רגולריזציה ובחירת מודל

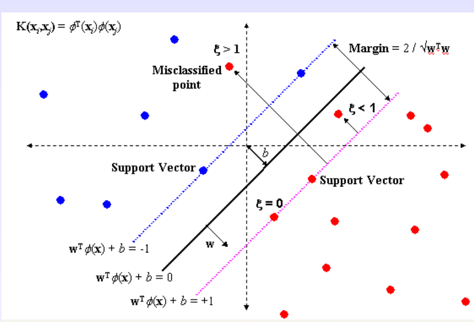
- נניח כי רוצים לבחור מודל מתוך מספר מודלים שונים עבור בעיית למידה.
- לדוגמא: מודל רגרסיה פולינומיאלית:

$$h_\theta(x) = g(\theta_0 + \theta_1 x + \theta_2 x^2 + \ldots + \theta_k x^k)$$

רוצים להחליט מהו k: ?k=0?, k=1?,…, k=10

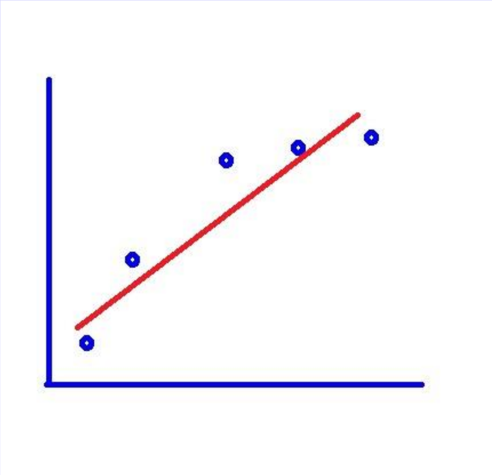
כיצד נבחר מודל שיהיה פשרה טובה בין ה- bias ל- variance?

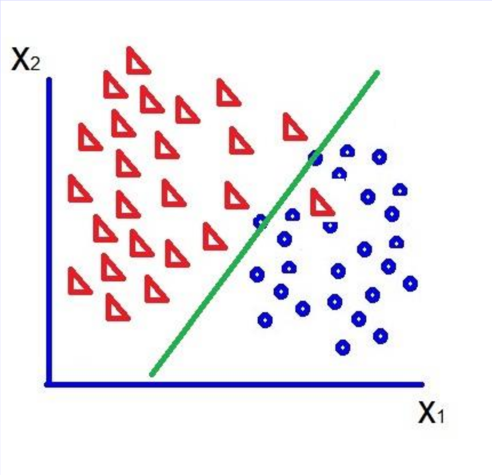
לחילופין, כיצד נבחר את הפרמטר C עבור SVM עם רגולריזציה?



בעיות בבחירת מודל

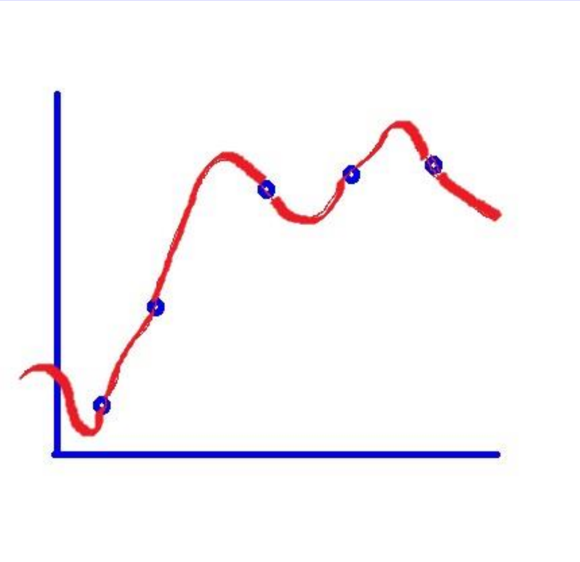
שגיאת ההטייה (Bias) – השגיאה הנגרמת מהנחות לא נכונות של אלגוריתם הלמידה (הנחות שגויות, כמו התאמה של ישר לינארי במקום פולינום מסדר שלישי, או על-מישור מפריד במקום משטח לא לינארי). שגיאת ה- Bias יכולה לגרום להחמצת הקשרים הרלבנטיים בין התכונות ופלטי המטרה - underfitting

