# הקדמה ללמידת מכונה

## הקדמה

למידת מכונה היא תת-תחום במדעי המחשב ובבינה מלאכותית, העוסק בפיתוח אלגוריתמים שבאמצעותם ניתן ליצור תוכנות מחשב הלומדות מתוך דוגמאות כיצד להסיק תכונות לא ידועות על מידע חדש. למידת מכונה פועלת במגוון משימות חישוביות בהן התכנות הקלאסי אינו אפשרי. המקרה הנפוץ ביותר בו נרצה להשתמש בלמידת מכונה הוא בעיית זיהוי שמומחה אנושי מסוגל לפתור, אך לא מסוגל לכתוב את הכללים לזיהוי בצורה מפורשת או שהם משתנים עם הזמן ולא ניתנים לכתיבה מראש. העקרון המרכזי בלמידת מכונה הוא לתת למחשב קבוצה גדולה של מידע מתויג וכלים שבאמצעותם המחשב יוכל ללמוד את המידע ולפתח מודל שידע להכריע בהסתברות גבוהה על כל מידע חדש.

## סוגי בעיות

**Classification** - נתונים k מחלקות. אנו מעוניינים לסווג כל אובייקט לאיזה מחלקה מ-k המחלקות הוא שייך. פעמים רבות מספר המחלקות הוא 2 המייצגים "כן" או "לא", אך גם יכולים להיות מספר רב של מחלקות. דוגמאות:

* בהינתן נתונים על אדם נרצה לדעת האם הוא עובד או מובטל.
* בהינתן תמונה של פרי נרצה לדעת מהו סוג הפרי.

**Regression** - בהינתן אובייקט x נרצה לנבא עבורו איזשהו ערך מספרי רצוף y. דוגמאות:

* בהינתן נתונים על אדם כלשהו נרצה לנבא מהו משקלו או גובהו.
* בהינתן גרסה של פלאפון נרצה לנבא מהו מחיר הפלאפון.

**Clustering** - בהינתן n אובייקטים לקבץ אותם למספר קבוצות, ושיוך דוגמה חדשה לקבוצה המתאימה ביותר. כל האובייקטים הנמצאים באותה קבוצה דומים זה לזה יותר מאשר לאובייקטים השייכים לקבוצות אחרות. לדוגמה, פילוח של לקוחות לפי התנהגות צרכנית ותכונות דמוגרפיות.

**Ranking** - בהינתן n אובייקטים לדרג אותם מהגבוה לנמוך לפי העדפות משתמש. לדוגמה, לדרג אתרי האינטרנט שנמצאו בחיפוש לפי העדפות המחפש כך שהטובים ביותר יופיעו קודם.

## מודל למידת מכונה

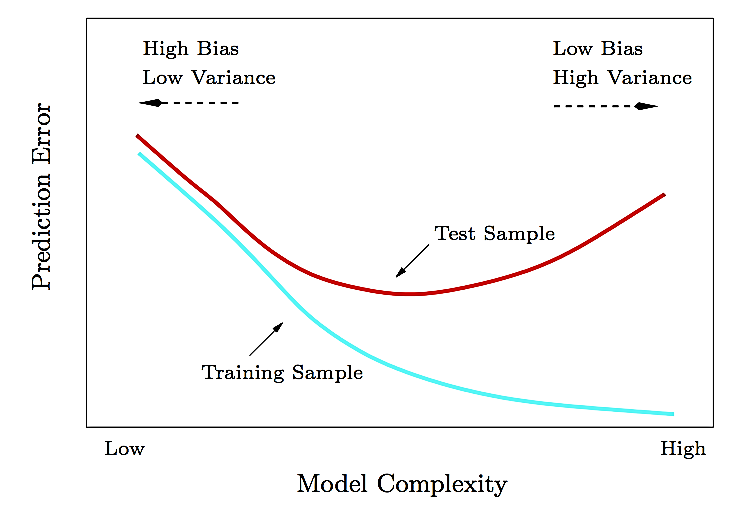
הרכיבים המרכזיים בפתירת בעיה באמצעות למידת מכונה הם:

