סיווג שרתי אינטרנט

רשתות נוירונים – למידה מדוגמאות

מאת

עידו גרינברג ויונתן צדרבאום

עבודה זו מוגשת כעבודה בהיקף של 2 יח' כמילוי חלקי של הדרישות לקראת קבלת ציון במדע חישובי

עבודה זו בוצעה בהדרכת דר. אייל כהן

יוני 2008

תמצית

הפרויקט המוצג להלן עוסק בסיווג של שרתי HTTP על פי התוצרת שלהם. לסיווג זה שימושים שונים, בעיקר בתחום אבטחת המידע. זיהוי השרת מתבסס על התגובה של האתר לבקשות שונות, המעובדות לכדי נתונים מספריים. במהלך הפרויקט מאומנות מספר רשתות נוירונים לעיבוד נתונים אלה ולזיהוי סוג השרת. לאחר שלב האימון נשמרה הרשת שהציגה את הביצועים הטובים ביותר, ושולבה בתוכנית הסופית.

תוכן העניינים

מבוא לשרתי אינטרנט

מטרת הפרוייקט

במסגרת העבודה שאפנו ליצור תוכנית מחשב המסוגלת לזהות את התוצרת של שרתי HTTP שונים. כדי להבין את משמעות המשפט האחרון במלואו, וכן בשביל להבין את הסיבה שבגינה בחרנו בנושא ייחודי זה, יש תחילה לבאר כמה מושגים:

- שרת: תוכנית מחשב המתקשרת עם משתמשים ומבצעת מטלות ספציפיות שהוקצו לה ע"י המשתמשים ושאותן היא נועדה לבצע (התוכנית משרתת את המשתמשים). דוגמאות שונות לסוגי שרתים: שרת אימייל (smtp) – המשתמש מטיל עליו שליחה של דואר אלקטרוני בשמו ליעדים שונים. שרת FTP – שרת המאפשר למשתמש לאחסן קבצים, וכן ולשלוף קבצים אל מחשבו האישי. דוגמא נוספת הינה שרת הדפסות – השרת אליו משתמשים שונים, ברחבי האוניברסיטה לדוגמא, שולחים בקשות להדפסת עבודותיהם.
- שרת HTTP: השרת הנפוץ ביותר, שרת זה מגיש ללקוח קבצים המכילים HTML, hyper text markup language, זוהי שפה כתובה אותה תוכנית הדפדפן שמותקנת במחשב האישי (internet explorer, firefox, netscape, opera etc) מבינה ושאותה היא מתרגמת לגרפיקה – מילים, טבלאות צבעים וכו'. התפקיד הנפוץ ביותר של סוג זה של שרת הוא להגיש למשתמש אתרים באינטרנט, מה שמכונה WWW-world wide web. עם זאת, שרתי HTTP משמשים גם כממשק להגדרה לשרתים אחרים, לדוגמא נתבים, מדפסות, טלפוניות איי פי, מסדי נתונים ואפילו מערכות הפעלה שלמות. הסיבה לכך נעוצה בעובדה שפרוטוקול HTTP נפוץ ביותר, אין צורך להתקין תוכנית ייעודית בשביל לשלוט בכל שרת נוסף, הדפדפן, שמותקן כבר בכל מחשב, הוא כל שנדרש. כמו כן זוהו פרוטוקול פשוט יחסית המותאם לעבודה על גבי רשת האינטרנט ומבצע את תפקידו הראשי – הצגת מידע למשתמש ושליחת מידע לשרת , ביעילות ופשטות.

שרתי HTTP, אם כן, מבצעים שלל תפקידים. מתוך כך משתמע כי תוכניות אלו מתוכנתות ע"י חברות שונות, כלומר קיימים סוגים שונים של שרתי HTTP. מטרת תוכניתנו היא לזהות את היצרן של שרת נתון.

שימושים אפשריים

ראינו, אם כן, את החשיבות הטכנולוגית של שרתים בכלל, ושל שרתי ה-HTTP בפרט. אבל מהו הערך של ידיעת סוג השרת. אנו נציג במסגרת זו שלושה שימושים מרכזיים למידע זה במסגרת עולם אבטחת המידע וניהול רשתות (אם כי קיימים עוד רבים):

- זיהוי של סוג שרת האינטרנט (WWW) הוא מידע חשוב כדי לפרוץ\להפיל את אותו שרת. שרת כאמור הוא תוכנית מחשב המתקשרת עם לקוחות. במקרה של שליחת מידע זדוני על ידי הלקוח עלול הדבר להוביל לפריצתו, אולם כל תוכנית רגישה לקלט אחר ולכן יש צורך לזהות תחילה את סוג השרת.
- לאחר חדירה לרשת פרטית של ארגון יש ערך רב לזיהוי של התקני חומרה שונים כמו נתבים, מדפסות וטלפוני איי פי. כפי שהוזכר בחלק הנ"ל לעיתים קרובות להתקנים אלו ממשקי HTTP, ולכן זיהויים עם תוכניתנו הינו אפשרי. הדבר מאפשר שלל פעולות זדוניות לתוקף – לדוגמה, אם גילה שבאיי פי (כתובת אינטרנטית המזהה התקן מסוים ברשת – מחשב, מדפסת וכו') מסוים נמצאת מדפסת הוא יכול לפעול כדי להשביתה להחליפה בשרת הדפסה מזויף. הדבר עלול לאפשר לו גישה למידע חסוי (אם מידע זה נשלח להדפסה). אותו דבר חל על שרת טלפון אי פי.
- במסגרת ניהול רשתות ניתן לעשות שימוש בתוכנה בשביל ספירת מלאי של ההתקנים ברשת החברה וכן זיהוי של התקנים לא מורשים שהותקנו על ידי עובד.

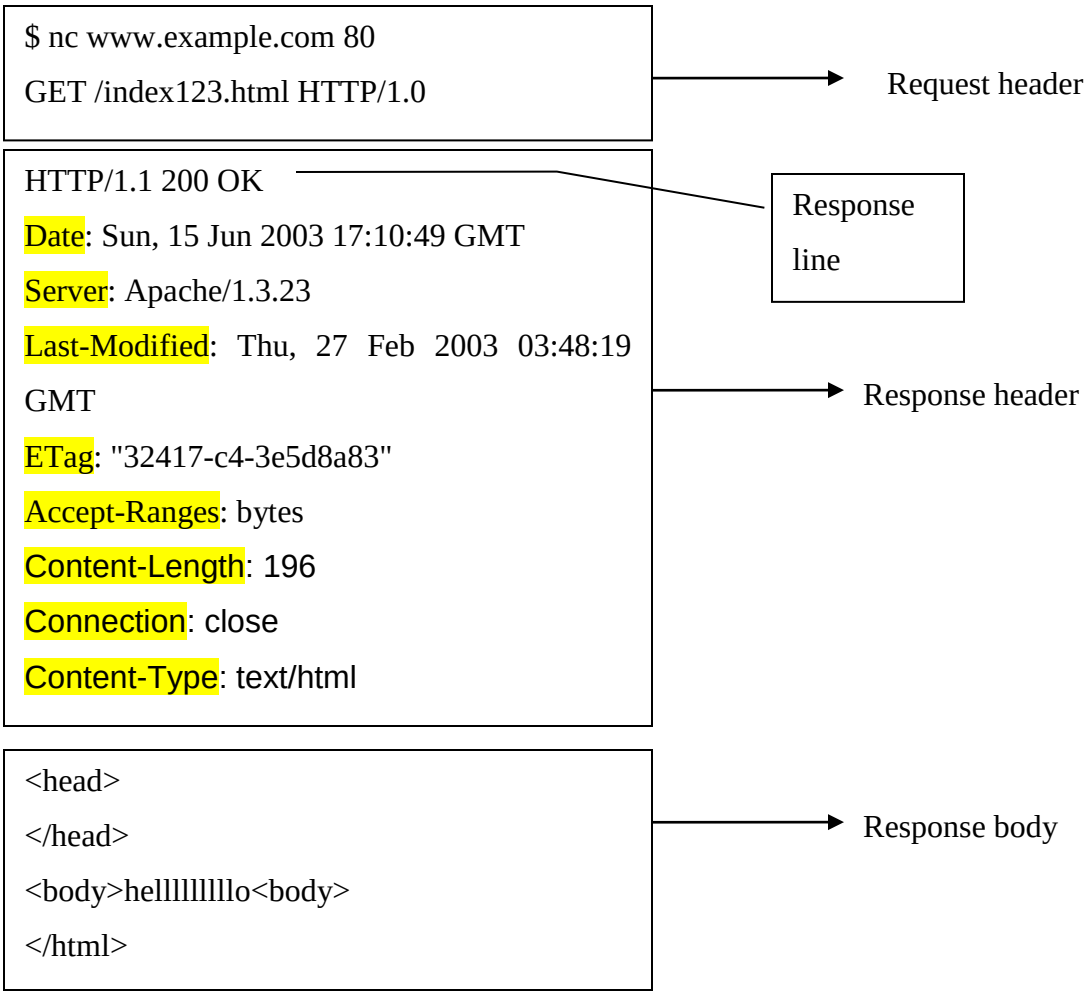
הערה חשובה: יש לציין כי מרבית השרתים מציינים את סוגם, אולם לפעמים מידע זה מושמט ולפעמים הוא שגוי מתוך אינטרסים של אבטחת מידע. אנו מזהים את סוג השרת בדרכים חלופיות כפי שיוסבר בהמשך.

סיווג שרתי אינטרנט

המשותף לכל שרתי ה-HTTP הוא השפה אותם הם מדברים – פרוטוקול ה-HTTP, זהו סטנדרט עולמי. הקורא האינטליגנט ישאל את עצמו בודאי כיצד ניתן לזהות שרת אחד מחברו אם כל השרתים כולם מדברים את אותה "שפה" – שהיא, כאמור, סטנדרט עולמי. התשובה היא שבעולם מושלם באמת לא ניתן לזהות את סוג השרת, אבל כידוע זהו עולם הרחוק משלמות. מתברר שלכל שרת "מבטא", המאפשר את זיהויו. הקורא האינטליגנט יסיק כי אם ניתן לזהות בכל זאת את סוג השרת משתמע שאותן חברות המפתחות את השרתים, מפתחות את השרתים כל אחת קצת שונה מן השניה, תוך התעלמות מן הסטנדרט ("באשמתן") או מכיוון שהסטנדרט מאפשר מרחב בישום של חלקים שונים בפרוטוקול ("באשמת" הסטנדרט).

ההבדלים המאפשרים סיווג של שרתים שונים נמצא ב-response header, זהו חלק מן התגובה של שרת לבקשה של קליינט. טקסט זה נמצא בראש תגובת השרת, לפני למשל דף HTML אם אחד כזה נשלח בתגובה (לא מחויב). הדוגמא למטה מתארת התקשרות לשרת ה-HTTP *www.example.com* בפורט 80 ולאחר מכן מתחילה התקשורת בין השרת ללקוח באמצעות פרוטוקול ה-HTTP. במקרה המתואר הקליינט מבקש את הדף /index123.html בקשה דומה הייתה יכולה להיות מיוצרת אוטומטית בדפדפן רגיל ולא באמצעות תוכנת nc ע"י כתיבת השורה:

http://www.exmple.com/index123.html, אולם בדרך זו לא ניתן לראות את התגובה המלאה של השרת (ה-response header).



* המילים המסומנות בצהוב הן headers שיקראו גם כותרות.

ההבדלים המופיעים ב-response header עליהם אנו מסתמכים בתהליך הזיהוי ניתנים לחלוקה לשלוש קטגוריות כלליות:

I. הבדלים סינטקטים: הודעות HTTP נדרשות לקיים מבנה מסוים, וזאת כדי לתקשר בקלות יחסית. עם זאת קיימים הבדלים בסדר הכותרות ובפורמט הכותרות ותוכנם.

- סדר הכותרות: למרות שפרוטוקול ה-HTTP מבחין בין שלושה סוגים של כותרות: "entity" לדוגמא הכותרת Allow, "general" לדוגמא הכותרת Date ו-"response" לדוגמא הכותרת Etag, ולמרות העובדה שהפרוטוקול מציע (לא מחייב) שסדר הכותרות יהיה קודם general אחר כך response ואחר כך entity, לא כל החברות מקיימות המלצה זו ולכן קיימים עדיין הבדלים.
- סדר הפריטים המופיעים בכותרת: לדוגמא הכותרת OPTION מכילה את כל סוגי הבקשות המורשות על ידי השרת. גם אם שני שרתים נתונים תומכים באותן בקשות, סדר הופעת הבקשות המורשות עלול להיות שונה.

II. הבדלים סמנטים: כששרת מקבל בקשה ממשתמש עליו לפרש את כוונתה בדרך מסוימת. בהתאם לפירושו, הוא בונה את המענה המתאים. השרת הוא המחליט אם הבקשה שקיבל לגיטימית, ואם לא הוא הצריך להסביר למה. כמו כן השרת מחליט איזה מידע לפלוט ב-response line, ב-headers וב-body. ההבדלים נובעים אפוא, מן הפירוש השונה של השרתים לבקשות זהות.

