, ולכן העיקרון המנחה בתשתית שתוצג בהמשך הוא לשלב למידה חישובית כ"עוד סקריפט ל-Pig". המימוש המוצע עושה שימוש ביכולות שהוזכרו קודם לכן במסמך זה, ונכון לחלק אותו לשלושה חלקים: חבילת למידה בשפת Java, תהליך אימון המודל באמצעות Pig, שימוש במודל הנלמד באמצעות Pig. מימוש שלושת החלקים הנ"ל מאפשר פתרון מלא מקצה לקצה ללמידת חישובית בסדר גודל גדול.

### חבילת למידה בשפת Java

לטוויטר קיימת חבילת ליבה בה ממומשים אלגוריתמי למידה חישובית רבים, אולם כל המעוניין יכול לממש חבילה זו בעצמו, וכן לעשות שימוש חוזר בחבילות למידה חישובית קיימות, נציג כאן את העיצוב המוצע לצורך מימוש הפתרון המדובר.

תחילה יש להגדיר ייצוג לוקטור תכונות. ניתן יהיה להשתמש פשוט ב-Map הממפה בין מזהה תכונה לערכה (חשוב להקפיד לממש וקטור זה בצורה קומפקטית שכן אחרת הוא ייצור תקורת זיכרון מיותרת שעלולה להכביד על תהליך הלמידה), נגדיר גם תווית שתייצג את הסיווג, וייתכן שלצורך נוחות נרצה להגדיר גם מחלקה של דוגמת אימון המכילה וקטור תכונה ותווית. ראה תרשים 9.



תרשים 9

נגדיר מסווג כאובייקט המממש את הממשק Classifier אשר מכילה את המתודה classify המקבלת אובייקט וקטור תכונות כפי שבחרנו להגדיר אותו ומחזירה תווית עבור וקטור התכונות שקיבלה. לכל מסווג יש את היכולת לסרלז (serialize) את עצמו וכן לטעון את עצמו משטף נתונים אבסטרקטי (deserialize). ראה תרשים 10.



תרשים 10

כמו כן ישנם שני סוגי מאמנים: מאמני אצווה ומאמנים מקוונים. מאמני אצווה מממשים את תבנית העיצוב Builder ומממשים את המתודה train המקבלת אוסף של (תווית, וקטור תכונות) ומחזירה מסווג מאומן. מאמנים מקוונים לעומת זאת מממשים פשוט את המתודה update המקבלת תווית ווקטור תכונה ומשפרת באמצעותה את המודל. ראה תרשים 11.



תרשים 11

בנוסף למחלקות אלה יש צורך ליצור עטיפה באמצעות מחלקות Pig אשר תאפשר שימוש במחלקות אלה.

### אימון המודל

לאימון המודל נייצג את וקטור התכונות ב-Pig לפי הסכמה הבאה:

|  |
| --- |
| (label: int, features: map[]) |

ניתן לעשות שימוש גם בפורמט SVMLight הנתמך ע"י חבילות רבות. בפורמט זה כל שורה תייצג דוגמת אימון, תתחיל בתווית {+1,-1} ולאחריה רווח אחד או יותר ואז התכונות יופיעו כ-תכונה:ערך. בפורמט זה קובץ דוגמאות אימון יראה למשל כך:

|  |
| --- |
| +1 age:18 heigth:190  -1 age:15 heigth:140  -1 age:14 heigth:140 |

בפתרון המוצע נממש את המאמן בפונקצית האחסון של Pig. בדרך כלל, פונקציות האחסון מקבלות רשומות פלט, מסרלזות אותן, וכותבות את התוצאה לדיסק. במקרה זה, פונקצית האחסון מקבלת רשומות פלט ומזינה אותם ללומד (מבלי לכתוב כל פלט). רק כאשר כל הרשומות עובדו כותבת פונקצית האחסון את המודל הנלמד לדיסק. מכאן נקבל כי סקריפט האימון יכול להיראות פשוט כך:

|  |
| --- |
| training = load `training.txt' using SVMLightStorage() as (target: int, features: map[]);  store training into `model/' using FeaturesLRClassifierBuilder(); |

בדוגמה הנ"ל FeaturesLRClassifierBuilder פונקצית אחסון ב-Pig אשר עוטפת את הלומד עבור מסווג רגרסיה לוגיסטי לפי הרעיון שמימשנו במקטע הקודם.