1. **נתונים** - קבוצת X של N אובייקטים, כך שלכל אובייקט יש תכונות מסוימות (features) ותיוג y (label). מידע זה יחולק ל-train ו-test.
2. **פונקציית מודל** - פונקציה שאם נכניס אליה אובייקט עם כל התכונות תחזה מהו הערך שלו. כל פונקציה כזו מורכבת מפרמטרים . כפי שנראה מודל זה עשוי להיות מורכב מאוד.
3. **פונקציית Loss** - פונקציה שבהינתן labels וחיזוי לכל אובייקט בדאטה, מחזירה מספר המציין כמה אי-דיוק/טעות ממוצעת יש בהשוואה ביניהם. ככל שהערך שמחזירה קטן יותר כך המודל טוב יותר. פונקציה נפוצה כזו היא סכום השגיאות של כל האובייקטים ב-train בריבוע (MSE - Mean Squared Error).
4. **אלגוריתם** **למידה** - ישמש למציאת פונקציית מודל שעבורה פונקציית ה- Lossמחזירה ערך מינימלי.

עבור כל בעיה שאנו מנסים לפתור בלמידת מכונה, אנו מניחים שקיימת פונקציה , כך שעבור כל אובייקט x עם תיוג y, מתקיים: . הערך מייצג את ה-irreducible error, שהיא טעות שלא ניתן להימנע ממנה. מטרתנו היא למצוא קירוב ככל הניתן לפונקציה זו.

## Train, Test, Validation

בכל אחד מסוגי הבעיות לעיל (סעיף ב') כדי שנוכל גם לבדוק את המודל שאנו מפתחים, לא נשתמש בכל האובייקטים שברשותנו לפיתוח המודל, אלא נשאיר מספר אובייקטים (שאנו יודעים לאיזה מחלקה הם שייכים / מה הערך שלהם) בצד כדי שנוכל להעריך כמה המודל שפיתחנו הוא איכותי. נשווה בין ערכי האמת למה שהמודל ניבא. חלוקה זו של המידע הקיים הוא חלוקה לקבוצות train ו-test.

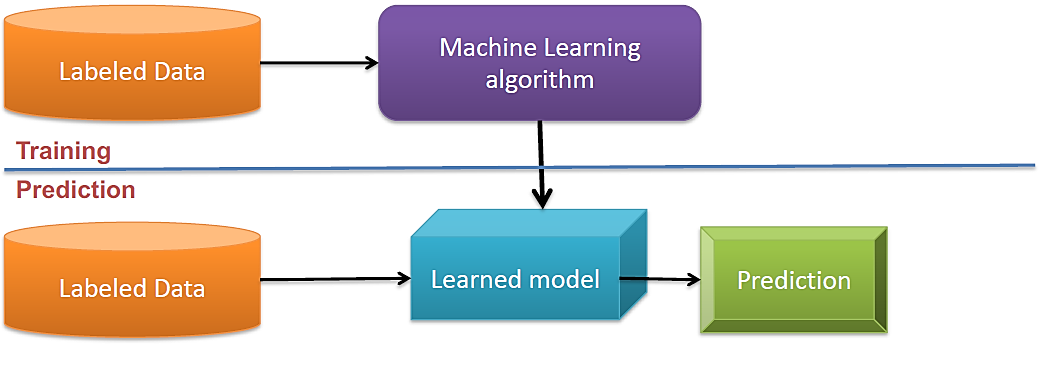
לא נרצה לבדוק את איכות המודל על המידע שעליו המודל התאמן, שכן בעיה מאוד נפוצה שיכולה לקרות היא Over-Fitting, שבה המודל יהיה מאוד מדויק על המידע שהתאמן עליו, אך על מידע חדש יציג תוצאות מאוד גרועות (ראה תמונה). לכן נרצה לחלק את המידע עוד לפני פיתוח המודל ל-train ו-test כך שעל ה-train נתאמן ועל ה-test נבדוק את איכות המודל. אמנם גם בחלוקה זו יכולה להיות בעיה, שבכל ניסוי נבדוק את הטעות ב-test ונשפר, כך שנקבל את המודל עבורו הטעות ב-test מינימלית. אמנם אז ה-test יאבד את המשמעות שלו, שכן יכול להיות שאנו מתאימים את המודל רק עבור ה-test ולא עבור מקרה כללי. כל המטרה ב-test היא שבודקים את הטעות בו פעם אחת בסוף הפיתוח של המודל כדי להעריך כמה המודל טוב ולא לפתח את המודל באמצעותו.