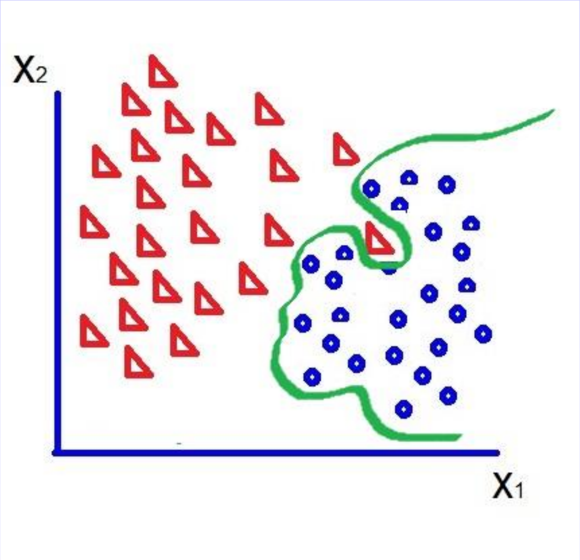




בעיות בבחירת מודל

שגיאת השונות (Variance) – השגיאה הנגרמת כתוצאה מרגישות לתנודות קטנות בנתוני האימון, כמו לדוגמא מידול של הרעש האקראי בנתוני האימון - overfitting





רגולריזציה ובחירת מודל

- נניח קבוצה סופית של מודלים: $M = \{M_1, M_2, \ldots, M_d\}$

ביניהם אנו רוצים לבחור.

לדוגמא עבור הדוגמא הראשונה של המודל הפולינומיאלי, נניח כי

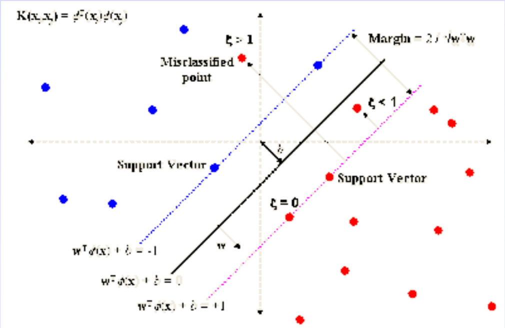
Mi הוא מודל הרגרסיה הלינארית מסדר i.

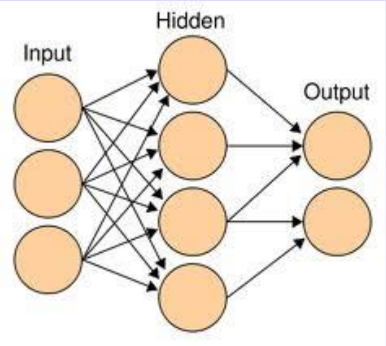
הערה: אפשר להרחיב ל- M אינסופי.

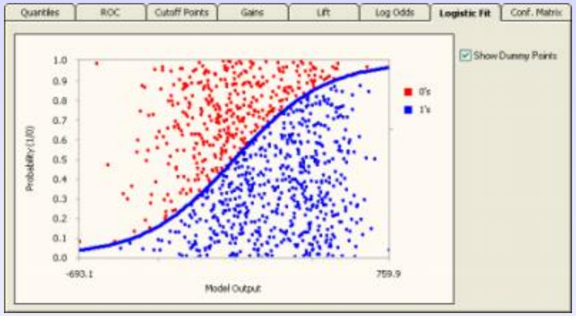
רגולריזציה ובחירת מודל

לחילופין, אם רוצים לבחור מודל מבין SVM, רשת עצבית או רגרסיה לוגיסטית, אזי M עשוי להכיל מודלים שונים.

$$M = \{M_1, M_2, \ldots, M_d\}$$







אימות מצולב (Cross-Validation)

- נניח כרגיל שנתונה קבוצת אימון S:

בהינתן מה שידוע לנו על מיזעור השגיאה או הסיכון האמפירי, נראה מה שבאופן התחלתי עשוי להראות כאלגוריתם לבחירת המודל:

1. אמנו כל אחד מהמודלים Mi על הקבוצה S כדי לקבל את ההיפותיזה h_i.
2. בחרו בהיפותיזה עם שגיאת האימון הקטנה ביותר.