פתרון זה בעייתי עבור למידת אצווה שכן על פונקצית האחסון לאגור את כל דוגמאות האימון טרם תחילת האימון, ואולם הקצאת הזיכרון עבור משימות reduce ו-store ב-Hadoop קטנה יחסית. לעומת זאת, הפתרון עובד מצוין עבור למידה מקוונת שאינה דורשת זיכרון רב שכן המאמן מקבל את דוגמת אימון, משפר על פיה את המודל ואז מוחק אותה. לומדים מקוונים הם עדיפים למבני נתונים גדולים, אבל לרוב לומדי אצווה מציגים ביצועים טובים יותר לפחות נתונים ולכן הם שימושיים לניסויים.

יתרון נוסף שמציע פתרון זה, הוא היכולת לשליטה במספר פונקציות ה-reduce ע"י שימוש בפקודה parallel. בדרך כזו ניתן לשלוט על מספר המודלים הנלמדים, ואז לעשות שימוש בשיטות הרכב. הנ"ל מתאפשר מאחר שכזכור לכל reduce יש פונקצית אחסון משלו. ניתן לעשות זאת למשל כך:

|  |
| --- |
| training = load `training.txt' using SVMLightStorage() as (target: int, features: map[]);  training = foreach training generate label, features, RANDOM() as random;  training = order training by random parallel $N;  training = foreach training generate label, features  store training into `model/' using FeaturesLRClassifierBuilder(); |

פעולת ערבול (מיון על סמך ערך אקראי) דוגמאות האימון היא פעולה מקובלת מאוד באימון מודלים, אך חשוב מכך היא מעוררת פעולת MapReduce, ובאמצעות קביעת הפרמטר N אנו קובעים את מספר המופעים של ה-reduce וכפועל יוצא מכך - כמה מודלים יאומנו, כאשר כל מודל יאומן עם חלק מדוגמאות האימון.



תרשים 12 - מדגים איך לומדים משולבים פונקציות האכסון ב-Pig. אנו שולטים במספר המודלים המאומנים ע"י שליטה על מספר ה-reduce: משמאל מסווג יחיד, ומימין שני מסווגים.

### השימוש במודל הנלמד

כאמור, המודל הנלמד הוא בסופו של דבר מופע של אובייקט ב-Java. מימוש עטיפת UDF לאובייקט זה תאפשר גישה אליו דרך Pig, ואז גם תהליך הלמידה יוכל להיות "עוד סקריפט ל-Pig". הסקריפט יטען את וקטורי התכונות עליהם אנו מעוניינים לבצע חיזוי, יעביר אותם ל-UDF, וזו תטען את המודל הנלמד, תבצע את החיזוי ותחזיר אותו. מובן שמאחר והמודל הנלמד הוא אובייקט Java השמור ב-HDFS אין מניעה כי הוא ייטען גם ע"י מערכות אחרות שאינן עושות שימוש ב-Pig.