הפתרון לכך הוא לחלק את כל האובייקטים שיש בידינו לשלושה חלקים:

1. **train** - באמצעותו לומדים ומפתחים את המשוואה הליניארית על ידי הפעלת שיטת Gradient Descent שתוארה לעיל.
2. **validation** - מטרתו היא לעזור לנו לקבוע את מספר התכונות המלאכותיות שנכניס למודל. נשחק עם מספר התכונות וסוגן ואז נבדוק את הטעות בקבוצת ה-validation. נבחר את התכונות עבורן הטעות בקבוצת ה-validation מינימלית.
3. **test** - שעליו מריצים את המודל הסופי פעם אחת. מטרתו היא להעריך את טיב המודל.

אם אין הרבה אובייקטים החלוקה המקובלת היא 60% ל-train, 20% ל-validation, ו-20% ל-test. אמנם כאשר יש הרבה אובייקטים אפשר להפחית את כמות האובייקטים ב-test וב-validation. לדוגמה, אם יש מיליון אובייקטים אין צורך שה-test יהיה 200,000 אלא אפשר להסתפק גם ב-2000, וכן ניתן להסתפק ב-10,000 ב-validation. ב-train תמיד נרצה שיהיו כמה שיותר. ה-train וה-test צריכים להיות די זהים ברמת ההתפלגות. אם ה-train מייצג מידע מסוג אחד וה-test מידע מסוג אחר, אין לדעת האם המודל יעבוד גם ב-test שכן זהו מידע שחדש לו.

## שלבי בניית המודל

לפי האמור לעיל, כל בניית מודל בלמידת מכונה מחולק לשני שלבים: שלב הלמידה ושלב הבדיקה. בשלב הלמידה יש לנו צמדים , כאשר כל זהו אובייקט ה-i ב-train ו- זהו ה-label המתאים לו. המטרה היא למצוא את הפרמטרים האופטימליים בפונקציית המודל עבורם הערך בפונקציית ה-Loss יהיה מינימלי. בשלב הבדיקה, נקרא גם Generalization, מריצים את פונקציית המודל על כל האובייקטים ב-test שאותם המודל לא ראה ובודקים כמה הוא טעה על מידע זה. הטעות נמדדת על ידי ערך מספרי שנרצה שיהיה כמה שיותר קרוב ל-0.

## בעיות בפיתוח מודל

ראשית נציין שני מדדים חשובים בבדיקת איכות מודל.

**Bias** (הטיה) - הוא ההבדל בין החיזוי הממוצע של המודל לבין הערך הנכון אותו אנו מנסים לחזות. כמו ההגדרה של loss. מודל עם bias גבוה הוא מודל פשוט מדי או שמניח הנחות לא מדויקות על המידע. Bias גבוה תמיד מוביל לשגיאות גבוהות בנתוני ה-train וה-test.

**Variance** (שונות) - היא שגיאה הנובעת מרגישות לשינויים קטנים ב-train set. שונות גבוהה יכולה לגרום לאלגוריתם למדל את הרעש האקראי בנתוני האימון, ולא את הפלט הרצוי. מודל עם שונות גבוהה מקדיש תשומת לב רבה לנתוני ה-train ואינו מכליל נתונים אחרים שלא ראה קודם. כתוצאה מכך, מודלים כאלה מתפקדים טוב מאוד בנתוני ה-train אך בעלי שיעורי שגיאה גבוהים בנתוני ה-test.