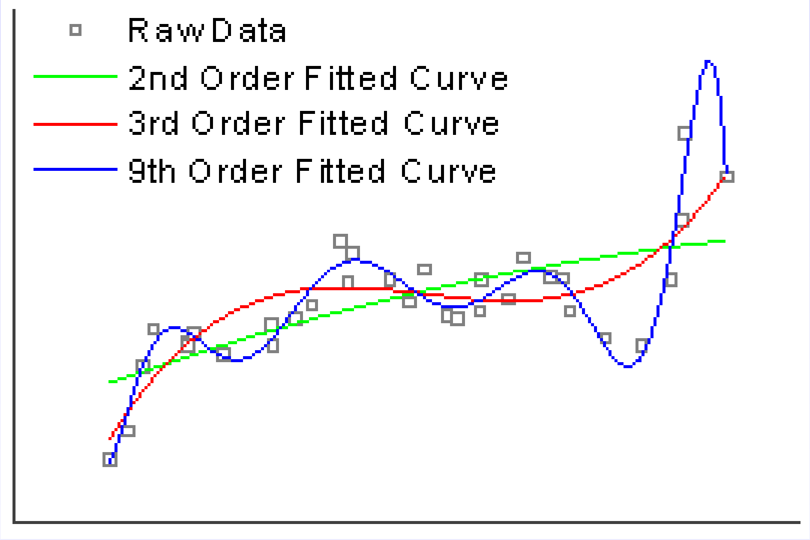
אימות מצולב (Cross-Validation)

האם האלגוריתם יעבוד?

אימות מצולב (Cross-Validation)

- התשובה היא כמובן <u>לא</u>.

לדוגמא, ברגע שבוחרים את סדר המודל ברגרסיה פולינומיאלית, ככל שסדר הפולינום גדול יותר, כך הוא יתאים טוב יותר לקבוצת האימון S והשגיאה האמפירית תהיה קטנה יותר.

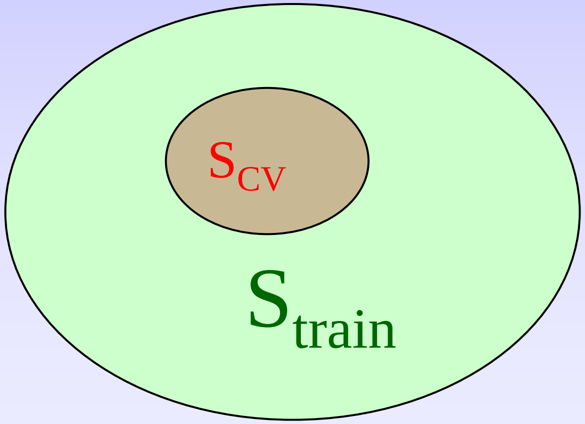


לכן אלגוריתם זה תמיד יבחר מודל מסדר גבוה, שכבר ראינו שהוא בחירה גרועה.

אימות מצולב (Cross-Validation)

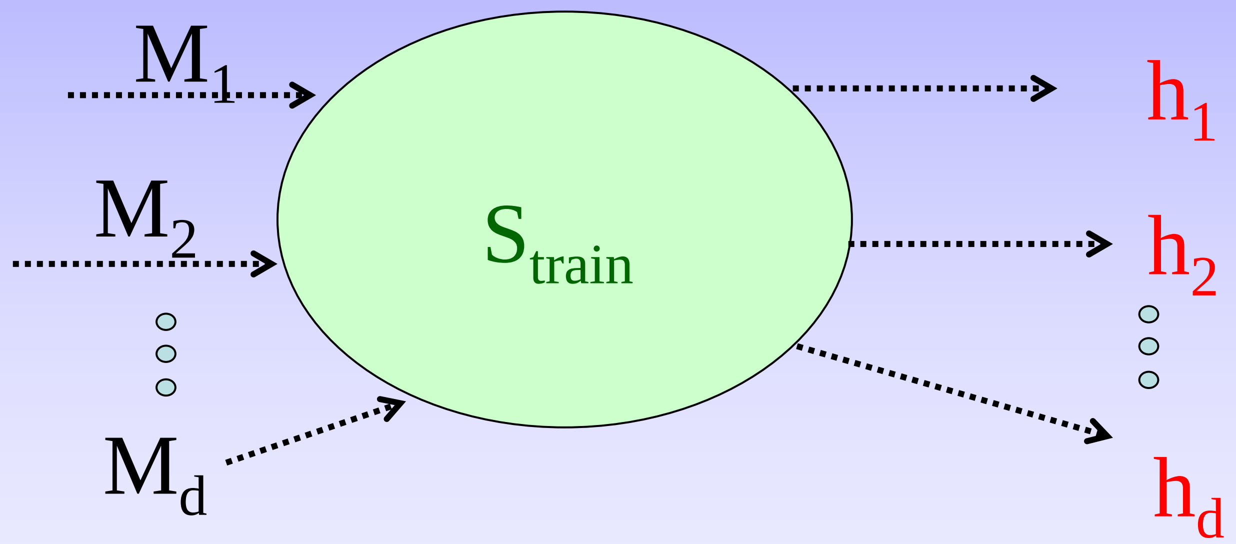
- נציע אלגוריתם שעובד טוב יותר:
- Hold-out cross-validation (simple cross validation)