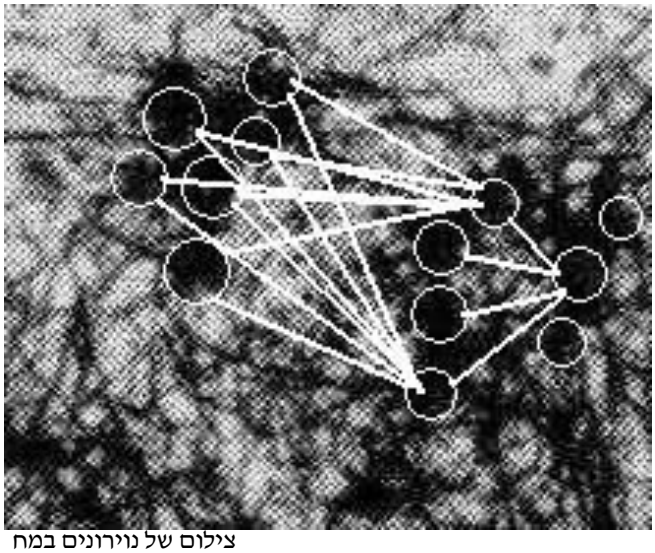
- Error codes למרות שרוב השרתים מזהים מקרה בו נשלחה אליהם בקשה לא לגיטימית שרתים שונים מפרשים באופן שונה את הבעיה בבקשה. התוצאה היא error code שונה משרתים שונים לבקשה מסוימת לא לגיטימית.
- כותרות בשימוש – למעט כותרות ספציפיות רוב הכותרות אופציונאליות ולכן נוכחותן תלויה בשרת. כך למשל, במצב בו ניתקל שרת אפאצ'י במצב בו הוא שולח הודעה עם קוד 501- סוג הבקשה לא נתמך, השרת יוסיף את הכותרת Allow שמכילה בתוכה את כל סוגי הבקשות הנתמכות, שרת ג'יגסו לעומת זאת לא יוסיף זאת.

III. הבדלים לקסיקלים: הבדלים אלו כוללים וריאציות במילים, פסוקיות, קפיטליזציה ופיסוק.

- Response Code Message- לכל תגובה של השרת מלווה קוד המתאר את סוג התגובה, למשל הצלחה (כמו בדוגמא) מבוטאת בקוד 200, דף לא נמצא – 404 וכו'. כל קוד כזה מלווה מילה או פראזה קצרה (בדוגמה -OK). בהודעה זו קיימים הבדלים בין סרברים במילים והן בקפיטליזציה (כל מילה עם אות גדולה בהתחלה או שלא וכו').
- קפיטליזציה בכותרות: למרות שפרוטוקול הHTTP מגדיר בדיוק איזה מילים רשאיות להופיע הכותרות-אין הוא מתייחס לעניין הקפיטליזציה, התוצאה היא שסרברים אחדים עלולים לפלוט :Content-length ואילו אחרים יפלטו Content-Length.

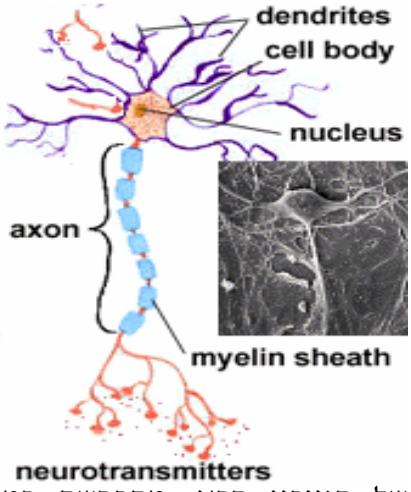
מבוא לרשתות נוירונים

קבלת ההחלטות במח



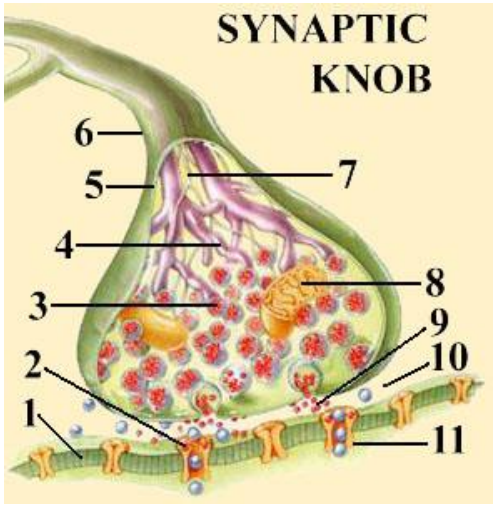
צילום של נוירונים במח

בבסיס רעיון תכנות רשתות הנוירונים עומד העיקרון לפיו עובד מח האדם. המח מכיל רשת נוירונים המורכבת מ-10^{10}-10^{12} נוירונים (=תאי עצב) המחוברים ביניהם; דוגמה פשטנית לתהליך עיבוד המידע במח היא כדלקמן: אותות חשמליים נוצרים בתאי עצב מיוחדים (תאי חישה) כתוצאה מגירוי חיצוני (למשל בידיים כשנוגעים בחפץ חם). אותות חשמליים אלה נשלחים לרשת הנוירונים במח. כל נוירון יחיד מעבד את האות המגיע אליו ושולח את האות המעובד לכ-10^{3}-10^{4} נוירונים אחרים המקושרים אליו. בסיום תהליך זה, בהתאם לאופי הגירוי, יתכן לדוגמה כי יִשלחו אותות חשמליים בחזרה לתאי העצב ביד, ויגרמו להרחקתה.



כל **נוירון** מורכב ממספר חלקים עיקריים: **דנדריטים**, האחראיים על קליטת האותות מנוירונים אחרים; **גוף התא**, האחראי על עיבוד האות החשמלי; ו**אקסון**, האחראי על העברת האות החשמלי המעובד תוך שמירה על עוצמתו. הדנדריטים והאקסונים מתפצלים פעמים רבות, כך שיתאפשר קשר של הנוירון עם אלפי נוירונים נוספים.

העברת האותות בין אקסון של נוירון מסוים לדנדריט של הנוירון הבא, מתרחשת במרחב מזערי הקרוי **סינפסה**. מן האקסון מופרשים **נוירוטרנסמיטורים** – גופים קטנים הנקלטים ע"י הדנדריט ומעבירים את האות באופן כימי. האות הכימי מתורגם לאות חשמלי העובר עיבוד בתא ומועבר הלאה.



מעבר אותות בסינפסה

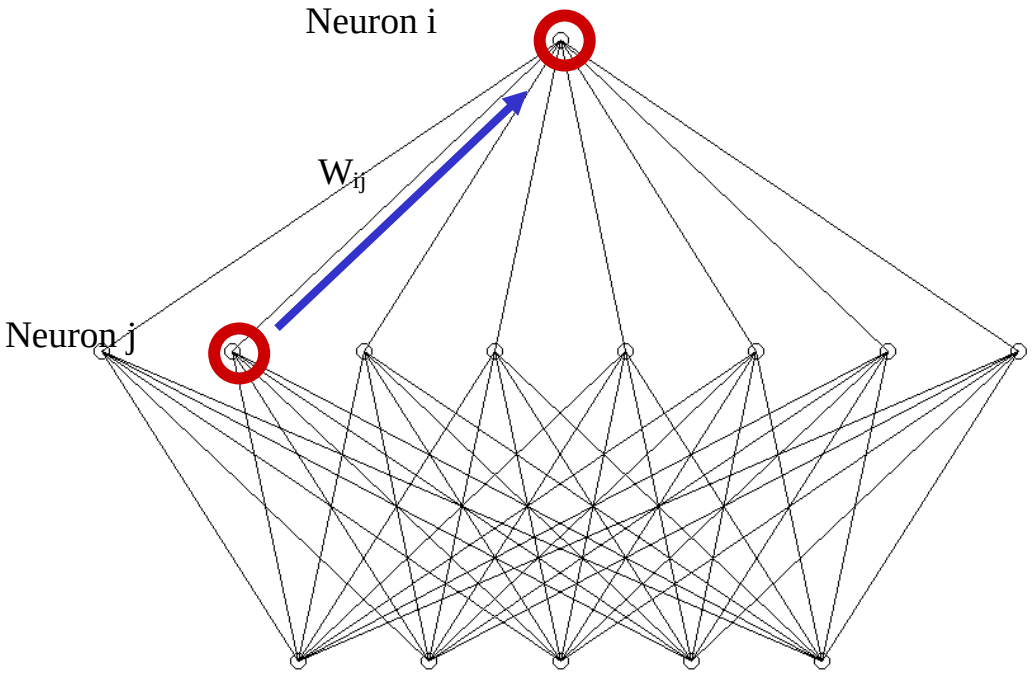
כאמור, נוירון מקבל כקלט אותות חשמליים מנוירונים רבים בו זמנית. הנוירון מבצע סכימה של כלל האותות ועיבוד של האות המתקבל. הנוירון מפרש את המידע באחת משתי דרכים עיקריות בהתאם לתפקידו וסוגו: התייחסות לעוצמת האות, המתבטאת בגודל

הפרש הפוטנציאל; או התייחסות לתדר האות, המתבטא בקצב שינוי הפוטנציאל. לאחר עיבוד המידע, התא מעביר אות חשמלי לנוירונים המקושרים אליו במידה שהתקיים תנאי כלשהו (לדוגמה, אם עוצמת האות עברה סף מסוים או התדירות גבוהה מתדירות סף מסוימת).

רשתות נוירונים – מביולוגיה למחשב

רשת הנוירונים המלכותית שואפת לחקות את המערכת המתוארת לעיל במובנים מסוימים: לרשת המלכותית מוזנת אינפורמציה (המקבילה לאותות חשמליים הנוצרים בתאי עצב מיוחדים), האינפורמציה מעובדת ברשת ומתקבל פלט (המקביל בדוגמה לאות החשמלי הנשלח אל היד). כמו כן, מוגדרים ברשת משקלים סינפטיים, עליהם יורחב בהמשך המבוא, וניתן לראותם כמקבילים למשקל האות המגיע מסינפסה מסוימת ביחס ליתר האותות הנקלטים ע"י הנוירון.

הרשתות בהן עוסק הפרוייקט נקראות feed-forward neural networks. רשתות אלה פשוטות יחסית לאחרות, ובנויות משכבות של נוירונים: שכבת קלט, שכבת פלט ושכבות ביניים. כל נוירון בשכבה נתונה מקושר באופן ישיר אל כל הנוירונים משתי השכבות הסמוכות (ולא מקושר לנוירונים בשכבה שלו או לנוירונים בשכבות שאינן סמוכות), והמידע מתקדם משכבה לשכבה.



תרשים כללי של רשת נוירונים

למעשה, רשתות הנוירונים הממוחשבות מדמות עיבוד מקבילי של נתונים, בדומה למח; זאת בניגוד לעיבוד הטורי המקובל במחשבים, בו כל אות מפעיל את האות הבא.

נוירונים בשכבה נתונה מקבלים ערכים מספריים לפי הקלט של השכבה: השכבה התחתונה, שכבת הקלט, מקבלת את הקלט של הרשת; יתר השכבות מקבלות כקלט את הפלט של השכבה הקודמת. כך המידע עובר משכבה לשכבה עד לשכבה העליונה ביותר – הפלט הסופי.

למעשה במודל החישובי לא מוגדרים הנוירונים עצמם, כי אם הקשרים ביניהם, במטריצות המסומנות ב-W ומכונות "מטריצות המשקלים הסינפטיים". בהמשך יוסבר תפקידן של המטריצות הללו בעיבוד המידע, וכיצד הן מייצגות את הקשרים בין הנוירונים.

מעבר אותות ברשת

עיבוד של אות בודד ברשת מוגדר ע"י שלושה פרמטרים: **המשקל הסינפטי (W_{ij})**, המאפיין קשר מסוים בין שני נוירונים (i,j); **ערך הסף (b_i)**, המאפיין את הקשר בין נוירון מסוים בשכבה המקבלת לכל הנוירונים בשכבה המוסרת; ו**פונקציית המעבר (f)**, המאפיינת את הקשר בין שתי שכבות ברשת.

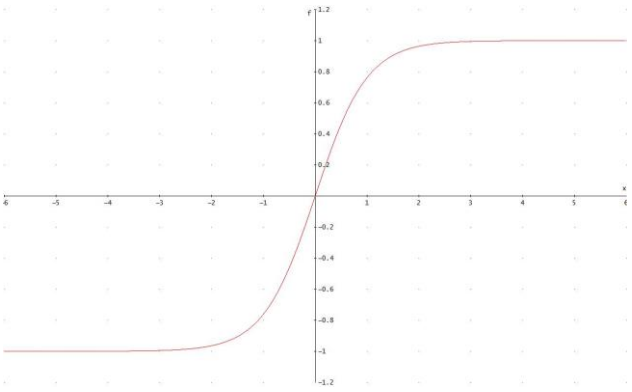
מעבר האותות בין שכבה א' לשכבה ב' מתרחש באופן הבא: כל נוירון בשכבה א' מאותחל לפי ערך איבר מתאים בווקטור p (שמוגדר תחילה לפי נתוני הבעיה: הוא יכול להכיל, לדוגמה, ערכי מניות מהשבוע האחרון, כאשר כל איבר ב-p מכיל ערך מניה ביום שונה). כפי שצוין קודם לכן, כל נוירון בשכבה ב' מחובר לכל הנוירונים בשכבה א'; במילים אחרות, ערך המידע בנוירון מושפע מכל הנוירונים בשכבה הקודמת. אם נמשיך בדוגמת המניות: ברשת נוירונים מסוימת בעלת שתי שכבות בלבד, שמטרתה להעריך ערך של מניה ביום מסוים על סמך ערכי המניה בשבוע האחרון, נבנה את השכבה התחתונה משבעה נוירונים, והשכבה העליונה תכיל נוירון בודד, שבתקווה לאחר עיבוד המידע יכיל את ערך המניה ביום שלמחרת בקירוב טוב.