עניין נוסף שיש לתת עליו את הדעת הוא אימות המודל הנלמד. טרם אישור המודל הנלמד לשימוש, נרצה לאמת אותו באמצעות הרצה על קבוצת בדיקה באמצעות נתונים מתויגים שלא היו חלק מתהליך האימון. נצפה כי החיזוי יניב תוצאות העונות לדרישות מסוימות המשתנות מבעיה לבעיה. תהליך זה יוכל גם הוא להתבצע ב-Pig בדומה לאופן שבו נבצע חיזוי באמצעותו, נטען את דוגמאות הבדיקה, נעביר ל-UDF המתאים את הנתונים ללא התגית ונקבל ממנו את החיזוי, ואז נוכל לבצע על חיזוי זה בדיקות שיאמתו כי המודל הנלמד אכן עומד בתנאים הנדרשים. הבדיקות אותן ניתן להוסיף הן רבות וגם הן משתנות לפי צורך למשל מה אחוז החיזויים הנכונים. סקריפט Pig כזה יכול להיראות למשל כך:

|  |
| --- |
| define Classify ClassifyWithLR(`model/`);  data = load `test.txt' using SVMLightStorage() as (target: double, features: map[]);  data = foreach data generate target, Classify(features) as prediction;  // Add your data tests here |

לפי אותו עיקרון, נוכל לייצר UDF היודע לחזות או לאמת מודל שאומן בשיטות הרכב, ה-UDF ימומש כך שהוא מקבל כתובת לתיקיית המודלים ואת וקטור התכונות, זה יטען את המודלים, יקבל מכל אחד מהם את החיזוי, ואז ימזג את התוצאות לכדי חיזוי (למשל סיווג) בודד באמצעות שיטה כלשהי, למשל הצבעת הרוב.

# ניסוי ניתוח רגשי על כמות נתונים גדולה

[[15](#Lin12)] מדגים את אופן מימוש הפתרון, ואת תוצאותיו באמצעות ניסוי ניתוח רגשי שבוצע במסגרת הפיתוח, שבמסגרתו נעשה שימוש בכלל הרכיבים שהוזכרו לעיל ואף נבחנה האפקטיביות שלהם.

## תיאור הבעיה

בבעיית הניתוח הרגשי אנו מבקשים לייצר אלגוריתם המסוגל להבין או לשער מה היתה תחושת הכותב בעת שכתב טקסט מסוים, למשל ציוץ. אנו מעוניינים לשאול שאלות כמו: האם היה שמח או עצוב? האם הרגש הוא חיובי או שלילי? האם הוא מרוצה או מאוכזב?

[[18](#Pak10)] מציג דוגמא למוטיבציה לעיסוק בתחום זה, ולכך שהוא זוכה להתעניינות כה רבה בשנים האחרונות. באמצעות ניתוח רגשי, חברות המאפשרות פרסומים ע"י המשתמשים, כמו למשל טוויטר, יכולות לקבל מידע מפרסומים אלו העונה על שאלות כדוגמת: מה אנשים חושבים על המוצר, השירות או החברה? האם תחושתם של אנשים ביחס למוצר היא חיובית או שלילית? או איך האנשים היו מעדיפים שהמוצר יהיה?

כאמור, הפלטפורמה של טוויטר מבוססת כולה על ציוצים שהם למעשה מיקרו-בלוגים, ומכאן שההתעניינות בתחום זה בחברה היא גדולה.

בניסוי המדובר, התמקדו מבצעיו בפתרון מונחה נתונים, ובחרו שלא לשלב יכולות כמו ניתוח טקסט. המטרה בניסוי היא לבצע סיווג בינארי: בהינתן ציוץ שידוע מראש כי הוא מביע רגש כלשהו משימת המסווג היא לחזות .

## אופן מימוש הפתרון

בכדי לייצר דוגמאות אימון, נעשה שימוש ב"טריק הסמיילי". ההנחה היא ברוב המקרים כאשר ציוץ מכיל סמיילי מחייך ":-)" הוא מביע רגש חיובי, וכאשר הוא מכיל סמיילי עצוב ":-(" הוא מביע רגש שלילי. מובן שאין זה נכון בכל המקרים, למשל כאשר תוכן ההודעה מכיל עוקצנות או ציניות, אך מבחן התוצאה כפי שניווכח בהמשך מראה כי איסוף דוגמאות אימון בשיטה זו הוא סביר למדי.