1. מחלקים באופן אקראי את קבוצת האימון S ל- S_{train} (70% מה- data) ול- S_{cv} (שאר ה- 30%). (כאן קבוצת ה- S_{cv} נקראת קבוצת ה- Hold-out cross-validation).



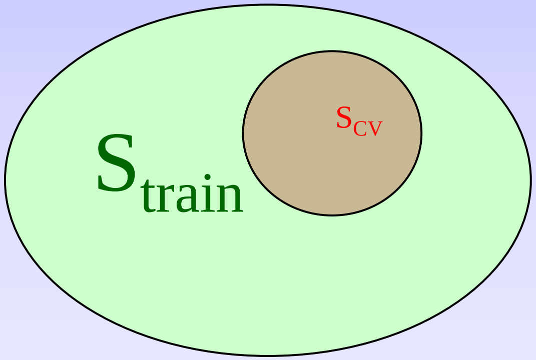
אימות מצולב (Cross-Validation)

2. מאמנים כל מודל M_i על S_{train} ומקבלים היפותיזה כלשהי h_i



אימות מצולב (Cross-Validation)

3. בוחרים ומוצאים כפלט את ההיפותיזה h_i עבורה השגיאה $\hat{\varepsilon s}_{cv}(h_i)$ היא הקטנה ביותר על קבוצת ה- Hold-out cross-validation, כאשר $\hat{\varepsilon s}_{cv}(h_i)$ מסמנת את שגיאת האימון של h על קבוצת הדוגמאות S_{cv}.



Hold-out cross-validation

- על-ידי בחינת המודלים על data עליה לא התאמנו, משיגים שערוך טוב יותר של שגיאת ההכללה האמיתית עבור ההיפותיזות h_i,
- אפשר לבחור את ההיפותיזה עם שגיאת ההכללה הקטנה ביותר.

Hold-out cross-validation

- בדרך-כלל מוציאים 1/3-1/4 מה- data, כאשר 30% היא בחירה אופיינית.

Hold-out cross-validation

- אפשרות אופציונלית: בצעד 3 של האלגוריתם – אפשר להחליף את הצעד על-ידי בחירת המודל M_i בהתאם לשגיאת ה- Hold-out cross-validation המינימלית:

$$\arg\min_i \hat{\varepsilon}s_{cv}(h_i)$$

ואז אימון מחדש על כל קבוצת האימון S.

(בדרך-כלל – רעיון טוב, עם יוצא דופן – אלגוריתמי למידה הרגישים לתנאי התחלה או לנתונים)

Hold-out cross-validation

- חיסרון: בזבוז של 30% מה- data.
- גם אם משתמשים בצעד האופציונלי – עדיין מאמנים כל מודל על 70% מהדוגמאות.
- מתאים עבור מספר דוגמאות גדול.
- בבעיות למידה עם מספר דוגמאות קטן (נניח בעייה עם m=20), נרצה לבצע משהו אחר.

שיטת ה- K-fold Cross-Validation

- בשיטה זו מוציאים כל פעם פחות data:

1. מחלקים את ה- data באופן אקראי ל- k תת-קבוצות זרות שבכל אחת מהן m/k דוגמאות אימון. נסמן קבוצות אלה ב- $S_1, S_2, \ldots, S_k$
2. עבור כל מודל M_i מעריכים אותו באופן הבא:

For j=1,2,…,k

על M_i אמנו את המודל ה-

$$S_1 \cup S_2 \cup \ldots, S_{j-1} \cup S_{j+1} \cup \ldots, \cup S_k$$

(כלומר על כל קבוצות ה- data פרט ל- S_j כדי לקבל היפותיזה כלשהי h_{ij}).