חישוב ערך שמקבל נוירון נעשה לפי הנוסחה הבאה:

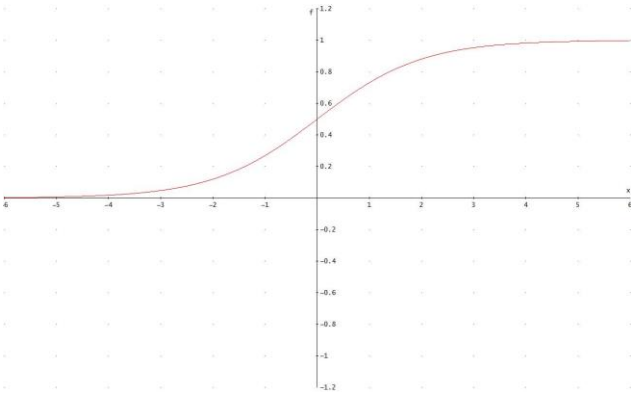
$$t = \sum[f(W \bullet p + b)]$$

נסביר את משמעותה:

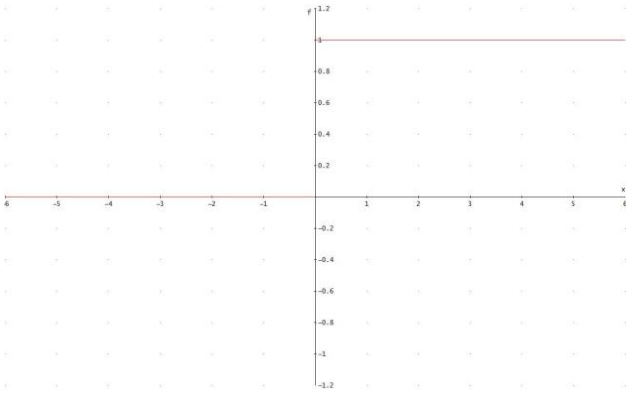
1. וקטור W שמימדיו בדוגמה 1x7 (נוירון יחיד המחובר לשבעה נוירונים מהשכבה שמתחתיו) מכיל משקלים סינפטיים: כל קשר בין הנוירון בשכבה העליונה לנוירון בשכבה התחתונה מקבל ערך מסוים; כלומר, למרות שערך הנוירון העליון מושפע מכל הנוירונים בשכבה התחתונה, הוא לא מושפע מכל נוירון במידה שווה (מעין ממוצע משוכלל). אם נמשיך להיצמד לדוגמת המניות, על ערך המניה המנובא עשוי להשפיע ערכה ביום הקודם, יותר מן הערך שהיה לה לפני שלושה ימים.
2. לערך שהתקבל ממכפלת W ב-p מתווסף ערך הסף b. לכל נוירון בשכבה המקבלת ערך סף משלו, ובדוגמה שלנו ישנו b יחיד (לנוירון יחיד).
3. לבסוף, לקבלת הפלט הסופי של הנוירון, מוכנס הערך W*p+b לפונקצית מעבר, שמטרתה לצמצם את טווח הפלטים. פונקציות המעבר המקובלות נחלקות לפונקציות רציפות, המחזירות ערך בטווח מסוים, ולפונקציות לא רציפות, המחזירות אחד משני ערכים קבועים:
   - tansig – טנגנס היפרבולי – מצמצמת את טווח הפלטים לערכים 1>t>-1.



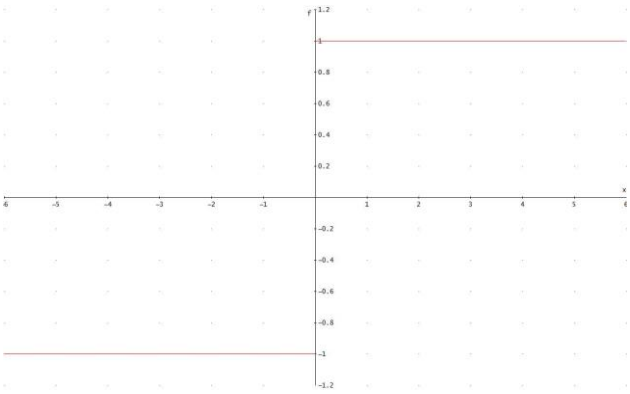
- logsig – מצמצמת את טווח הפלטים לערכים 0<t<1.



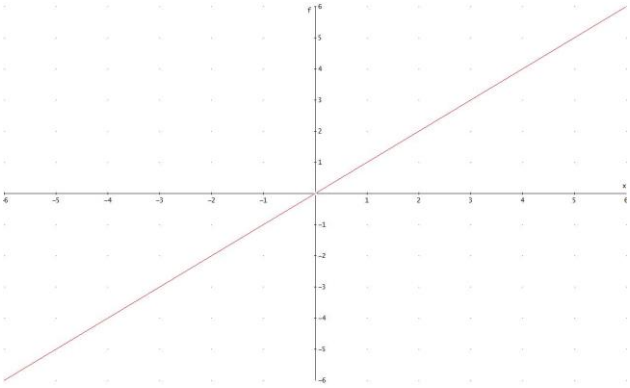
- hardlim – מחזירה 0 או 1.



- sign – מחזירה 1- או 1.



- purelin – פונקצית קו ישר.



חשיבות הפרמטרים של הרשת (W,b,f)

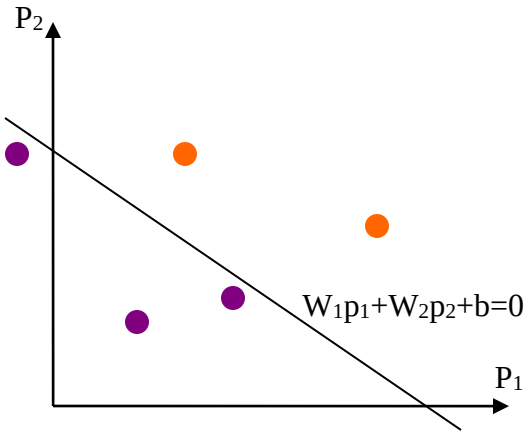
נניח רשת המקבלת שני ערכים p_1,p_2, ונדרשת להחזיר 0 עבור צמדי ערכים מסוימים ו-1 עבור אחרים. על מנת לצמצם את טווח הפלטים ל-0 או 1 נשתמש בפונקציה hardlim, המחזירה 0 לערכים שליליים ו-1 לחיוביים. הפונקציה תקבל את הגודל הבא:

$$W \bullet p + b = W_1 p_1 + W_2 p_2 + b$$

את הביטוי האחרון ניתן לרשום כמשוואת קו ישר, כאשר ערכי הקלט הם המשתנים:

$$Ax + By + c$$

במידה שהביטוי הנ"ל קטן מ-0, הנקודה (p_1,p_2) נמצאת מתחת לישר $W_1 p_1 + W_2 p_2 + b = 0$, ולהיפך. בהתאם לכך, ניתן לומר כי הפרמטרים של הרשת מגדירים את הקו המפריד בין נקודות המקבלות פלט שונה, כאשר W_1,W_2 קובעים את שיפוע הקו ו-b את נקודות החיתוך עם הצירים p_1 ו-p_2.

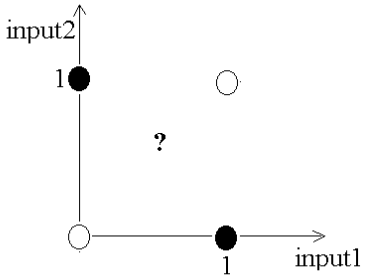


הדוגמה האחרונה מתייחסת לרשת פשוטה במיוחד, אך באופן דומה, בכל רשת מגדירים הפרמטרים W,b,f את ההפרדה בין קלטים בעלי פלטים שונים.

יש לציין כי לעיתים לא קיים קו ישר המאפשר מיון נכון של הקלטים, ובמקרה זה יש להיעזר בשכבת ביניים בין הקלט והפלט. שכבת הביניים מהווה שלב בעיבוד הנתונים: היא מאפשרת את סידור הנתונים, שימור המידע המשמעותי והדחקה של המידע שאינו רלוונטי. במילים אחרות, שכבת הביניים מאפשרת יצירה של רשת גמישה יותר, המסוגלת לפתור בעיות מסובכות.

הפונקציה XOR, לדוגמה, מקבלת כקלט שני ערכים הניתנים להצגה גרפית במערכת צירים, וקל לראות כי היא אינה ניתנת לכתיבה כרשת נוירונים ללא שכבת ביניים (משום שאין קו ישר היוצר הפרדה נכונה):

input1	input2	output
0	0	0
0	1	1
1	0	1
1	1	0

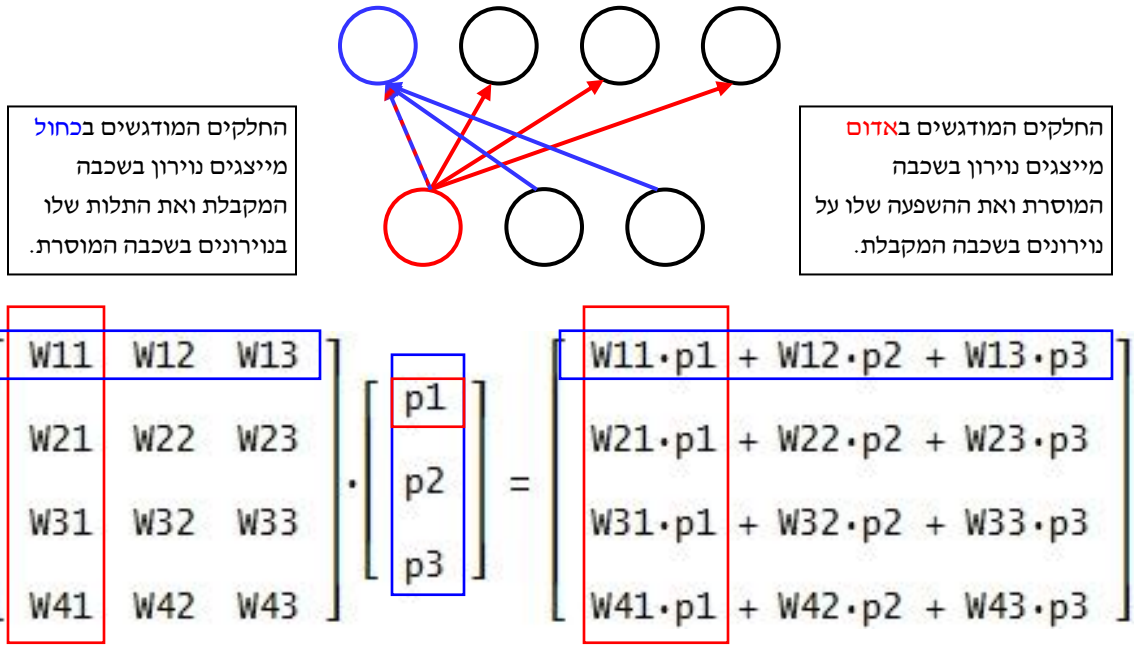


<u>הפעולה המתמטית כייצוג של הקשר בין הנוירונים ברשת</u>

הדוגמאות שתוארו עד כה הציגו מקרה בו קיים נוירון אחד בלבד בשכבה המקבלת. הנוסחה הכללית יותר לחישוב ערכי הנוירונים בשכבה נתונה:

$$t_i = \sum[\, f\,(W_{ij} \bullet p_j + b_i)]$$

במקרה הכללי קיימים N נוירונים בשכבה המקבלת. כעת W זו מטריצה ולא וקטור: כל שורה מייצגת את הקשרים בין נוירון בשכבה המקבלת לנוירונים בשכבה המוסרת (מספר השורות כמספר הנוירונים בשכבה המקבלת), וכל טור מייצג את הקשרים בין נוירון בשכבה המוסרת לנוירונים בשכבה המקבלת (מספר הטורים כמספר הנוירונים בשכבה המוסרת). b, כאמור, מאפיין כל נוירון בשכבה המקבלת, ולכן הוא כעת וקטור בעל N איברים. פונקצית המעבר אינה מושפעת מהמעבר לבעיה כללית יותר היות שהיא מאפיינת את הקשר בין שתי שכבות ולא בין נוירונים מסוימים.



$$\begin{bmatrix} W11 & W12 & W13 \\ W21 & W22 & W23 \\ W31 & W32 & W33 \\ W41 & W42 & W43 \end{bmatrix} \cdot \begin{bmatrix} p1 \\ p2 \\ p3 \end{bmatrix} = \begin{bmatrix} W11 \cdot p1 + W12 \cdot p2 + W13 \cdot p3 \\ W21 \cdot p1 + W22 \cdot p2 + W23 \cdot p3 \\ W31 \cdot p1 + W32 \cdot p2 + W33 \cdot p3 \\ W41 \cdot p1 + W42 \cdot p2 + W43 \cdot p3 \end{bmatrix}$$

כפל מטריצות ברשת

יצירה של רשת נוירונים – למידה מדוגמאות

קיימות שתי שיטות עיקריות ליצירה של רשתות נוירונים. הראשונה נקראת אלגוריתם גנטי, והיא מתבססת על דירוג ביצועיהן של רשתות שרירותיות: בכל שלב נשמרות הרשתות המוצלחות בלבד, הן מועתקות ונערכים בהן שינויים אקראיים, כך שנוצרת קבוצה חדשה ואיכותית יותר של רשתות אקראיות. כאשר אחת הרשתות מגיעה לדירוג היעד ניתן להפסיק את הלמידה.