לצורך הבדיקה הוכנו קבוצות אימון שכללו {1,10,100} מיליון ציוצים באנגלית המכילים סמיילים ויותר מ-20 תווים שפורסמו לפני ה-1 לספטמבר 2011, דוגמאות אלו הכילו כמות זהה של דוגמאות חיוביות ושליליות. וכן הוכנה באותה אופן קבוצת בדיקה שתבדוק את איכות המודל הנלמד, מ-1 מליון ציוצים שפורסמו אחרי ה-1 לספטמבר 2011. הסמיילים שימשו כדי לסווג את הציוצים והם הוסרו מהם בעת האימון והבדיקה. קבוצות אלו נשמרו ב-HDFS.

הניסוי השתמש במסווג רגרסיה לוגיסטית פשוט המשתמש ב- SGDמקוון, תוך שימוש ב- hashed byte 4-grams כתכונות. בשיטה זו מייצא התכונה מתייחס לציוץ כמערך בתים גולמי, מעביר חלון בגודל 4 בתים לאורך המערך, ומגבב את תוכן הבתים, הערך שמתקבל הוא מזהה התכונה. תכונות טופלו כבינאריות (כלומר, ערכי תכונות היו תמיד אחד, גם אם הציוץ הכיל מופעים רבים של אותו byte 4-gram). לא נעשה כל ניסיון לבצע עיבוד לשוני. יצירת וקטור התכונות הייתה חלק מפונקצית טעינה ל-Pig שמממשת שיטה זו.

לאחר מכן, בוצע אימון של המודל עבור כל אחד מקבוצות האימון עם כל הנתונים, ועבור קבוצות האימון בגודל {10,100} מיליון ציוצים בוצע אומנו מודלים גם באמצעות שיטות הרכב ע"י שליטה במספר ה-reduces כפי שהודגם במקטע ‎7.4.2.

תיאור מלא של אופן המימוש, כולל הסקריפטים שנכתבו ל-Pig המבצעים את הסינון ואת פעולת האימון והבדיקה ניתן למצוא ב- [[15](#Lin12)].

## תוצאות הניסוי

לאחר אימון המודלים, סקריפט Pig הריץ על כל מודל מאומן או הרכב מודלים מאומנים את קבוצת הבדיקה ומנה את מספר החיזויים הנכונים.



תרשים 13 - תוצאות ניסוי ניתוח רגשי, בציר האופקי מספר המסווגים ובאנכי כמות הסיווגים הנכונים (הפסים מייצגים ממוצע של 10 ניסיונות לכל תנאי ופסי הגישה תורמים 95% מרווחי ביטחון) [[15](#Lin12)].

תוצאות הניסוי מוצגות בתרשים 13 ומהן עולה כי ככל שהאימון נעשה ע"י מספר גדול יותר של דוגמאות אימון, כך עולה גם דיוק המסווג, עוד ניתן לראות כי עבור מסווג יחיד עליה חדה קרתה בין מליון ל-10 מליון דוגמאות אימון, ועליה קלה יותר מ-10 מיליון דוגמאות ל-100 מליון. פרט חשוב ומעניין נוסף מראה כי השימוש בשיטות הרכב הניבו חיזוי הגבוה משמעותית מחיזוי ע"י מסווג יחיד.

[[15](#Lin12)] לא מספק זמני ריצה של האימון, אך מציין כי בדיקות לא פורמאליות הראו כי היה קצר יותר לאמן הרכבים בגלל שמודלים נלמדים במקביל, ואין מכונה יחידה אליה כל הנתונים חייבים להיכנס.

# מימוש לבעיית זיהוי משתמשים דומים

פרק זה מציג מימוש לבעיית זיהוי משתמשים דומים באמצעות למידה חישובית בסדר גודל גדול. הוא סוקר מימוש פרקטי שבוצע באמצעות התשתית שהוצגה, ואף מראה כיצד השימוש בה הביא לשיפור ביצועים משמעותי.