בחנו את ההיפותיזה h_{ij} על S_j וקבלו את השגיאה $\hat{\varepsilon}s_j(h_{ij})$

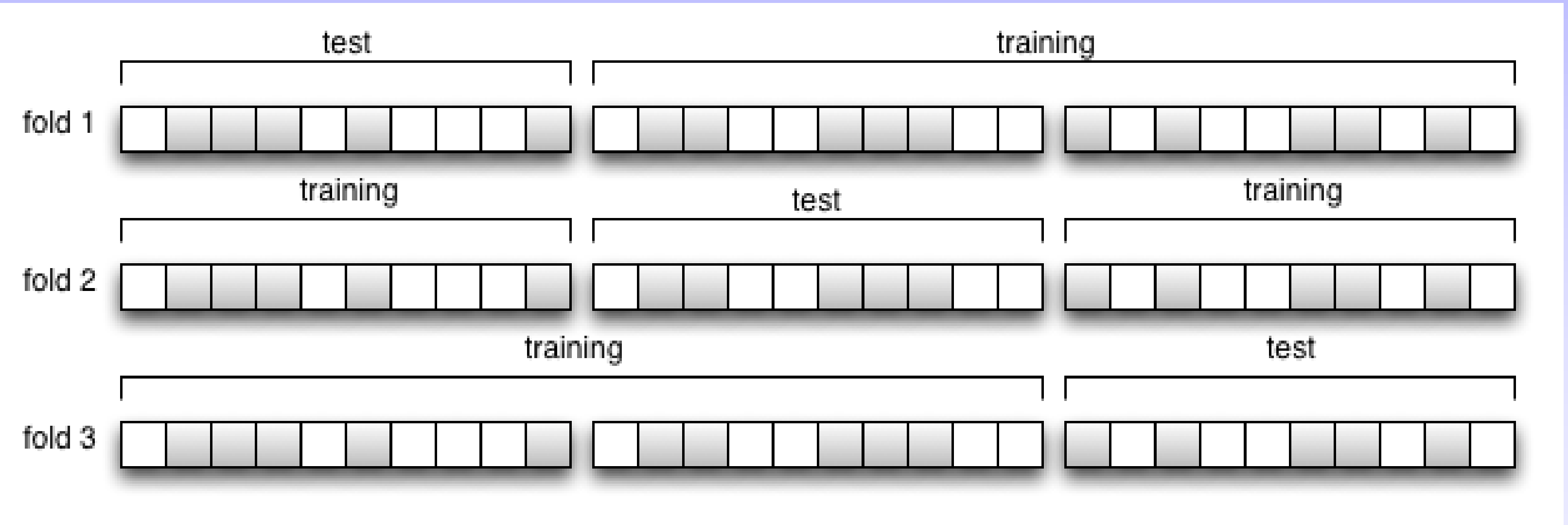
שיטת ה- K-fold Cross-Validation

2. (המשך) מחשבים את **שגיאת ההכללה** של המודל ה- M_i כממוצע של השגיאה $\hat{\varepsilon s}_{cv}(h_i)$ על כל ה- j –ים.
3. בוחרים את המודל M_i עם שגיאת ההכללה הקטנה ביותר, ומאמנים מחדש את המודל הנבחר על כל קבוצת האימון S.

מוציאים לפלט את התשובה הסופית – ההיפותיזה h.