### Underfitting

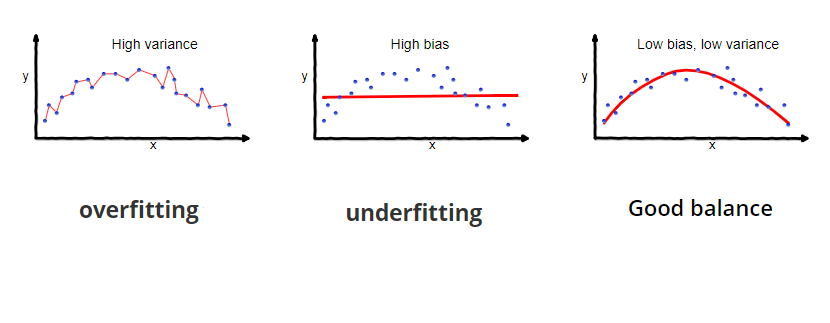
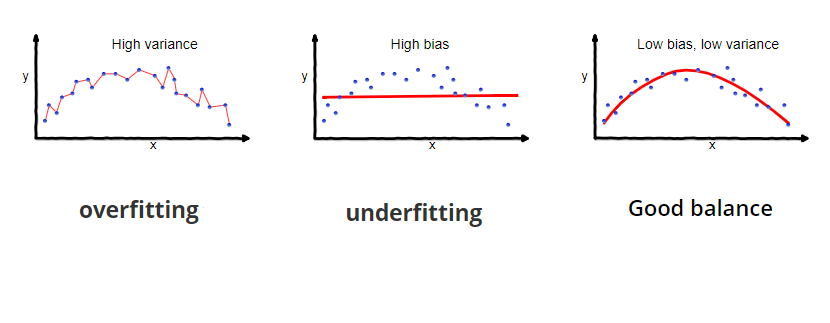
בעיה המתרחשת כאשר מודל אינו מסוגל ללמוד את הדפוס הבסיסי של הנתונים. למודלים אלה יש בדרך כלל bias גבוה ו-variance נמוכה, המתבטאים בטעות גבוהה גם ב-train וגם ב-test. זה קורה כשיש לנו כמות קטנה מאוד של נתונים לבניית מודל מדויק או כשאנחנו מנסים לבנות מודל לינארי עם נתונים לא לינאריים. כמו כן, מודלים עם בעיה זו בדרך כלל פשוטים מאוד לתפיסת הדפוסים המורכבים בנתונים כמו רגרסיה לינארית ולוגיסטית.

הפתרונות לבעיה זו הם בדרך כלל הוספת מידע או להשתמש באלגוריתם למידה מורכב יותר שידע לזהות את הדפוסים במידע. מודל מורכב יותר גם יכול להיות פשוט לאמן יותר צעדים.

### Overfitting

בעיה המתרחשת כאשר המודל שלנו לוכד את הרעש ב-data יחד עם הדפוס הבסיסי. זה קורה כשאנחנו מכשירים את המודל שלנו הרבה על מערך נתונים רועש. למודלים אלו bias נמוך ושונות גבוהה, המתבטאים בטעות קטנה ב-train וטעות גדולה ב-test, כלומר המודל מותאם מדי ל-train אך לא למידע אחר. מודלים אלה מורכבים מאוד ונוטים להתאמת יתר.

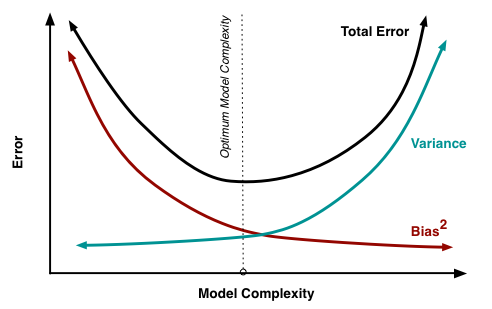
הפתרונות לבעיה זו הם בדרך כלל לפשט את המודל, להוריד תכונות או להשתמש במידע פחות רועש, כלומר שיש בו דפוס בולט. אפשרות נוספת, עליה נדון בהמשך, היא להפעיל רגולריזציה על המשקלים במודל.



### Bias-Variance tradeoff

כדי לקבל מודל איכותי עלינו למזער הן את ה-bias והן את ה-variance. אמנם זה לא טריוויאלי לעשות זאת שכן הם אחד על חשבון השני. מורכבות המודל היא הסיבה שיש tradeoff בין bias ל-variance. אלגוריתם לא יכול להיות מורכב יותר ופחות מורכב בו זמנית. אם המודל שלנו פשוט מדי, כלומר יש לו מעט מאוד פרמטרים, ייתכן שיש לו bias גבוה ו-variance נמוך. מצד שני אם המודל יהיה יותר מורכב, כלומר יש מספר גדול של פרמטרים, אז יהיה לו variance גבוה ו-bias נמוך.