## תיאור הבעיה

כזכור, הרעיון ברשת טוויטר הוא שמשתמשים מפרסמים ציוצים, ומשתמשים אחרים עוקבים אחריהם. [[14](#Goe)] מציג מימוש לבעיית המשתמשים הדומים (Similar to) באמצעות למידה חישובית. במאמר מגדירים שני משתמשים כדומים אם תוכן שאותו הם מייצרים דומה, והבעיה המתוארת היא לזהות משתמשים אלה. למשל "ליידי גאגא" דומה ל"'קייטי פרי" מאחר ששתיהן זמרות מפורסמות.

לבעיה המתוארת יש מגוון שימושים בטוויטר, למשל, משתמש שעוקב אחרי ליידי גאגא ייתכן שיתעניין גם בקייטי פרי, ועל כן ניתן להציע לו לעקוב אחריה. שימוש נוסף ביכולת הוא עבור פרסום, למשל ייתכן שנייקי תרצה להציג פרסומות למשתמשים העוקבים אחרי חברות דומות לה כמו אדידס. כמו כן, ניתן לעשות שימוש במשתמשים דומים כדי לשפר את חווית המשתמש, בכך שיוצג לו המלצות לתוכן שמעניין אותו. וקיימים וודאי שימושים נוספים.

נציין כי טרם מימוש זה היה קיים בטוויטר יישום "ייתכן שתרצה לעקוב גם אחרי..." המבוסס על אלגוריתם CosineFollow (לקריאה נוספת [[14](#Goe)]), אך כפי שיוצג בהמשך, השימוש בלמידה חישובית הגדיל את איכות ההצעות.

בעיית המשתמשים הדומים היא בעיה ידועה בספרות, אך עבור טוויטר קיימות מספר נקודות שמבדילות אותה מרוב עבודת המחקר שנעשתה בנושא. ראשית, לטוויטר מאות מיליוני משתמשים, דבר שמאלץ לבצע התאמה של כל אסטרטגיה ידועה כך שתוכל לעבוד על כמות נתונים בסדר גדול. ובנוסף, גרף המשתמשים של טוויטר מאוד דינאמי ומשתנה כל הזמן, ולכן יש לעדכן גם את הדמיון בין המשתמשים בהתאם לשינויים. שנית, יש המון פרמטרים לפיהם ניתן לקבוע דמיון בין שני משתמשים, ומובן שבבעיה בסדר גודל גדול יש קושי לקחת את כל הפרמטרים בחשבון. שלישית, מאחר ומדובר בכמות משתמשים עצומה, יש להניח כי כלל המשתמשים אינם מהווים קבוצה הומוגנית, ויש לקחת פרט זה בחשבון (למשל, לליידי גאגא יש יותר מ-30 אלף עוקבים, בעוד שלרוב הזמרים המפורסמים יש פחות מ-3000, ובכל זאת ליידי גאגא יכולה להיחשב דומה לזמרת מפורסמת אחרת).

## מימוש הפתרון

המימוש לבעיית המשתמשים התבססה על התשתית שתוארה ב- [[15](#Lin12)] ושהובאה גם במסמך זה. תהליך למידת המודל בוצע באמצעות רגרסיה לוגיסטית מבוססת SGD מקוון כפי שיושמה בחבילת התשתית של טוויטר.