הפרוייקט עושה שימוש בשיטה השניה – למידה מדוגמאות: על סמך דוגמאות של קלט, שעבורן קיים פלט רצוי ידוע, מדורגים הביצועים של רשת אקראית. בהתאם לתוצאות נעשים שינויים ברשת, עד להשגת תוצאות איכותיות. הלמידה מתבצעת בשלבים הבאים:

1. יצירת רשת נוירונים ראשונית:
   א. פרמטרים מוגדרים מראש: מספר השכבות, גודל השכבות ופונקציות המעבר.
   ב. משתנים הנקבעים באופן שרירותי: מטריצות המשקלים הסינפטיים (W), ערכי הסף (b).
2. תהליך הלמידה של הרשת:

א. הכנסת הקלט לרשת הנוכחית וקבלת פלט.

ב. השוואת הפלט הנוכחי לפלט הרצוי:

i. אם הפער גדול מערך מסוים ("שגיאת המטרה" – error goal), נעשה שינוי מתאים ברשת, ושלב (2) מבוצע פעם נוספת. את השינוי ברשת ניתן לעשות במספר שיטות, כאשר הפרוייקט הנוכחי משתמש בשיטה הנקראת **Error Back Propagation** (EBP).

ii. אם הפער עומד בשגיאת המטרה, ניתן לעבור לשלב הבא.

3. בחינת הרשת: תהליך הלמידה מתבצע על דוגמאות מסוימות. רשת איכותית היא רשת המסוגלת לזהות חוקיות משותפת לכל הדוגמאות, ולהכליל אותה גם על דוגמאות חיצוניות. לשם כך יש לבחון את ביצועי הרשת על דוגמאות שלא נכללו בתהליך הלמידה; תוצאות לא איכותיות מעידות בדרך כלל על אחד משני המקרים הבאים:

א. **אי-התאמה (under-fitting)** – הרשת לא הצליחה להגיע לביצועים איכותיים מספיק עבור הדוגמאות הנתונות ללמידה. סיבה שכיחה לכך היא הגדרת שגיאת מטרה גבוהה מדי.

ב. **התאמת-יתר (over-fitting)** – הרשת הגיעה להתאמה מופרזת לדוגמאות הנתונות; היא מאומנת באופן ספציפי עבור דוגמאות אלה, משתמשת בנתונים שאינם רלוונטיים לדוגמאות אחרות, ואינה מסוגלת להכליל את הלמידה למקרים נוספים. סיבה שכיחה לכך היא הגדרת שגיאת מטרה נמוכה מדי.

<u>פרמטרים ברשת המאמנת</u>

קיימים מספר פרמטרים המשפיעים על יכולת הלמידה של רשת הנוירונים:

1. **שגיאת המטרה (error goal)** – מגדירה את השגיאה המקסימלית אליה אמורה הרשת להגיע בסיום תהליך הלמידה. כאמור, פרמטר זה יקבע את מידת ההתאמה של הרשת לדוגמאות הנתונות בסיום תהליך הלמידה.

2. **גודל שכבת הביניים** – מגדיר את מספר האיברים בשכבת הביניים שבין הקלט והפלט. גודל זה קובע את כמות המידע שתכיל שכבת הביניים; כמות מעטה מדי לא תאפשר את השגת שגיאת המטרה, ואילו כמות גדולה מדי תגרום להתאמת-יתר.

3. **קצב הלמידה (learning rate)** – בכל שלב בתהליך הלמידה נעשה שינוי ברשת על מנת לשפר את התאמתה לדוגמאות. קצב הלמידה מגדיר את גודל השינוי, וקביעה נכונה של ערכו חיונית על מנת להימנע מ"מלכודות" הקיימות בשיטה של Error Back Propagation, כפי שיוסבר בהמשך.

4. **מספר הדוגמאות ואיכותן** – מומלץ להשתמש במספר רב ככל האפשר של דוגמאות, המשקפות באופן אמין את כלל המקרים, על מנת לאפשר הכללה של הרשת לכמה שיותר מקרים.

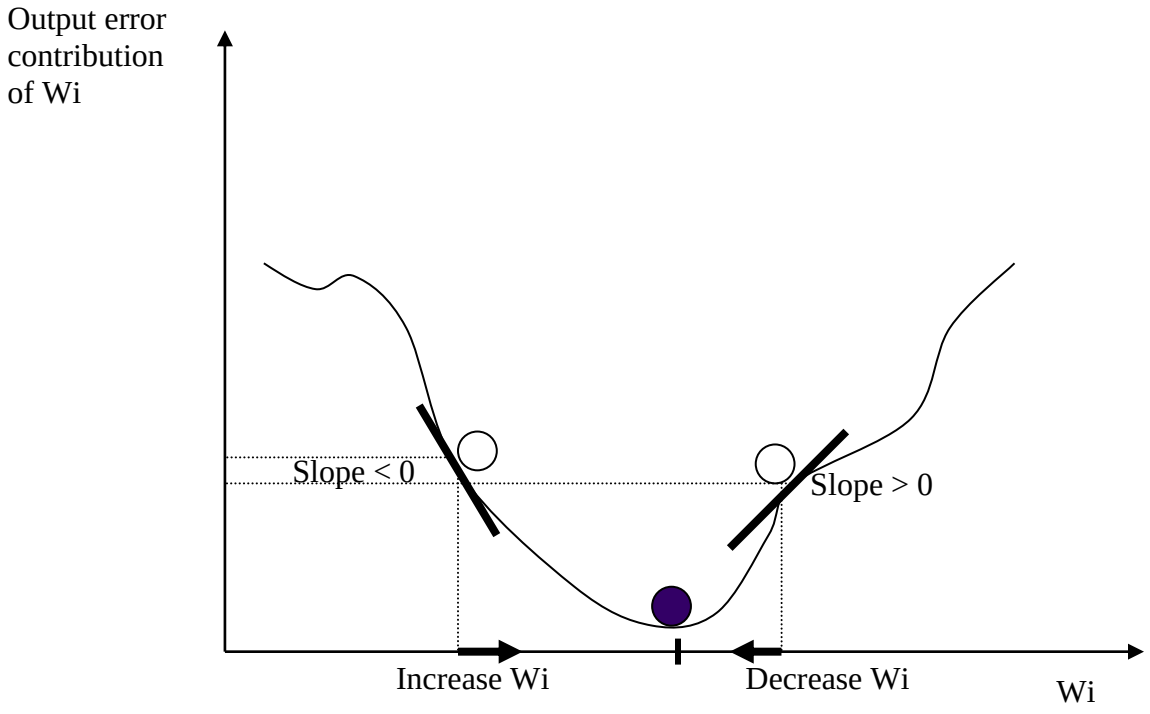
Error Back Propagation

בכל שלב בתהליך הלמידה, יש לשנות את המשקלים הסינפטיים ואת ערכי הסף של הרשת על מנת לשפר את ביצועיה; שינוי מושכל של משתנים אלה, בניגוד לשינוי שרירותי, מקל על מציאת הערכים האופטימליים, ועשוי לקצר במידה ניכרת את תהליך הלימוד כולו. Error Back Propagation (EBP) היא שיטה מקובלת לשינוי מושכל של הרשת.

השיטה מתבססת על הצגה אלגברית של השגיאה כפונקציה של משתני הרשת (W,b). עבור רשת בעלת שכבת ביניים אחת, לדוגמה, תיראה הפונקציה כך:

$$error = trueOutput - f_2(W_2 \bullet f_1(W_1 \bullet input + b_1) + b_2)$$

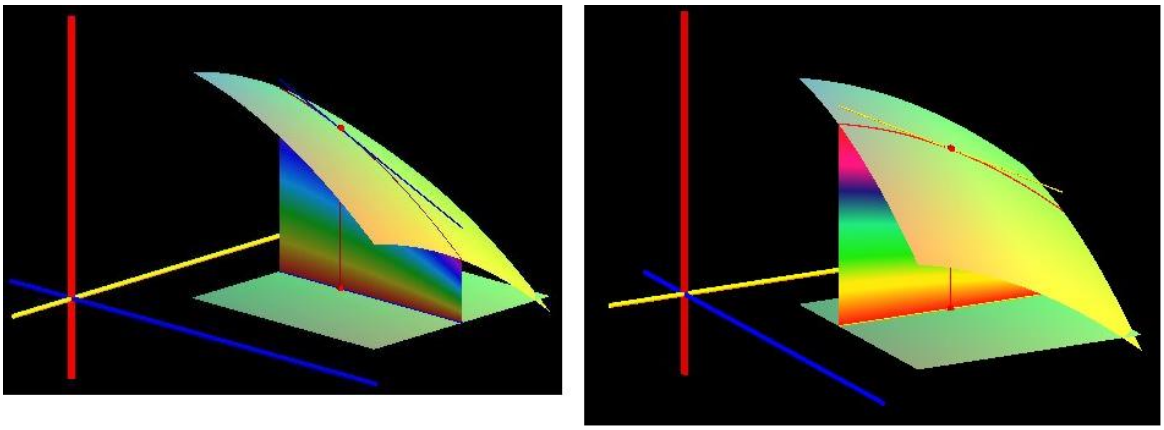
על סמך עיקרון הפרדת משתנים, ניתן לגזור את הפונקציה בנפרד עבור כל אחד ממשתני הרשת. היות שהשאיפה היא להגיע לשגיאה נמוכה ככל האפשר, יש לשנות את הגורם הנבדק נגד כיוון הנגזרת:



למעשה, ניתן לומר כי השגיאה הקטנה ביותר מתקיימת כאשר הנגזרת מתאפסת. ואולם, נגזרת פונקצית השגיאה היא ביטוי מורכב, וברוב המקרים לא ניתן להשוות אותה ל-0 ולהגיע לפיתרון אנליטי. השיטה הנומרית של שינוי ערכים נגד כיוון הנגזרת מבטלת את הצורך בפיתרון אנליטי, כאשר שינוי הערכים נפסק במידה והנגזרת קטנה מאוד (במצב זה ההנחה היא שלא ניתן להקטין עוד את השגיאה).

הגזירה של הפונקציה לפי משתנה מסוים מתוך כמה אפשריים נקראת "גזירה חלקית", והיא מומחשת בתרשימים שלהלן: התרשימים מציגים פונקציה של שני משתנים, כאשר בכל תרשים

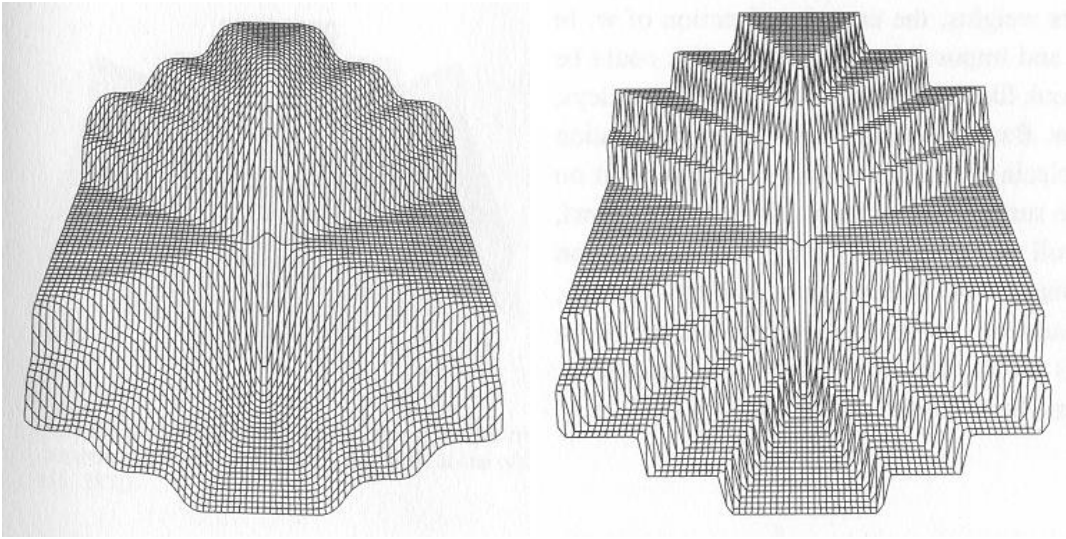
מבוצעת גזירה לפי משתנה אחר. למעשה, כאשר משתנה מסוים "מוקפא", הגרף הופך לדו-מימדי, במישור שבו ערך המשתנה המוקפא קבוע.