לכן כדי לבנות מודל טוב, עלינו למצוא את האיזון הנכון בין bias ל-variance, מבלי להגיע ל-Overfitting או Underfitting, וגם למזער את השגיאה הכוללת. איזון אופטימלי של bias ו-variance לעולם לא יגיע ל-Overfitting או Underfitting. חישוב האיזון הנכון ביניהם נעשה באמצעות הערך המינימלי של הנוסחה למטה, כאשר irreducible error מייצגת את הטעות שלא ניתן להימנע ממנה ונובעת מרעש של על המידע.



## סוגי למידה

נהוג לחלק את אלגוריתמי למידת המכונה למספר סוגים:

* למידה מונחית (supervised learning) - כל דוגמה מגיעה עם תיוג. מטרת האלגוריתם היא ללמוד מהמידע המתויג כיצד לחזות את התיוג של דוגמאות חדשות שאותן לא פגש בתהליך הלמידה. לדוגמה סיווג מיילים ל"חשוב" ו"ספאם" כאשר נתונים מיילים שכבר מסווגים. החסרונות בלמידה כזו שקשה להחליט מהו המידע הטוב ביותר ללמוד עליו ומהו האלגוריתם הטוב ביותר ללמוד איתו.
* למידה בלתי מונחית (unsupervised learning). מטרת האלגוריתמים היא למצוא ייצוג פשוט וקל להבנה של אוסף הנתונים. שיטות נפוצות מסוג זה הן חלוקה לקבוצות (clustering), ושיטות להורדת מימד כגון ניתוח גורמים ראשיים (PCA). החיסרון בלמידה זו שבדרך כלל מביאה לתוצאות פחות טובות מלמידה מונחית.
* למידה מונחית חיזוקים (reinforcement learning) - אלגוריתם הלמידה מקבל משוב חלקי על ביצועיו (רק לאחר סיום ביצוע המטלה) ועליו להסיק אילו מהחלטותיו הביאו להצלחה/כישלון. בסוף תהליך הלמידה האלגוריתם יודע בכל מצב מהי הפעולה הטובה ביותר. למידה זו משמשת כיום בעיקר בפיתוח תוכנה שתדע לשחק משחקים.

## סוגי מודלים

* מודל יוצר (generative model) - זהו מודל שבו מחשבים עבור כל מחלקה y מה ההסתברות שאובייקט x שייך למחלקה זו , וכן מה ההסתברות שאובייקט כלשהו שייך למחלקה y . דוגמה למודלים מסוג זה הם Naïve Bayes ו-Bayesian network.
* מודל מפלה (discriminative model) - הוא מודל שבו מחשבים רק מה ההסתברות שאובייקט x שייך למחלקה y . *דוגמה למודלים מסוג זה הם* Linear Regression, SVM, Decision Trees. *בדרך כלל מודל מפלה הוא יעיל יותר מאשר מודל יוצר.*

# עקרונות למידת מכונה

## PAC Learning

PAC learning (Probably approximately correct learning, ובתרגום חופשי: "למידת קרוב לוודאי בערך נכון") הוא מודל לניתוח מתמטי של בעיות מתחום הלמידה החישובית. מודל זה קובע מתי בעיה ניתנת לפתרון או לפתרון יעיל באמצעות למידת מכונה. המודל הוצג ב־1984 על ידי מדען המחשב הבריטי לסלי וליאנט.

במודל זה הלומד (אלגוריתם למידה) מקבל דוגמאות וצריך לבחור פונקציה מכלילה (הנקראת במודל זה "חוק") מתוך מחלקה מוגדרת של פונקציות אפשריות (מחלקת חוקים נקראת "היפותזה"). המטרה היא שבהסתברות גבוהה ("קרוב לוודאי"), הפונקציה הנבחרת תהיה בעלת שגיאת הכללה ("בערך נכון") נמוכה. אלגוריתם הלמידה נדרש להיות מסוגל ללמוד פונקציה בהינתן בחירה של יחס קירוב נדרש, מידת וודאות או התפלגות של דוגמאות. המודל הורחב בהמשך כדי לטפל גם ברעש על המידע, או דוגמאות המסווגות בצורה שגויה.