יצירת דוגמאות האימון התבססה על הלוג (יומן אירועים) של היישום "ייתכן שתרצה לעקוב אחרי...". כאשר משתמש ביקר בעמוד של אחד המשתמשים שאחריהם הוא עוקב אותם נכנה "משתמש המטרה", באמצעות היישום הוצעו מספר משתמשים שייתכן כי ירצה לעקוב אחריהם שאת כל אחד מהם נכנה "המשתמש המוצע". אם המשתמש נענה להצעה אז (משתמש המטרה, המשתמש המוצע) הוגדרו כדוגמת אימון חיובית, אחרת הוגדרו דוגמת אימון שלילית. וקטור התכונות הורכב ממספר תכונות שנמצא כי יש להם תרומה משמעותית למודל הרגרסיה הלוגיסטית, תוכנות אחרות הושמטו כדי להביא לשיפור ביצועים. כמו כן, נמצא כי תכונות מסוימות הן משמעותיות עבור קבוצה אחת של משתמשים (סלבריטאים למשל), בעוד שתכונות אחרות היו משמעותיות יותר עבור קבוצה אחרת (למשל לא סלבריטאים). לכן, חולקו המשתמשים לקבוצות ועבור כל מודל אומן מודל משלו.

## תוצאות הניסוי

כדי לאמת את איכות הפתרון בצעו בטוויטר בדיקות המבוססת על בדיקה אנושית וכן בדיקות שכללו השוואה לתוצאות האלגוריתם CosineFollow לאחר הטמעת הפתרון.

בבדיקה האנושית הוצגו ל-200 משתמשים 5 דוגמאות למשתמשים שאותרו כדומים מכל קבוצה והיה עליהם להחליט אם הם דומים, שונים, או שקשה להחליט (למשל חסר עוד מידע). קביעתם נשמרה והושוותה לחיזוי המקורי (מובן שככל שמשתמשים הוגדרו פחות כידוענים כך התקבלו יותר תשובות של "קשה להחליט"). בדיקה זו נעשתה עבור הפתרון באמצעות למידה חישובית וכן עבור האלגוריתם CosineFollow. הבדיקה האנושית הראתה כי מלבד קבוצה אחת גבר בביצועיו הפתרון באמצעות למידה חישובית על האלגוריתם CosineFollow.

בנוסף, משהוטמע הפתרון, נראתה עליה של כ-38% בשימוש ביישום "ייתכן שתרצה לעקוב גם אחרי..." וכן עליה של כ-32% ביישום נוסף בשם "דומה ל..." ב-IPhone ו-28% בשימוש ממכשירי אנדרואיד. שיפור זה מומר בכל יום למיליוני מעקבים נוספים.

# סיכום

מסמך זה פתח במתן רקע ללמידה חישובית, מתוך מטרה להציג את הבעיות הכרוכות במימוש למידה חישובית בסדר גדול. לאחר מכן, הוצגו שיטות ואלגוריתמים שונים המאפשרים שינוי של אלגוריתמים קיימים, או מהווים בעצמם אלגוריתמים המהווים פתרונות לבעיית הלמידה החישובית בסדר גודל גדול. מובן שנוסף לרקע שהוצג והשיטות השונות קיים מידע רב נוסף ובאפשרותו של הקורא להרחיב ידיעותיו באמצעות ההפניות שצוינו בכל אחד מהפרקים.

מטרתו העיקרית של מסמך זה היא להציג את יישום הלמידה החישובית בסדר גודל גדול כפי שבוצעה ברשת החברתית טוויטר, וכן להראות כיצד יישום זה מניב פירות בראיה של תהליכים עסקיים. יחד עם זאת, מעט מאוד ידוע אודות מימושים של למידה חישובית בסדר גודל גדול בארגונים אחרים, ולכן אין הכרח בכך שהיישום שהוצג בהיבט של ארכיטקטורה והשיטות השונות שבהן עשה שימוש אכן צריך לשמש כבסיס מנחה למימוש פתרונות דומים.

אין ספק כי כמות המידע שנצברת ברשתות החברתיות ובכלל ברשת אינטרנט תמשיך ותגדל עם השנים, מה גם שנראה כי למידה חישובית היא בעלת עוצמה לחיזוי באמצעות נתונים אלו, אשר על כן, ארגונים יפתחו שיטות שונות ליישום הלמידה. בתקווה שהקהילה המפתחת אכן תתבסס על שיתוף ידע, ייתכן שתתגבש ארכיטקטורה מומלצת ליישום הלמידה.