גזירה חלקית לפי שני משתנים שונים

"מלכודות" ב-EBP

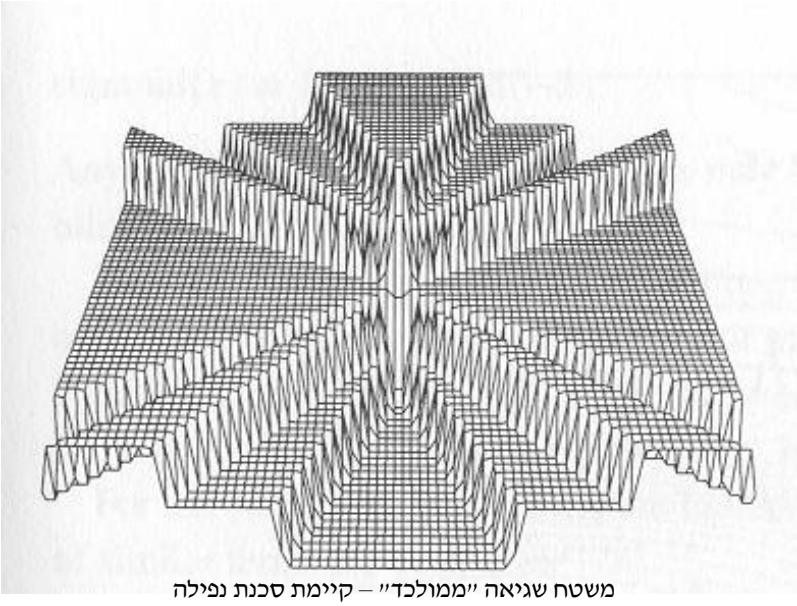
לעיתים קרובות, בפרט במקרים בהם התשובה חד משמעית (לדוגמה 0 או 1), השגיאה אינה קטנה באופן הדרגתי. במצב זה, המשטח המייצג את פונקצית השגיאה בנוי ממעין מדרגות, המקשות על תהליך הלימוד; הרשת מזהה נגזרות קטנות ומתקשה להתקדם. מסיבה זו, גם כאשר הפלט הסופי צריך להיות חד משמעי, בתהליך הלימוד ייעשה שימוש בפונקציות רציפות. בנוסף, יש להקפיד על כמות מספיקה של דוגמאות, המאפשרת התקדמות הדרגתית בלמידה:



משטח השגיאה עבור דוגמאות מעטות (ימין) ועבור דוגמאות רבות (שמאל)

במקרה בעייתי אחר, פונקצית השגיאה מגיעה למינימום מקומי. כל שינוי קטן של הרשת מגדיל את השגיאה, ואף על פי כן ניתן להגיע לשגיאה קטנה יותר. במצב זה באה לידי ביטוי חשיבותו של קצב הלמידה (learning rate): קצב למידה קטן אמנם חיוני למציאת המיקום המדויק של מינימום

השגיאה, אך קצב למידה גדול מונע נפילה למינימום מקומי. מסיבה זו, רשת האימון עושה שימוש בקצב למידה משתנה.



משטח שגיאה "ממולכד" – קיימת סכנת נפילה
לחריצים, שתמנע הגעה לשגיאה המינימלית

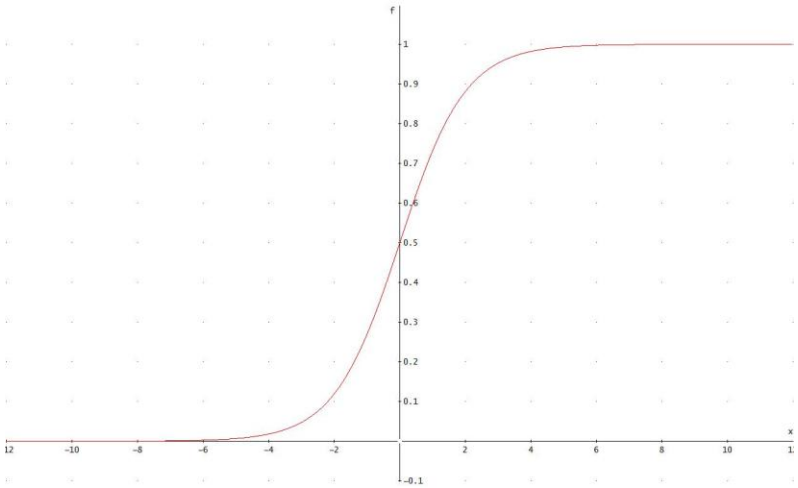
כתיבת רשת אימון ב-EBP

להלן מספר דגשים אליהם יש להתייחס בעת כתיבת רשת אימון בשיטת Error Back Propagation:

1. ראשית יש להקפיד כי כל משתני הרשת מתאימים במימדיהם לשכבות הקלט, הביניים והפלט. עבור קלט של n_1 איברים, שכבת ביניים של n_2 איברים ופלט של n_3 איברים, יהיו המימדים כדלקמן:

$W_1(n_2Xn_1)$, $b_1(n_2X1)$, $W_2(n_3Xn_2)$, $b_2(n_3X1)$

2. השיטה עושה שימוש בנגזרת השגיאה. ביטוי השגיאה מכיל את פונקציות המעבר, ומכאן שפונקציות אלה חייבות להיות רציפות (tansig, logsig) ולא קטועות (hardlim, sign).
3. יש להגדיר באופן מאוזן את שגיאת המטרה, גודל שכבת הביניים וקצב הלמידה (ראה "יצירה של רשת נוירונים – למידה מדוגמאות" – "פרמטרים ברשת המאמנת").
4. פונקציות המעבר מתאפיינות בשיפועים גדולים בקרבת ציר y, הקטנים ככל שערכי x מתרחקים מן ה-0. עבור הפונקציה logsig, לדוגמה, ההבדל בין y(0.2) לבין y(0.3) עשוי להיות משמעותי, בעוד שההבדל בין y(100) לבין y(1000) הינו זניח, שכן עבור ערכים מסדרי הגודל הללו הפונקציה שואפת ל-1. לכן, יש להקפיד שהקלט יתרכז בקרבת ה-0, ומומלץ לחלק את כל אברי הקלט באיבר הגדול ביותר, כך שכל הערכים ינועו בטווח $-1<x<1$.



5. על מנת לבחון את ביצועי הרשת, מומלץ לחלק את הדוגמאות הנתונות לקבוצת למידה וקבוצת מבחן (יחס החלוקה המקובל הוא 70% לקבוצת הלמידה ו-30% לקבוצת המבחן). תהליך הלמידה יתבסס על הדוגמאות שבקבוצת הלמידה, ובסיומו תיבחן הרשת על הדוגמאות שבקבוצת המבחן. כך ניתן לוודא כי הרשת מסוגלת להכליל את ביצועיה למקרים שאינם מקבוצת הלימוד.

התוכנה

שלבי הפעולה של התוכנה

כאמור, התוכנית נועדה לזהות את סוג השרת עליו יושב אתר מסוים, לפי תגובותיו של השרת לבקשות שונות. לפיכך, השלב הראשון בפעילות התוכנה הוא עיבוד התשובות של השרת למספרים המשמשים קלט לרשת הנוירונים. מספרים אלה מייצגים מאפיינים שונים של השרת (ראה "מבוא" – "סיווג שרתי אינטרנט"), ומאפשרים לרשת הנוירונים לקבוע את סוגו.

התוכנה מורכבת משני יישומים, המבצעים את שני השלבים בפעילותה.

שלב ראשון – שליחת בקשות ועיבוד התשובות

את השלב הראשון מבצע סקריפט (תוכנת מחשב המתבצעת פקודה אחר פקודה, ללא קומפילציה מוקדמת). לסקריפט מספר שלבי פעולה (לפירוט ראה "יצירת התוכנה" – "יצירת הסקריפט"):

1. טעינה של רשימת כתובות URL של אתרי אינטרנט.
2. שליחת בקשות מוגדרות לכל האתרים.
3. עיבוד התשובות של השרתים לערכים מספריים.
4. סידור הערכים המספריים במטריצה, כך שכל ווקטור עמודה מייצג אתר מסוים. מטריצה זו היא "פלט ביניים" – הפלט של השלב הראשון, המשמש קלט לשלב השני.

שלב שני – הכנסת הקלט לרשת הנוירונים וקבלת פלט

את השלב השני מבצעת רשת נוירונים. הרשת מקבלת את המטריצה שיצר הסקריפט (לאחר שקובץ פלט הסקריפט הומר מקובץ טקסט לקובץ של MATLAB), ומחזירה את סוגי השרתים של האתרים השונים. גם הרשת פועלת במספר שלבים:

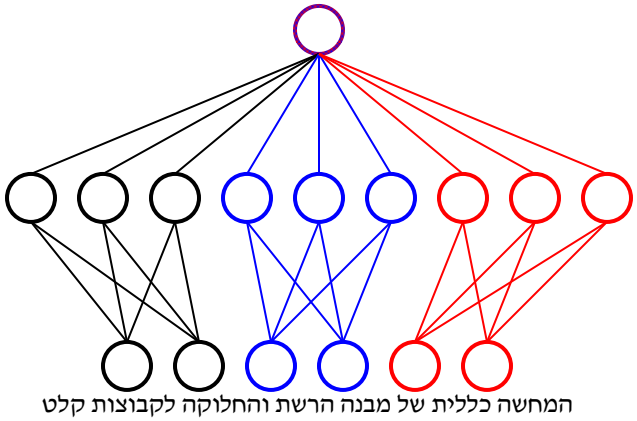
1. טעינת הקלט – המטריצה שיצר הסקריפט. כל עמודה במטריצה מייצגת אתר מסוים, וכל שורה מייצגת נתון מסוים עבור כל האתרים. שורות בהן כל האיברים זהים מיותרות, ולכן נמחקו. מספר העמודות במטריצה כמספר האתרים. מספר האיברים בכל עמודה הוא 226.
2. עיבוד הקלט לשכבת ביניים תוך שימוש במטריצה המאפסת (ראה "המטריצה המאפסת" – תת הפרק הבא). המעבר נעשה באמצעות פונקצית המעבר "tansig". מספר האיברים בכל עמודה (המייצגת אתר) בשכבת הביניים הוא 339.
3. עיבוד שכבת הביניים לשכבת פלט, שבה לכל עמודה שלושה איברים. כל איבר מייצג את ההסתברות להשתייכות השרת לסוג מסוים, וערכו נע בין 0 ל-1 (שימוש בפונקצית המעבר "logsig").
4. האיבר הגדול ביותר בכל עמודה בשכבת הפלט מכתיב את הפלט הסופי – סוג השרת עליו יושב האתר. בנוסף, בהתאם לשלושת הערכים, מחושבת ההסתברות להערכה מוצלחת של התוכנה (מדד ל"מידת הביטחון" של התוכנה במסקנה שלה).

המטריצה המאפסת

יש לשים לב למאפיין אחד של הקלט של רשת הנוירונים – ניתן לחלקו לקבוצות, בהתאם לבקשות שנשלחו לאתרים (לכל אתר נשלחו מספר בקשות, וכל תשובה עובדה לקבוצת ערכים המהווה חלק מן הקלט). הרשת מעבדת את הנתונים בכל קבוצה בנפרד ומעבירה אותם לקבוצת נוירונים מתאימה בשכבת הביניים. כך נוצרת שכבת ביניים שגם בה כל קבוצת נוירונים מייצגת בקשה מסוימת.

המבנה המתואר לעיל אינו מאפיין רשת סטנדרטית, בה כל נוירון מקושר לכל הנוירונים מן השכבות הסמוכות (fully-connected). ברשת זו מנותקים קשרים מסוימים, וניתן למעשה לראות זאת כחלוקה של הרשת למספר תת-רשתות.

ההנחה העומדת מאחורי עריכה זו של הרשת, היא שרצוי לעבד תחילה כל קבוצה בקלט בנפרד, ורק לאחר מכן להשתמש בכל הקבוצות המעובדות על מנת לגבש מסקנה סופית. יש לזכור כי הקלט הראשוני בנוי כך שניתן יהיה ליצור אותו בקלות באמצעות סקריפט מתאים ומספר פעולות מתמטיות פשוטות, ולא בהכרח באופן שיקל על הוצאה ישירה של המסקנות מתוכו. לכן יש לעבד אותו באופן מקומי לפני העיבוד הכולל אשר יוביל לפלט הסופי.



המחשה כללית של מבנה הרשת והחלוקה לקבוצות קלט

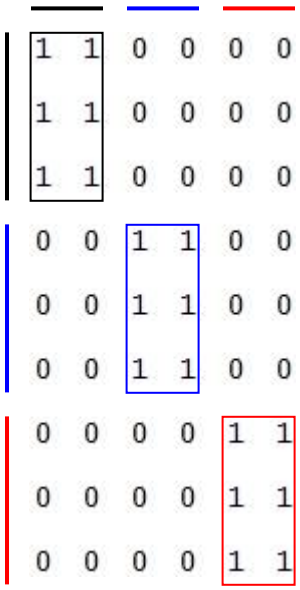
המטריצה המאפסת

כבר ראינו כי במטריצת המשקלים הסינפטיים, האיבר W_{ij} מקשר בין הנוירון j מהשכבה התחתונה לבין הנוירון i מהשכבה העליונה. לפיכך, על מנת לנתק את הקשרים הבלתי רצויים, יש לאפס את האיברים המתאימים במטריצה W. בדוגמה שבשרטוט, ברצוננו לנתק את הנוירונים $j=3,4$ (בכחול) שבשכבה התחתונה מכל הנוירונים שבשכבת הביניים, למעט הנוירונים $i=4,5,6$. לכן, כל האיברים של W שערכי האינדקס שלהם $3\leq j\leq 4$ וגם $\{1\leq i\leq 3$ או $7\leq i\leq 9\}$, יתאפסו.