שיטת ה- K-fold Cross-Validation

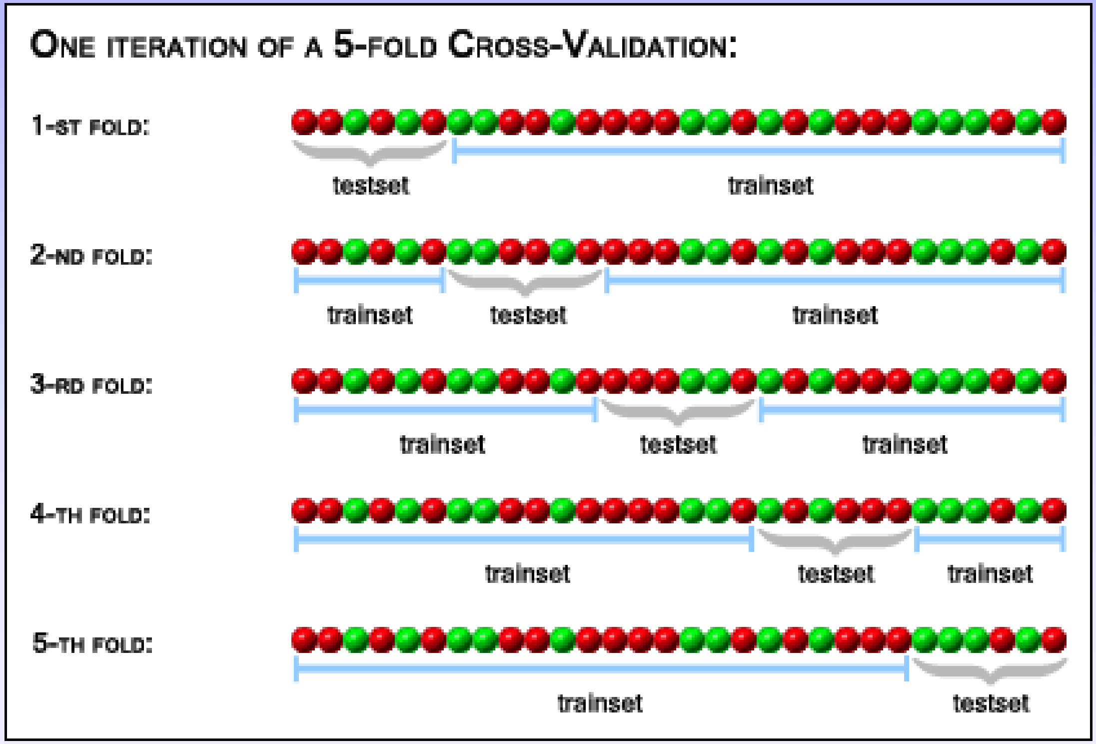
- בחירה אופיינית של מספר ה-folds: k=10



3-fold cross validation

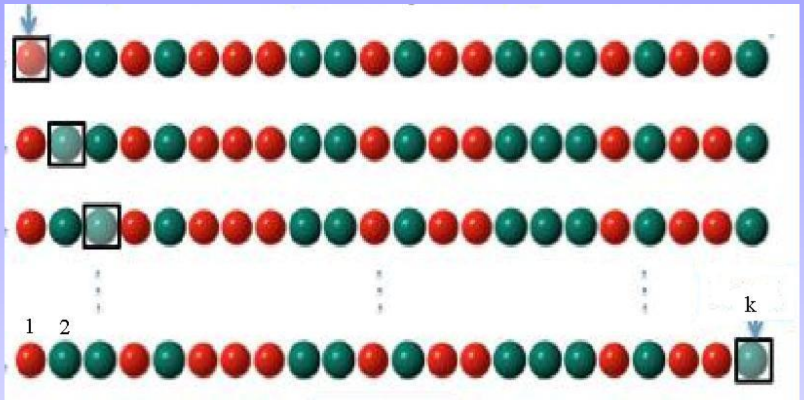
K-fold Cross-Validation

- למרות שמוציאים עכשיו מה- data רק 1/k מהדוגמאות, כלומר הרבה פחות מקודם, החישוביות כאן יקרה יותר מאשר hold out CV, כי עכשיו מאמנים כל מודל k פעמים:



Leave-one-out Cross-Validation

- במקרה שהדוגמאות מאוד מעטות – נשתמש לעתים במקרה הקיצוני של k=m, כלומר נוציא כל פעם מקבוצת האימון רק דוגמא אחת.



- במקרה זה נאמן בכל פעם את המודל על קבוצת האימון S שתכיל את כל הדוגמאות למעט אחת, ונבחן על הדוגמא היחידה אותה הוצאנו מקבוצת האימון S.
- מספר השגיאות מתוך הסה"כ משערך את שגיאת ההכללה.

Leave-one-out Cross-Validation

- הראינו איך אפשר להשתמש בשיטות ה- cross-validation כדי לבחור במודל מתאים מתוך קבוצת מודלים
- אפשר להשתמש בשיטות כדי לשערך מודל יחיד – אם לדוגמא מממשים אלגוריתם למידה (או מפתחים אלגוריתם חדש) ורוצים לבחון אותו על קבוצת נתונים, שימוש ב- cross-validation הוא דרך הגיונית לעשות זאת.