### הגדרה פורמלית

בהינתן בעיה P, נגדיר את X להיות מרחב כל הקלטים של הבעיה ו-Y מרחב כל הפלטים של הבעיה. עוד נסמן ב-H את קבוצת כל הפונקציות הממפות , כלומר מקבלות מופע של הבעיה ומחזירות פלט , לא בהכרח ש-y הוא הפלט הנכון עבור x. אם קיים אלגוריתם למידה A המקבל כקלט שני פרמטרים ומחזיר פונקציה , כך שבהסברות אזי טועה לכל היותר ב-על X, נאמר כי הבעיה P היא PAC Learnable. אם בנוסף קיים אלגוריתם A שעושה זאת בזמן פולינומי, אזי P היא Efficiently PAC Learnable.

### משפטים מרכזיים

יהי X מרחב קלטים של בעיה, D התפלגות על X, היפותזה H שהיא קבוצה **סופית** של פונקציות (חוקים) הממפות , ומדגם אקראי של n קלטים לפי D.

1. Generalization bound ללא טעות במדגם - אם קיימת פונקציה שהיא עקבית על S, כלומר לא טועה על אף קלט מ-S, אזי קיים כך שבהסברות לכל הפחות הטעות האמיתית של h על X כולו, שנסמן אותה ב-e(h), היא לכל היותר:

כלומר, אם יש קבוצה סופית של חוקים ומתוכם יש חוק שהוא טוב על המדגם, אזי בהסתברות מאוד גבוהה חוק זה הוא גם טוב על כל העולם.

1. Generalization bound כשיש טעות במדגם - נניח כי עבור פונקציה הטעות של h על S היא . אזי קיים כך שבהסברות לכל הפחות הטעות האמיתית של h על X, שנסמן אותה ב-e(h), היא לכל היותר:

כלומר אם יש קבוצה סופית של חוקים ומתוכם יש חוק שהטעות שלו על המדגם היא קטנה, אזי בהסתברות מאוד גבוהה הטעות שלו גם על העולם כולו היא קטנה.

ניתן לראות שיש חשיבות גדולה למספר החוקים בהיפותזה H. בסעיף הבא נלמד כיצד סופרים כמה חוקים בכל היפותזה.

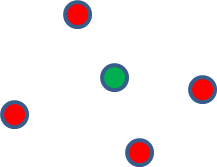
## VC Dimension

VC Dimension (Vapnik-Chervonenkis dimension) הוא מדד בתחום למידת מכונה המתאר את רמת כושר ההפרדה (הקיבולת, הסיבוכיות או העושר) של קבוצת חוקים שניתנת ללמידה באמצעות אלגוריתם למידה. מדד זה משמש, כמו שנראה במשפטים בהמשך, כדי לדעת את מספר החוקים בהיפותזה כלשהי, כדי שנוכל להציב ערך זה בנוסחאות שלמדנו בסעיף קודם. לפני שנסביר מהו מימד VC יש להסביר מהו "ניפוץ".

**ניפוץ (Shattering)** - בהינתן היפותזה H (קבוצה של חוקים/מסווגים) וקבוצה S המכילה n נקודות, נאמר כי H מנפצת את S אם יש חוק ב-H לכל סיווג אפשרי של S. כלומר, ב-H צריכים להיות חוקים שכל אחד מספק סיווג שונה על S.

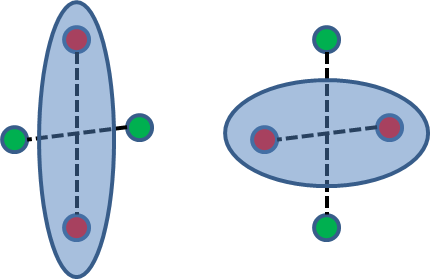
**מימד VC** - בהינתן היפותזה H, המימד VC של H הוא המספר המקסימלי של נקודות ש-H יכולה לנפץ עבור איזשהו סידור שלהם. במילים אחרות, אם H יכולה לנפץ איזשהו סידור של d נקודות אך לא יכולה לנפץ d+1 נקודות, אזי המימד VC של H הוא d. באופן דומה, כדי להוכיח שמימד VC של היפותזה H הוא d, יש להראות ש-H מסוגלת לנפץ **איזשהו סידור** של d נקודות (חסם תחתון) אך לא יכולה לנפץ d+1 נקודות (חסם עליון).