# ביבליוגרפיה

x

|  |  |
| --- | --- |
| [1] | Apache Software Foundation. Apache Hadoop Wiki. [Online]. <http://wiki.apache.org/hadoop/> |
| [2] | Apache Software Foundation. Apache Pig. [Online]. <https://pig.apache.org/> |
| [3] | Apache Software Foundation. Pig UDF Manual. [Online]. <http://pig.apache.org/docs/r0.8.1/udf.html> |
| [4] | Avrim Blum, *On-line algorithms in machine learning*.: Springer, 1998. |
| [5] | Leon Bottou, "Large-Scale Machine Learning with Stochastic Gradient Descent," *Proceedings of COMPSTAT*, pp. 177-186, 2010. |
| [6] | Cheng Chu et al., "Map-reduce for machine learning on multicore," *Advances in neural information processing systems*, no. 19, p. 281, 2007. |
| [7] | Ronan Collobert, *Large Scale Machine Learning*.: PhD thesis, Université Paris VI, 2004. |
| [8] | Coursera. ML:Main. [Online]. <https://share.coursera.org/wiki/index.php/ML:Main> |
| [9] | Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer, "Online Passive-Aggressive Algorithms," *The Journal of Machine Learning Research*, no. 7, pp. 551--585, 2006. |
| [10] | Jeffrey Dean and Sanjay Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, no. 51.1, pp. 107-113, 2008. |
| [11] | Thomas G. Dietterich, "Ensemble Methods in Machine Learning," *Multiple Classifier Systems*, no. 1857, pp. 1-15, Springer 2000. |
| [12] | Nicole B and others Ellison, "Social network sites: Definition, history, and scholarship," *Journal of Computer-Mediated Communication*, no. 13.1, pp. 210-230, 2007. |
| [13] | Facebook. News Room - Company Info - Key Facts. [Online]. <http://newsroom.fb.com/content/default.aspx?NewsAreaId=22> |
| [14] | Ashish Goel, Aneesh Sharma, Dong Wang, and Zhijun Yin, "Discovering Similar Users on Twitter," 2013. |
| [15] | Jimmy Lin and Alek Kolcz, "Large-scale machine learning at twitter," *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, no. 793-804, 2012. |
| [16] | Tom M Mitchell, "Machine learning.," *Burr Ridge, IL: McGraw Hill*, no. 45, 1997. |
| [17] | Tam T Nguyen, Kuiyu Chang, and Siu Cheung Hui, "Distribution-aware online classifiers," *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Two*, pp. 1427-1432, 2011. |
| [18] | Alexander and Paroubek, Patrick Pak, "Twitter as a Corpus for Sentiment Analysis and Opinion Mining," *LREC*, 2010. |
| [19] | Herbert and Siegmund, David Robbins, "A convergence theorem for non negative almost supermartingales and some applications," *Herbert Robbins Selected Papers*, pp. 111-135, 1985. |
| [20] | Phil Simon, *Too Big to Ignore: The Business Case for Big Data*.: Wiley. com, 2013. |
| [21] | Stochastic Gradient Descent. [Online]. <http://en.wikipedia.org/wiki/Stochastic_gradient_descent> |
| [22] | W Nick and Kim, YongSeog Street, "A streaming ensemble algorithm (SEA) for large-scale classification," *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 377-382, 2001. |
| [23] | Top Sites. [Online]. <http://www.alexa.com/topsites> |
| [24] | Twitter. About Twitter. [Online]. <https://about.twitter.com/> |
| [25] | Yahoo! Inc. Apache Hadoop Tutorial, Module 4: MapReduce. [Online]. <http://developer.yahoo.com/hadoop/tutorial/module4.html#dataflow> |

x

1. <http://www.myspace.com> [↑](#footnote-ref-1)
2. <http://www.linkedin.com> [↑](#footnote-ref-2)