ברירת תכונות (Feature selection)

- נניח בעיית לימוד מודרכת, עם מספר תכונות n גדול מאוד, כך ש- n>>m.
- הנחה: רק מספר קטן של תכונות הן רלבנטיות או תורמות למשימת הלמידה.
- במקרה כזה, אם מספר דוגמאות האימון לא גדול מאוד, עשוייה להיווצר בעייה של התאמת יתר (overfitting).

Why Reduce Dimensionality?

- Reduces time complexity: Less computation
- Reduces space complexity: Less parameters
- Saves the cost of observing the feature
- Simpler models are more robust on small datasets
- More interpretable; simpler explanation
- Data visualization (structure, groups, outliers, etc) if plotted in 2 or 3 dimensions

Lecture Notes for E Alpaydın 2010 Introduction to Machine Learning 2e © The MIT Press (V1.0)

ברירת תכונות (Feature selection)

- עבור n תכונות 2^n תת-קבוצות של תכונות אפשריות (כל אחת מהתכונות כלולה או לא כלולה בתת-הקבוצה).
- אפשר לראות את בעיית ברירת התכונות כבעיית בחירת מודל מתוך סה"כ 2^n מודלים.
- עבור מספר גדול של תכונות n תהליך ההערכה והבחירה עשוי להיות חישובית יקר מדי, ולכן משתמשים בפרוצדורת חיפוש היוריסטית כדי לבחור תת-קבוצה מתאימה של תכונות.

Feature Selection vs Extraction

- Feature selection: Choosing $k<d$ important features, ignoring the remaining $d - k$

  Subset selection algorithms
- Feature extraction: Project the

  original x_i , i =1,...,d dimensions to

  new $k<d$ dimensions, z_j , j =1,...,k

  Principal components analysis (PCA), linear discriminant analysis (LDA), factor analysis (FA)

חיפוש קדימה (Forward search)

1. Initialize $\mathcal{F} = \emptyset$.
2. Repeat {
   (a) For $i = 1, \ldots, n$ if $i \notin \mathcal{F}$, let $\mathcal{F}_i = \mathcal{F} \cup \{i\}$, and use some version of cross validation to evaluate features $\mathcal{F}_i$. (I.e., train your learning algorithm using only the features in $\mathcal{F}_i$, and estimate its generalization error.)
   (b) Set $\mathcal{F}$ to be the best feature subset found on step (a).

   }
3. Select and output the best feature subset that was evaluated during the entire search procedure.

בשלב 3 בוחרים ומוציאים לפלט את תת-הקבוצה שהוערכה כטובה ביותר בכל פרוצדורת החיפוש.

Subset Selection

- There are 2^d subsets of d features
- Forward search: Add the best feature at each step
  - Set of features F initially Ø.
  - At each iteration, find the best new feature

    $j = \mathrm{argmin}_i \, E (F \cup x_i)$
  - Add x_j to F if $E (F \cup x_j) < E (F)$

- Hill-climbing $O(d^2)$ algorithm
- Backward search: Start with all features and remove one at a time, if possible.
- Floating search (Add k, remove l)

ברירת תכונות (Feature selection)

מודל אחר – Backward search

1. Initialization: $F=\{1,2,\ldots n\}$
2. Repeat – {

    a. For $i=1,2,\ldots,n$ if $i \not\subseteq F$

    let $F_i=\{ Fi-\{i\} \}$

    and use some version of cross-validation to evaluate features F_i that has the least significant contribution to the classification by estimation the generalization error by using the learning algorithm.

    b. Set F to be the best feature subset found on step (a)