### דוגמאות

1. **מרווח חד-כיווני**: נתונים שורה של נקודות, כל חוק בהיפותזה זו הוא מיקום על שורה זו שבה כל הנקודות מצד ימין מסווגת 0 ומצד שמאל 1. המימד VC של מרווח חד-כיווני הוא 1.
2. **מרווח דו-כיווני**: כמו מרווח ח-כיווני, אלא שלכל מיקום אפשר להחליט איזה צד מסווג 0 ואיזה 1. מימד ה-VC של מרווח דו-כיווני הוא 2.
3. **קו דו-כיווני**: כל חוק בהיפותזה זו הוא קו שבו ניתן להחליט איזה צד מסווג 0 ואיזה 1. מימד ה-VC של מרווח דו-כיווני הוא 3.
4. **מלבן חד כיווני**: כל חוק הוא מלבן (בשני ממדים) שכל הנקודות בתוכו מסווגות 0, וכל הנקודות מחוץ מסווגות 1. מימד ה-VC של מלבן חד-כיווני הוא 4.

הוכחה: מלבן יכול לנפץ 4 נקודות (טריוויאלי), אך אינו יכול לנפץ 5 נקודות, מפני שעבור כל סידור של 5 נקודות לא נוכל לסווג את 4 הנקודות הכי קיצוניות לכל כיוון כ-0 ואת הנקודה החמישית כ-1.

1. **מעגל חד-כיווני**: כל חוק הוא מעגל (בשני ממדים) שכל הנקודות בתוכו מסווגות 0, וכל הנקודות מחוץ מסווגות 1. מימד ה-VC של מלבן חד-כיווני הוא 3.

הוכחה: מעגל יכול לנפץ 3 נקודות (טריוויאלי), אך אינו יכול לנפץ 4 נקודות. אם הנקודות מסודרות כך שיש נקודה אחת שמוכלת בצורה הקמורה של שלושת האחרות, אזי מעגל לא יכול לסווג את שלושת הנקודות החיצוניות 0 ואת הפנימית 1. אם הנקודות מסודרות בצורה כללית, אזי אם יש מעגל שיכול להכיל שתי נקודות מנוגדות לא יכול להיות שיש מעגל שמכיל רק את שני הנקודות האחרות (ראה תמונה).

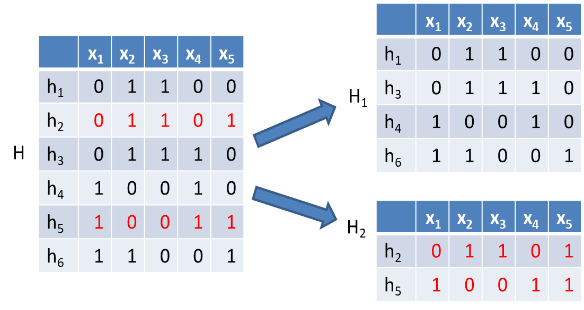
1. **כדור חד-כיווני**: כל חוק הוא כדור (בשלושה ממדים) שכל הנקודות בתוכו מסווגות 0, וכל הנקודות מחוץ מסווגות 1. מימד ה-VC של מלבן חד-כיווני הוא 4.

### משפט Sauer

בהינתן היפותזה H שהמימד VC שלה הוא d וקבוצה S עם n נקודות, נסמן את להיות כל הסיווגים השונים ש-H יכולה לתת ל-S. לפי משפט Sauer מתקיים:

כלומר נוכל לחשב חסם עליון למספר הסיווגים השונים ש-H יכול לתת ל-S. באמצעות חסם זה נוכל להציב ערך זה בנוסחאות Generalization Bound שלמדנו ב-PAC learning כדי לקבל את הטעות האמיתית של חוק כלשהו ב-H על S.

**מסקנה**: אם המימד VC קטן מ-n ויש חוק שהטעות שלו על המדגם קטנה, אזי בהסתברות גבוהה נקבל שהטעות של h על העולם כולו היא קטנה. ככל שהמימד VC קטן יותר כך הטעות על העולם קטנה יותר ונקבל חוק טוב יותר. במילים אחרות, למידה היא אפשרית אם החוק הוא פשוט, כלומר יש מעט חוקים.