כל קבוצת נוירונים בשכבה התחתונה מקושרת לקבוצת נוירונים מסוימת בשכבה העליונה, כך שלמעשה יש לאפס את מרבית הקשרים במטריצה W. על מנת לעשות זאת, ניתן לכפול את W במטריצה שאיבריה מקבלים את הערכים 0 או 1 – לא כפל מטריצות, אלא כפל של כל שני איברים מתאימים זה בזה – כך שאיברים מסוימים יתאפסו והיתר לא ישתנו.

להלן תרשים המציג את המטריצה המאפסת. ערכו של איבר, המקשר בין נוירונים מאותה הקבוצה, יהיה 1 (משום שהאיבר המתאים במטריצה W נשמר). מודגשים בתרשים האיברים האחראיים על

הקשרים בקבוצה השחורה, בקבוצה הכחולה ובקבוצה האדומה. לעומתם, האיברים המקשרים בין נוירונים מקבוצות שונות – ערכם 0.



שימוש בתוכנה

על מנת להשתמש בתוכנה, יש לבצע מספר פעולות פשוטות:

1. הרצת הסקריפט (אפשרית במערכת ההפעלה לינוקס או במערכת ההפעלה חלונות באמצעות תוכנת העזר Cygwin). לדוגמה ב-bash:

   ./h.sh www.example.com > results

2. המרת קובץ הטקסט שיוצר הסקריפט לקובץ של MATLAB (ניתן לעשות זאת באמצעות התוכנה notepad++ במערכת ההפעלה חלונות).

   Open → save as → M file.

3. טעינת הקובץ המומר ל-MATLAB והפעלת הפונקציה Server עם הקובץ המומר כקלט.

סיכום פעילות התוכנה

לסיכום, התוכנה מקבלת כתובת URL של אתר אינטרנט, שולחת לו מספר בקשות, מעבדת את התשובות לערכים מספריים ומחזירה לפי הערכים הללו את סוג השרת עליו יושב האתר.

להצגה גרפית של אופן פעילות התוכנה ראה "נספחים" – "דיאגרמת קשרים של התוכנה".

יצירת התוכנה

כאמור, התוכנה מורכבת משני יישומים, המבצעים את שני שלבי העבודה. היישום הראשון הוא הסקריפט, היוצר את הקלט עבור היישום השני, שהוא רשת הנוירונים.

יצירת הסקריפט

הסקריפט נכתב ופֻתַח ידנית (בניגוד לרשת הנוירונים). מטרתו לקבל רשימה של כתובות URL של אתרי אינטרנט, ולהחזיר מטריצה אשר תשמש קלט לרשת הנוירונים. הוא מבצע מספר פעולות, אשר צוינו כבר בפרק הקודם, ומוסברות כאן בהרחבה:

1. טעינת רשימת כתובות ה-URL.
2. שליחת בקשות לכל האתרים. המטרה היא לבדוק את התנהגותו של השרת במקרים שונים, חלקם סטנדרטיים ואחרים אינם שגרתיים (ראה "מבוא" – "סיווג שרתי אינטרנט").
3. עיבוד התשובות לערכים מספריים. כל תשובה מכילה כאמור מספר Headers (כותרות), ועבור כל Header נבדקים מספר נתונים:

א. מיקום ה-Header ביחס ליתר (למעשה מספר סידורי. אם ה-Header אינו מופיע יתקבל הערך 0).

ב. מספר התווים בתוכנו של ה-Header .

ג. מספר האותיות הגדולות (Capital Letters) ב-Header (כותרת+תוכן).

כמו כן, עבור בקשות הצפויות לגרום לשגיאה, נבדקים שני נתונים נוספים:

א. מספר התווים בהודעות השגיאה.

ב. קוד השגיאה – מוגדר ווקטור בו כל איבר מייצג קוד מסוים. ערכם של כל האיברים בווקטור 0, למעט האיבר המייצג את הקוד המתאים, המקבל 1.

להלן הבקשות הנשלחות:

1) GET – בקשה רגילה לקבלת דף האינטרנט.
2) GET + בקשת כתובת URL ארוכה מהמותר – צפוי אחד מקודי השגיאה הבאים: 200, 202, 400, 404, 406, 413, 414, 500, אחר.
3) GET + בקשת כתובת URL שאינה קיימת – צפוי אחד מקודי השגיאה הבאים: 301, 302, 307, 400, 403, 404, 406, 500, אחר.
4) HEAD – בקשה לגיטימית ל-HEADERS ללא התוכן.
5) OPTIONS – בקשה לקבלת רשימת הבקשות המותרות (ראה "מבוא" – "סיווג שרתי אינטרנט" – "I. הבדלים סינטקטים"). מוגדר ווקטור בעל 8 איברים – כל איבר מייצג בקשה מסוימת. ערך האיבר הוא מספרה הסידורי של הבקשה ברשימה (בקשה שלא מופיעה מקבלת 0).

6) DELETE – בקשה למחיקת דף האינטרנט – צפוי אחד מקודי השגיאה הבאים: 202, 400, 401, 403, 405, 406, 407, 412, 417, 500, 503, אחר. בנוסף, מוגדר ווקטור בעל 8 איברים לרשימת הבקשות המותרות, בדומה לבקשה הקודמת (OPTIONS).

7) בקשה לא מוגדרת (מילה שרירותית) – הבדיקות זהות לאלו של הבקשה הקודמת (DELETE).

8) GET + ציון גרסת HTTP שאינה קיימת – צפוי אחד מקודי השגיאה הבאים: 200, 202, 400, 406, 409, 412, 417, 500, 501, אחר.

4. יצירת מטריצה המבטאת את תשובותיהם המעובדות של השרתים. הנתונים המספריים שהתקבלו מעיבוד התשובות של שרת, מסודרים ברצף ויוצרים עמודה במטריצה (עמודה לכל שרת). לפיכך, כל שורה מייצגת את ערכיו של נתון מסוים לכל האתרים.

• *טבלה המתארת את תכנון עיבוד הנתונים בסקריפט מצורפת כנספח.*

יצירת רשת הנוירונים

רשת הנוירונים מפותחת אוטומטית. למעשה, מדובר בקביעת ערכים למטריצות המשקלים W ולערכי הסף b, באמצעות השיטה Error Back Propagation ללמידה מדוגמאות (ראה "מבוא" – "יצירה של רשת נוירונים – למידה מדוגמאות").

בשיטה זו, על המשתמש לספק לרשת קלט ופלט, וכן להגדיר מספר פרמטרים. הרשת המאמנת מבצעת את יתר העבודה, לרבות בחינת הרשת הסופית.

<u>פעולות המשתמש</u>

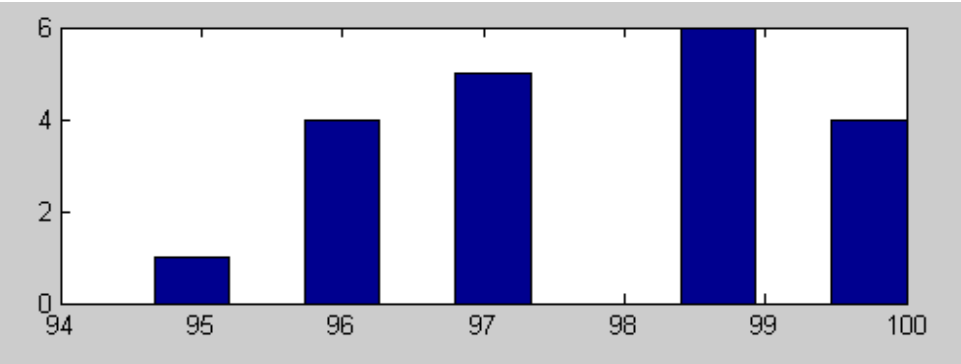
1. **מציאת אתרים לצורך יצירת קלט ופלט** – שליחת בקשות שונות, חלקן יוצא דופן, לאתרים באינטרנט אינה חוקית ונחשבת ל"תקיפה" של אתרים אלה. לכן, לצורך יצירת הקלט והפלט נעשה שימוש ברשת אתרים פנימית של חברת המחשבים CheckPoint.
2. **יצירת קלט ופלט** – הקלט נוצר על ידי הסקריפט המהווה חלק מן התוכנה. הפלט נוצר באמצעות תוספת לסקריפט זה, אשר הוסיפה לכל עמודת קלט את הפלט הרצוי (כל עמודה מייצגת אתר אחד). הפלט כאמור מורכב משלושה איברים, המייצגים את ההסתברות שאתר מסוים ישתייך לשרת מסוים. לכן, הפלט הרצוי לכל אתר הוא 0,0,1 ; 0,1,0 או 1,0,0.
3. **הגדרת פרמטרים של הרשת המאמנת** – הפרמטרים העיקריים מוצגים בפרק המבוא. בנוסף אליהם, יש להגדיר את אופן חלוקת הקלט לקבוצות (לצורך הגדרת המטריצה המאפסת, המוסברת בפרק הקודם): מוגדר ווקטור בו כל איבר מייצג את מספר האיברים בקבוצה המתאימה בקלט (גודל הווקטור כמספר הקבוצות בקלט, השווה למספר הבקשות שנשלחו לכל אתר).

<u>פעולות הרשת המאמנת</u>

הרשת המאמנת פועלת במספר שלבים, תוך התבססות על הדגשים המופיעים בפרק המבוא (״יצירה של רשת נוירונים – למידה מדוגמאות״ – ״כתיבת רשת אימון ב-EBP״). הקוד המלא של הרשת מופיע בנספחים.

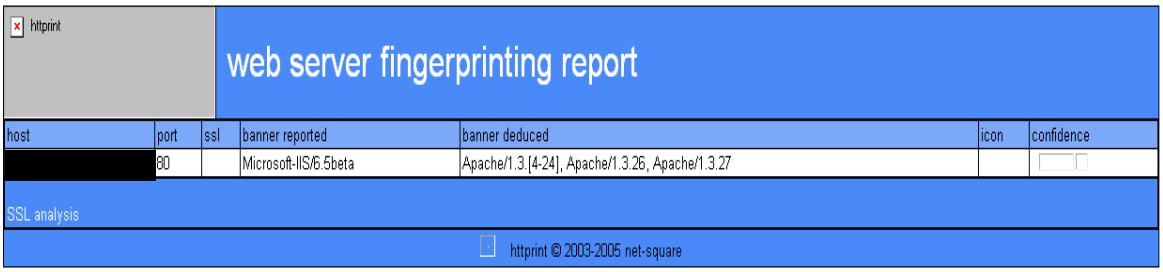
תוצאות

במסגרת הפרויקט בנינו בסיס נתונים המכיל 249 שרתי http (ראה "יצירת התוכנה" – "יצירת רשת הנוירונים" – "פעולות המשתמש") יחד עם מאפיינים שונים שלהם. מאגר נתונים זה חולק ביחס של 7:3 כשרובו מנוצל בתהליך לימוד הרשת, ושאריתו בתהליך בדיקת ההצלחה שלה. אימנו 20 רשתות נוירונים שונות, המהוות 20 מוחות שונים המיועדים לזהות סוגי שרתים. הרשתות שונות אחת מן השניה מכיוון שחלוקת מאגר הנתונים לקבוצות לימוד ומבחן והגדרת הרשת הראשונית (אתחול הערכים הראשוניים של הרשת) נעשות באופן רנדומאלי לפני הלימוד של כל רשת חדשה. להלן תוצאות הבדיקה:



ניתן לראות כי הרשתות מתפקדות היטב כשממוצע ההצלחה בזיהוי האתרים עומד על 97.86.
לתוכנית הסופית בחרנו כמובן ברשת שביצעה זיהוי מושלם לכל השרתים בבדיקה (היו ארבע כאלה בניסוי זה, כפי שניתן לראות).

רשת זו זיהתה "בעולם האמיתי" שרת שהציג עצמו כשרת Microsoft IIS (99.8%) Apace, וזאת בדומה לתוכנה מקצועית אחרת, המבצעת גם היא זיהוי סוגי שרתים, אולם בטכניקה שונה.



תוצאות התוכנה המקצועית שהופעלה על השרת

סיכום וביקורת

בתחילת הפרויקט שאפנו ליצור תוכנה שתוכל לזהות את התוצרת המדויקת וכן את הגרסה של שרת מסוים. הנחתנו הייתה כי שרתים שונים ניתנים לסיווג על פי אופן תגובתם לנוכח פעולות שונות מצד הלקוח. לאחר חקר קטן, בעזרת מקורות שונים באינטרנט, התבררה הנחה זו כנכונה. החלטנו אפוא כי קיימת כאן אפשרות סבירה לניצול מוצלח של הכלי שלמדנו בשיעורי מדע חישובי – רשתות נוירונים. ליקטנו מידע רב אודות שרתים שונים בחוות השרתים של חברת המחשבים CheckPoint, ללא ידיעה מוקדמת אם אכן כל פיסת מידע תבוא לידי שימוש. לאחר מכן עובד המידע למטריצה מספרית, שורות בהן כל הערכים היו זהים נמחקו.

השלב הבא, שלב הפיתוח, נפתח בצורך לעצב רשת נוירונים מתאימה. החלטנו כמובן כי נבחר ברשת הלומדת לפי דוגמאות. לצערנו נאלצנו בשלב זה לצמצם את הפרויקט לזיהוי תוצרת השרת בלבד, ללא הגרסה המדויקת, וזאת מכיוון שמאגר הנתונים שלנו איננו רחב מספיק.

הרשת הציגה ביצועים יפים מאוד בשלב הבדיקה. באופן טבעי, אנו מציעים שבהמשך תורחב העבודה לזיהוי הגרסה המדויקת של השרת. אנו מציעים כי פרויקט ההמשך יעשה זאת באופן הבא: יבנו שתי רשתות נוספות, אחת לשרתי IIS ואחת לשרתי Apache. רשתות אלו יאומנו לזיהוי הגרסה המדויקת של השרת. התוכנית הסופית תשלב בין שני העבודות; תחילה תזוהה תוצרת השרת ולאחר מכן הגרסה בעזרת הרשת המתאימה.

הקושי העיקרי בפרויקט היה תכנון עיבוד תגובות האתרים לכדי נתונים מספריים. נעזרנו בחומרים שמצאנו באינטרנט על מנת לאתר את סוג הנתונים החיוניים וכן את סוג הבקשות העשויות לגרום להתנהגות ייחודית מצד השרת, ופיתחנו דרכים להצגתם כקבוצות מספרים.

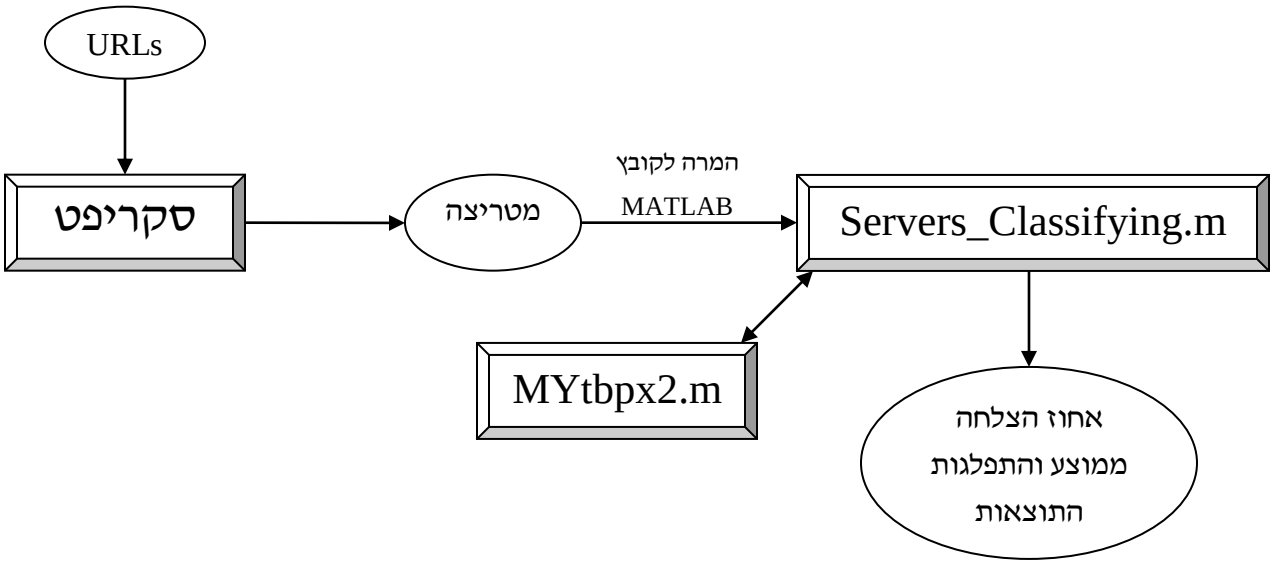
ולסיום, ברצוננו להודות לחברת המחשבים CheckPoint, אשר סייעה לנו בתכנון עיבוד הנתונים, פיתרון תקלות טכניות וכתיבת הסקריפט.

ביבליוגרפיה

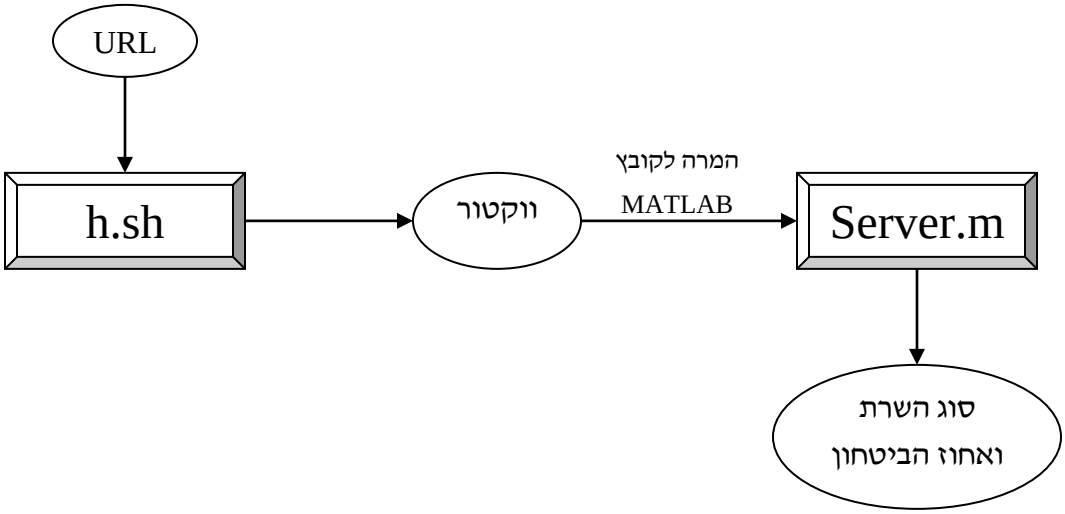
1) Hmap: A Technique and Tool for Remote Identification of HTTP Servers/ Dustin Lee
2) Blackhat Presentation: Identifying Web Servers
3) Httprint: Introduction to fingerprinting http://www.net-square.com/httprint/httprint_paper.html
4) List of Http Headers: http://en.wikipedia.org/wiki/HTTP_headers
5) List of HTTP status codes: http://en.wikipedia.org/wiki/List_of_HTTP_status_codes
6) Reading material and presentations by Dr. Eyal Cohen, HEMDA

נספחים

דיאגרמת קשרים של הרשת המאמנת



דיאגרמת קשרים של התוכנה



תכנון עיבוד הנתונים בסקריפט

הטבלה הבאה היא תכנון של עיבוד התגובות של השרתים לנתונים מספריים. כל שורה מתארת בקשה מסוימת, ואת כל הנתונים הנבדקים עבור התגובה לבקשה זו.

	בקשה	הערות	סה"כ איברים	מיוחדת לבקשה		
1	legitimate		3N	-		
2	GET - too		3N+9	קוד שגיאה - ווקטור בעל 9 איברים, כל איבר מייצג קוד שגיאה מסוים. כל האיברים מתאפסים למעט זה המייצג את הקוד המוצג (איבר זה מקבל 1). הקודים: 4,202,002 04,404,00 414,314,6 השגיאה אחר,005,	בינתיים ללא בדיקה של מילות	
3	GET - non		3N+9	בדיקה זהה לבקשה הקודמת, עם 9 קודים אחרים: 3,203,103 04,004,70 604,404,3 other,005,		
4	legitimate	צפוי להניב תוצאות זהות לGET הרגיל	3N	-		
5	OPTIONS		3N+8	מיקום המוצגות בALLOW - וקטור בעל 8 איברים, כל איבר מייצג METHOD מסוימת, ערך האיבר הוא מיקום המופיעה ברשימה METHOD (אפס פירושו לא		
6	DELETE	not permitted in most cases	3N+20	זהה לבקשה מס' 2. 21 קודים אפשריים: 4,004,202 04,304,10 704,604,5 , 5,714,214 ,005,30 אחר	זהה לבקשה מס' 5 - וקטור בעל 8 איברים המתייחס לALLOW	

7	not defined	something arbitrary like TEST. use the same word for all the sites!	3N+20	זהה לבקשה מס' 5 - וקטור בעל 8 איברים המתייחס לALLOW	זהה לחלוטין לבקשה הקודמת (אותם קודי שגיאה)	
8	GET - non	for example HTTP/9.8	3N+10		זהה לבקשה מס' 2. 01 קודים אפשריים: 4,202,002 04,604,00 ,9 5,714,214 105,00,אח ר	
1-8	בדיקות משותפות לכל הב		3N	מיקום הHEADER - RS וקטור בעל N איברים, כל איבר מייצג HEADER מסוים, ערך האיבר הוא מיקום הHEADER (אפס פירושו לא נמצא)	מספר תווים לHEADER - R וקטור בעל N איברים כאשר כל איבר מייצג HEADER מסוים, וערכו הוא מספר התווים בתוכן של הHEADER	אותיות גדולות - וקטור בעל N איברים, כל איבר מייצג HEADER מסוים, ערך האיבר הוא מספר האותיות הגדולות (CAPITAL) (S בHEADER עצמו + R בתוכן. לדוג': Server: Apache - 2 אותיות גדולות
סה"כ	כל הבקשות	הפלט - ווקטור עמודה ארוך המורכב מן הווקטורים של כל הבקשות, מונחים אחד אחרי השני	24N+76			
	הפלט הסופי - מטריצה בה כל עמודה היא ווקטור בעל 42N+76 איברים המייצג אתר מסוים; כל שורה היא ווקטור המייצג נתון מסוים בכל האתרים שנבדקו.	*אם יותר נוח - אפשר להפוך בין העמודות והשורות (ואנחנו כבר נסובב את המטריצה במאטלאב)				

קוד הרשת המאמנת

Servers_Classifying

```
clear,close all

% 1. General Parameters
experiments_num=10;
train_size=0.7;
middleLayer=1.5;
pSizes=[51 61 61 51 59 72 72 62];
epochs_between_display=40;
epochs_num=200;
error_goal=0.1;
learning_rate=0.01;

% 2. Define data
   % 2. A. Load data
   load database.m
   % 2. B. Sort data according to output & Rotate
   data=sortrows(sortrows(database,491),492)';
   % 2. C. Compare numbers of different servers
   data=data(:,245:493);
   % 2. D. Seperate to p & t
   pp=data(1:489,:);
   p=[];
   t=data(490:492,:);
   % 2. E. Resize in order to have values between 0-1
   pSizes2=pSizes;
   erased_rows=[];
   for i=1:size(pp,1)
      if max(pp(i,:))~=min(pp(i,1))
         p=[p;pp(i,:)/max(pp(i,:))];
      else
         erased_rows=[erased_rows i];
            if (i>0 && i<=pSizes2(1))
               pSizes(1)=pSizes(1)-1;
            else
             if (i>pSizes2(1) && i<=sum(pSizes2(1:2)))
               pSizes(2)=pSizes(2)-1;
             else
             if (i>sum(pSizes2(1:2)) && i<=sum(pSizes2(1:3)))
               pSizes(3)=pSizes(3)-1;
               else
            if (i>sum(pSizes2(1:3)) && i<=sum(pSizes2(1:4)))
               pSizes(4)=pSizes(4)-1;
               else
            if (i>sum(pSizes2(1:4)) && i<=sum(pSizes2(1:5)))
               pSizes(5)=pSizes(5)-1;
               else
            if (i>sum(pSizes2(1:5)) && i<=sum(pSizes2(1:6)))
               pSizes(6)=pSizes(6)-1;
            else
            if (i>sum(pSizes2(1:6)) && i<=sum(pSizes2(1:7)))
               pSizes(7)=pSizes(7)-1;
               else
```

```
            if (i>sum(pSizes2(1:7)) && i<=sum(pSizes2(1:8)))
                pSizes(8)=pSizes(8)-1;
            end
            end
            end
            end
            end
             end
             end
            end
        end
    end
    pp=[];
    pSizes2=[];
    % 2. G. Calculate middle layers's size
    middleSize=round(middleLayer*size(p,1));
    % 2. H. Initialize train & test groups
    p1=p(:,1:round(size(p,2)/3));
    t1=t(:,1:round(size(t,2)/3));
    p2=p(:,round(size(p,2)/3)+1:2*round(size(p,2)/3));
    t2=t(:,round(size(t,2)/3)+1:2*round(size(t,2)/3));
    p3=p(:,2*round(size(p,2)/3)+1:size(p,2));
    t3=t(:,2*round(size(t,2)/3)+1:size(t,2));
    sizeTrain=round(train_size*size(p,2)/3);
    p_train=zeros(size(p,1),3*sizeTrain);
    t_train=zeros(size(t,1),3*sizeTrain);


% 3. Experiment = train a net and evaluate it
results=[];
for experiment=1:experiments_num
    experiment

    % 3. A. Define train & test groups
    p_test=[];
    t_test=[];
    for k=1:3
        eval(['pDouble=p' num2str(k) ';'])
        eval(['tDouble=t' num2str(k) ';'])
        for i=1:sizeTrain
            rnd=ceil( size(pDouble,2)*rand() );
            p_train(:,sizeTrain*(k-1)+i)=pDouble(:,rnd);
            t_train(:,sizeTrain*(k-1)+i)=tDouble(:,rnd);
            pDouble(:,rnd)=[];
            tDouble(:,rnd)=[];
        end
        p_test=[p_test pDouble];
        t_test=[t_test tDouble];
        pDouble=[];
        tDouble=[];
    end

    % 3. B. Train
        % Initialize Parameters
        W1=rand(middleSize,size(p_train,1))/10;
        b1=rand(middleSize,1)/10;
```

```
        W2=rand(size(t_train,1),middleSize)/10;
        b2=rand(size(t_train,1),1)/10;
        % Train
        nntwarn off
        [W1,b1,W2,b2,te,tr] =
MYtbpx2(W1,b1,'tansig',W2,b2,'logsig',p_train,t_train,[epochs_between_display,epochs_num,erro
r_goal,learning_rate],pSizes,middleLayer);

    % 3. C. Train test
%     train_test=feval('logsig',W2*feval('tansig',W1*p_train,b1),b2);
%     [y,i]=max(train_test);
%     [Y,I]=max(t_train);
%     result=size(t_train,2);
%     for k=1:size(t_train,2)
%         if i(k)~=I(k)
%             result=result-1;
%         end
%     end
%     result=100*result/size(t_train,2);
%     results=[results result];

    % 3. D. Real test
    test=feval('logsig',W2*feval('tansig',W1*p_test,b1),b2);
    [y,i]=max(test);
    [Y,I]=max(t_test);
    result=size(t_test,2);
    for k=1:size(t_test,2)
        if i(k)~=I(k)
            result=result-1;
        end
    end
    result=100*result/size(t_test,2)
    results=[results result];

end


% 4. Summarize & show the results
Score=mean(results)
hist(results)
```