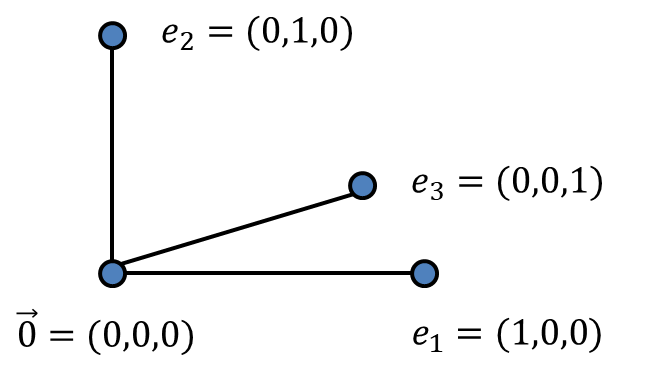


**הוכחת המשפט:** ניקח את כל החוקים ב-H שנותנים סיווג שונה על S, ונחלק אותם לשני קבוצות ו- בדרך הבאה: אם יש שני חוקים שמסווגים שונה **רק** את הנקודה האחרונה , אזי נעביר את החוק שסיווג את כ-1 ל-. כל השאר יהיו ב-. נשים לב שעבור הקבוצה מספר התיוגים השונים ב- ו- לא משתנה: , אולם הממד VC של הוא לכל היותר d-1: . כעת, נוכיח את משפט Sauer באינדוקציה על מספר הנקודות n ב-S.

* בסיס - עבור n=d יש לכל היותר סיווגים שונים, שאכן קטן יותר מ-.
* הנחה - נניח כי המשפט נכון עבור n-1.
* צעד - נוכיח עבור n. לפי הנחת האינדוקציה ומשפט פסקל מתקיים:

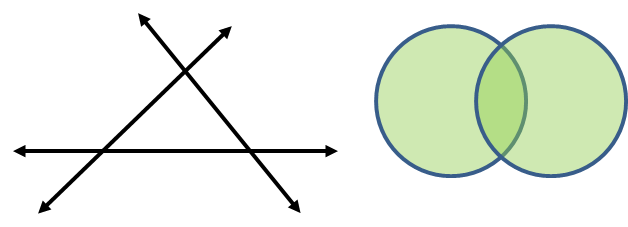
### משפטים חשובים

1. **מישור ב-d-1 ממדים**: כל חוק הוא מישור ב-d-1 ממדים בעולם של d ממדים . המישור מבדיל בין הנקודות המסווגות 0 לנקודות המסווגות 1. מימד ה-VC של מישור ב-d-1 ממדים הוא d+1.

הוכחה: מישור ב-d-1 ממדים יכול לנפץ d+1 נקודות. נסדר d נקודות במרחק של יחידה אחת מראשית הצירים, כך שכל נקודה נמצאת בציר אחר. נקודה נוספת נציב בראשית הצירים. עבור כל תת-קבוצה S של d+1 נקודות אלו יש מישור בעל d-1 מימדים שמכיל כל הנקודות בקבוצה זו ולא נקודות אחרות. עבור d+2 נקודות, לפי משפט רדון (Radon) כל קבוצה של d+2 נקודות ב- ניתנת לחילוק לשתי קבוצות זרות כך שהצורות הקמורות הנוצרות על ידן נחתכות. אין אף מישור שיכול לסווג את כל הנקודות מחוץ לחיתוך כ-0 ואת הנקודות בתוך החיתוך 1.

1. **מישור עם מרווח:** נסמן את רדיוס המעגל המקיף את כל הנקודות במדגם ב-R, ואת המרווח המינימלי בין המישור לנקודה כלשהי ב-. מספר החוקים המקסימלי ב-H על S הוא .

המסקנה העולה משני המשפטים שציינו עתה, היא שאם d קטן אזי מישור (חוק ליניארי) הוא מסווג מאוד טוב, אולם אם d מאוד גדול אזי נקבל באמצעות משפט Sauer חסם לא יעיל על . לכן במקרים כאלו מישור שהוא רחוק מן הנקודות במדגם הוא מסווג טוב. ככל שהמישור מתרחק מן הנקודות כך הטעות שלו על העולם תהיה קטנה יותר והמסווג יהיה טוב יותר.

1. **איחוד חוקים**: יהיו קבוצות חוקים עם אותו מימד VC , ויהיה איחוד כל קבוצות חוקים אלו, . אזי ניתן לתת חס עליון למימד VC של H:

דוגמה לאיחוד של קבוצות חוקים הוא איחוד של מספר קווים או מספר מעגלים.