MYtbpx2

הפונקציה MYtbpx2 משמשת לאימון רשת נוירונים. היא מבוססת על הפונקציה tbpx2 המובנית ב-MATLAB. להלן קטעי הקוד שנוספו לפונקציה זו לצורך הפרוייקט:

שורות 12-16 (יצירת המטריצה המאפסת):

```
mask=ones(round(sum(pSizes)*middleLayer),sum(pSizes));
for request=1:max(size(pSizes))
   mask( round(sum(pSizes(1:request-
1))*middleLayer)+1:round(sum(pSizes(1:request))*middleLayer) , 1:sum(pSizes(1:request-1))
)=0;
```

```
   mask( round(sum(pSizes(1:request-
1))*middleLayer)+1:round(sum(pSizes(1:request))*middleLayer) ,
sum(pSizes(1:request))+1:sum(pSizes) )=0;
end
```

שורה 116 (שימוש במטריצה המאפסת):

```
  new_w1=new_w1.*mask;
```

קוד התוכנית

הסקריפט

```
#!/bin/sh

if [ "${1}" == "" ]; then
  echo "Usage: ${0} <ip>"
  exit
fi

Headers[0]="Accept-Ranges" Headers[1]="Age" Headers[2]="Allow"
Headers[3]="Cache-Control" Headers[4]="Content-Encoding"
Headers[5]="Content-Language" Headers[6]="Content-Length"
Headers[7]="Content-Location" Headers[8]="Content-Disposition"
Headers[9]="Content-MD5" Headers[10]="Content-Range"
Headers[11]="Content-Type" Headers[12]="Date" Headers[13]="ETag"
Headers[14]="Last-Modified" Headers[15]="Location" Headers[16]="Set-
Cookie"
ECodes1[0]="500" ECodes1[1]="414" ECodes1[2]="413" ECodes1[3]="406"
ECodes1[4]="404" ECodes1[5]="400" ECodes1[6]="202" ECodes1[7]="200"
ECodes2[0]="500" ECodes2[1]="406" ECodes2[2]="404" ECodes2[3]="403"
ECodes2[4]="400" ECodes2[5]="307" ECodes2[6]="302" ECodes2[7]="301"
ECodes3[0]="407" ECodes3[1]="406" ECodes3[2]="405" ECodes3[3]="403"
ECodes3[4]="401" ECodes3[5]="400" ECodes3[6]="202" ECodes3[7]="500"
ECodes3[8]="503" ECodes3[9]="417" ECodes3[10]="412"
ECodes4[0]="409" ECodes4[1]="406" ECodes4[2]="400" ECodes4[3]="202"
ECodes4[4]="200" ECodes4[5]="501" ECodes4[6]="500" ECodes4[7]="417"
ECodes4[8]="412"
Methods[0]="HEAD" Methods[1]="GET" Methods[2]="POST" Methods[3]="PUT"
Methods[4]="DELETE" Methods[5]="TRACE" Methods[6]="OPTIONS"
Methods[7]="CONNECT"
RequestDef="\r\nHost: host\r\nConnection: Close\r\nHeader1:
header1\r\nHeader2: header2\r\n\r\n"
Servers[0]="Apache" Servers[1]="IIS"

function writeX
{
  I="${1}"
  while [ ${I} -gt 0 ]; do
    echo -ne "X";
    let I--
    done
}

function send
{
  exec 7<>/dev/tcp/"${1}"/80
  echo -ne "${REQ}${RequestDef}" >&7
  cat <&7
}

function sendreq
{
#  echo -ne "${BANNER}"
  RESPONSE="$(send "${1}")"
#  echo " done"
```

```
}

function processVectors
{
  I=0
  while [ "${Headers[${I}]}" != "" ]; do
  H="$(echo -ne "${RESPONSE}" | grep -n "${Headers[${I}]}:")"
  if [ "${H}" == "" ]; then
    V1[${I}]="0";
    V2[${I}]="0";
    V3[${I}]="0";
  else
    T=$(echo -ne "${H}" | cut -d: -f1)
    let T=T-1
    V1[${I}]="${T}" # position vector
    T=$(echo -ne "${H}" | cut -d: -f3 | wc -m)
    V2[${I}]="${T}" # byte count vector
    T=$(echo -ne "${H}" | sed 's/[^A-Z]//g' | wc -m)
    V3[${I}]="${T}" # capital count vector
  fi
  let I++
  done
}
function printVectors
{
  I=0
  while [ "${V1[${I}]}" != "" ]; do
    echo -ne "${V1[${I}]}\t"
    let I++
  done
  I=0
  while [ "${V2[${I}]}" != "" ]; do
    echo -ne "${V2[${I}]}\t"
    let I++
  done
  I=0
  while [ "${V3[${I}]}" != "" ]; do
    echo -ne "${V3[${I}]}\t"
    let I++
  done

}
function processMethods
{
  H=$(echo -ne "${RESPONSE}" | grep "Allow:" | cut -d: -f2)
  I=1
  ALLOW=""
  T=$(echo -ne ${H} | cut -d, -f1)
  while [ "${T}" != "" ]; do
  ALLOW="${ALLOW}${T}\n"
  let I++
  T=$(echo -ne $(echo -ne ${H} | cut -d"," -f${I}))
  done

  I=0
  while [ "${Methods[${I}]}" != "" ]; do
  H="$(echo -ne "${ALLOW}" | grep -n "${Methods[${I}]}")"
```

```
  if [ "${H}" == "" ]; then
    M[${I}]="0";
  else
    T=$(echo -ne "${H}" | cut -d: -f1)
    M[${I}]="${T}" # position vector
  fi
  let I++
  done
}


function processError1
{
  I=0
  ERes=$(echo -ne "${RESPONSE}" | head -1)
  ECode=$(echo -ne "${ERes}" | cut -d" " -f2)
  FLAG=0
  while [  "${ECodes1[${I}]}" != "" ]; do
    if [ "${ECodes1[${I}]}" == "${ECode}" ]; then
      FLAG=1
      E[${I}]="1"
    else
      E[${I}]="0"
    fi
    let I++
  done

  if [ "${FLAG}" == "0" ]; then
    E[${I}]="1"
  else
    E[${I}]="0"
  fi

  let I++

  EMsg=$(echo -ne "${ERes:12}")
  E[${I}]="${#EMsg}"

  let I++
  E[${I}]=""

}

function processError2
{
  I=0
  ERes=$(echo -ne "${RESPONSE}" | head -1)
  ECode=$(echo -ne "${ERes}" | cut -d" " -f2)
  FLAG=0
  while [  "${ECodes2[${I}]}" != "" ]; do
    if [ "${ECodes2[${I}]}" == "${ECode}" ]; then
      FLAG=1
      E[${I}]="1"
    else
      E[${I}]="0"
    fi
    let I++
```

```
  done

  if [ "${FLAG}" == "0" ]; then
    E[${I}]="1"
  else
    E[${I}]="0"
  fi

  let I++

  EMsg=$(echo -ne "${ERes:12}")
  E[${I}]="${#EMsg}"

  let I++
  E[${I}]=""
}
function processError3
{
  I=0
  ERes=$(echo -ne "${RESPONSE}" | head -1)
  ECode=$(echo -ne "${ERes}" | cut -d" " -f2)
  FLAG=0
  while [  "${ECodes3[${I}]}" != "" ]; do
    if [ "${ECodes3[${I}]}" == "${ECode}" ]; then
      FLAG=1
      E[${I}]="1"
    else
      E[${I}]="0"
    fi
    let I++
  done

  if [ "${FLAG}" == "0" ]; then
    E[${I}]="1"
  else
    E[${I}]="0"
  fi

  let I++

  EMsg=$(echo -ne "${ERes:12}")
  E[${I}]="${#EMsg}"

  let I++
  E[${I}]=""

}
function processError4
{
  I=0
  ERes=$(echo -ne "${RESPONSE}" | head -1)
  ECode=$(echo -ne "${ERes}" | cut -d" " -f2)
  FLAG=0
  while [  "${ECodes4[${I}]}" != "" ]; do
    if [ "${ECodes4[${I}]}" == "${ECode}" ]; then
      FLAG=1
      E[${I}]="1"
```

```
    else
      E[${I}]="0"
    fi
    let I++
  done

  if [ "${FLAG}" == "0" ]; then
    E[${I}]="1"
  else
    E[${I}]="0"
  fi

  let I++

  EMsg=$(echo -ne "${ERes:12}")
  E[${I}]="${#EMsg}"

  let I++
  E[${I}]=""

}
function processType
{
  H=$(echo -ne "${RESPONSE}" | grep "Server:" | cut -d: -f2)
  I=0
  FLAG=""
  while [ "${Servers[${I}]}" != "" ]; do
    T=$(echo -ne "${H}" | grep  "${Servers[${I}]}")
    if [ "${T}" != "" ]; then
      S[${I}]="1"
      FLAG="0"
    else
      S[${I}]="0"
    fi
    let I++
  done

  if [ "${FLAG}" == "" ]; then
    S[${I}]="1"
  else
    S[${I}]="0"
  fi
}
function printType
{
    I=0

    while [ "${S[${I}]}" != "" ]; do
    echo -ne "${S[${I}]}\t"
    let I++
    done
}


function printError
{
  I=0
```

```
  while [ "${E[${I}]}" != "" ]; do
    echo -ne "${E[${I}]}\t"
    let I++
  done
}
function printMethods
{
  I=0

  while [ "${M[${I}]}" != "" ]; do
    echo -ne "${M[${I}]}\t"
    let I++
  done
}



# req 1
BANNER="legitimate GET"
REQ="GET / HTTP/1.1"
sendreq "${1}"
processVectors
printVectors

# req 2
BANNER="GET with long URL (10000?)"
REQ="GET /$(writeX 10000) HTTP/1.1"
sendreq "${1}"
processVectors
printVectors
processError1
printError


# req 3
BANNER="GET non existing URL"
REQ="GET /nonexistingurl HTTP/1.1"
sendreq "${1}"
processVectors
printVectors
processError2
printError

#req 4
BANNER="legitimate HEAD"
REQ="HEAD / HTTP/1.1"
sendreq "${1}"
processVectors
printVectors


#req 5
BANNER="OPTIONS"
REQ="OPTIONS / HTTP/1.1"
sendreq "${1}"
processVectors
```

```
printVectors
processMethods
printMethods

#req 6
BANNER="DELETE"
REQ="DELETE / HTTP/1.1"
sendreq "${1}"
processVectors
printVectors
processError3
printError
processMethods
printMethods

#req 7
BANNER="undefined method"
REQ="BLAHBLAH / HTTP/1.1"
sendreq "${1}"
processVectors
printVectors
processError3
printError
processMethods
printMethods



#req 8
BANNER="unknown protocol ver"
REQ="GET / HTTP/4.9"
sendreq "${1}"
processVectors
printVectors
processError4
printError


#server type
BANNER="server header"
REQ="GET / HTTP/1.1"
sendreq "${1}"
processType
printType

echo
```

התוכנה ב-MATLAB

```
function [] = Server(input)
nntwarn OFF
load NET

input=input(1,1:489)';
input(erased_rows)=[];
```

```
result=feval('logsig',W2*feval('tansig',W1*input,b1),b2);
[Certainty,index]=max(result);

if index==1
    server='Apache';
end
if index==2
    server='Microsoft IIS';
end
if index==3
    server='Neither Microsoft nor Apache';
end

server
Certainty
```