    }

3. Output the best subset of features

Subset Selection

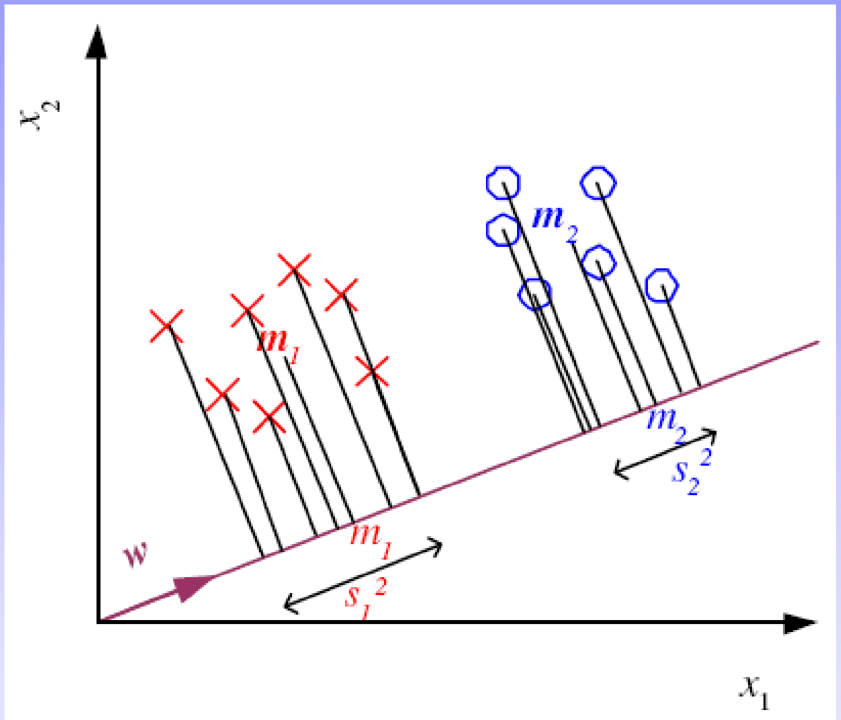
- Backward search: Start with all features and remove one at a time, if possible.
- Floating search (Add *k*, remove *l*)

ברירת תכונות (Feature selection)

חסרונות: עלות חישובית גבוהה (חיפוש קדמי מלא כרוך ב- $o(n^2)$ קריאות לאלגוריתם הלמידה)

Linear Discriminant Analysis

- Find a low-dimensional space such that when ***x*** is projected, classes are well-separated.
- Find ***w*** that maximizes



$$J(\mathbf{w}) = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2}$$

$$m_1 = \frac{\sum_t \mathbf{w}^T \mathbf{x}^t r^t}{\sum_t r^t} \qquad s_1^2 = \sum_t \left(\mathbf{w}^T \mathbf{x}^t - m_1\right)^2 r^t$$

- Between-class scatter:

$$
\begin{aligned}
(m_1 - m_2)^2 &= \left(\mathbf{w}^T\mathbf{m}_1 - \mathbf{w}^T\mathbf{m}_2\right)^2 \\
&= \mathbf{w}^T(\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T\mathbf{w} \\
&= \mathbf{w}^T\mathbf{S}_B\mathbf{w} \text{ where } \mathbf{S}_B = (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T
\end{aligned}
$$

- Within-class scatter:

$$
\begin{aligned}
s_1^2 &= \sum_t \left(\mathbf{w}^T\mathbf{x}^t - m_1\right)^2 r^t \\
&= \sum_t \mathbf{w}^T(\mathbf{x}^t - \mathbf{m}_1)(\mathbf{x}^t - \mathbf{m}_1)^T\mathbf{w}r^t = \mathbf{w}^T\mathbf{S}_1\mathbf{w}
\end{aligned}
$$

$$
\text{where } \mathbf{S}_1 = \sum_t (\mathbf{x}^t - \mathbf{m}_1)(\mathbf{x}^t - \mathbf{m}_1)^T r^t
$$

$$
s_1^2 + s_1^2 = \mathbf{w}^T\mathbf{S}_W\mathbf{w} \text{ where } \mathbf{S}_W = \mathbf{S}_1 + \mathbf{S}_2
$$

Fisher's Linear Discriminant

- Find ***w*** that max

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} = \frac{\left|\mathbf{w}^T (\mathbf{m}_1 - \mathbf{m}_2)\right|^2}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$

- LDA solution: $\mathbf{w} = c \cdot \mathbf{S}_W^{-1} (\mathbf{m}_1 - \mathbf{m}_2)$

- Parametric solution :

$$\mathbf{w} = \Sigma^{-1} (\mu_1 - \mu_2)$$

$$\text{when } p(\mathbf{x} \mid C_i) \sim \mathcal{N}(\mu_i, \Sigma)$$

K>2 Classes

- Within-class scatter:

$$\mathbf{S}_W = \sum_{i=1}^{K} \mathbf{S}_i \qquad \mathbf{S}_i = \sum_t r_i^t \left(\mathbf{x}^t - \mathbf{m}_i\right)\left(\mathbf{x}^t - \mathbf{m}_i\right)^T$$

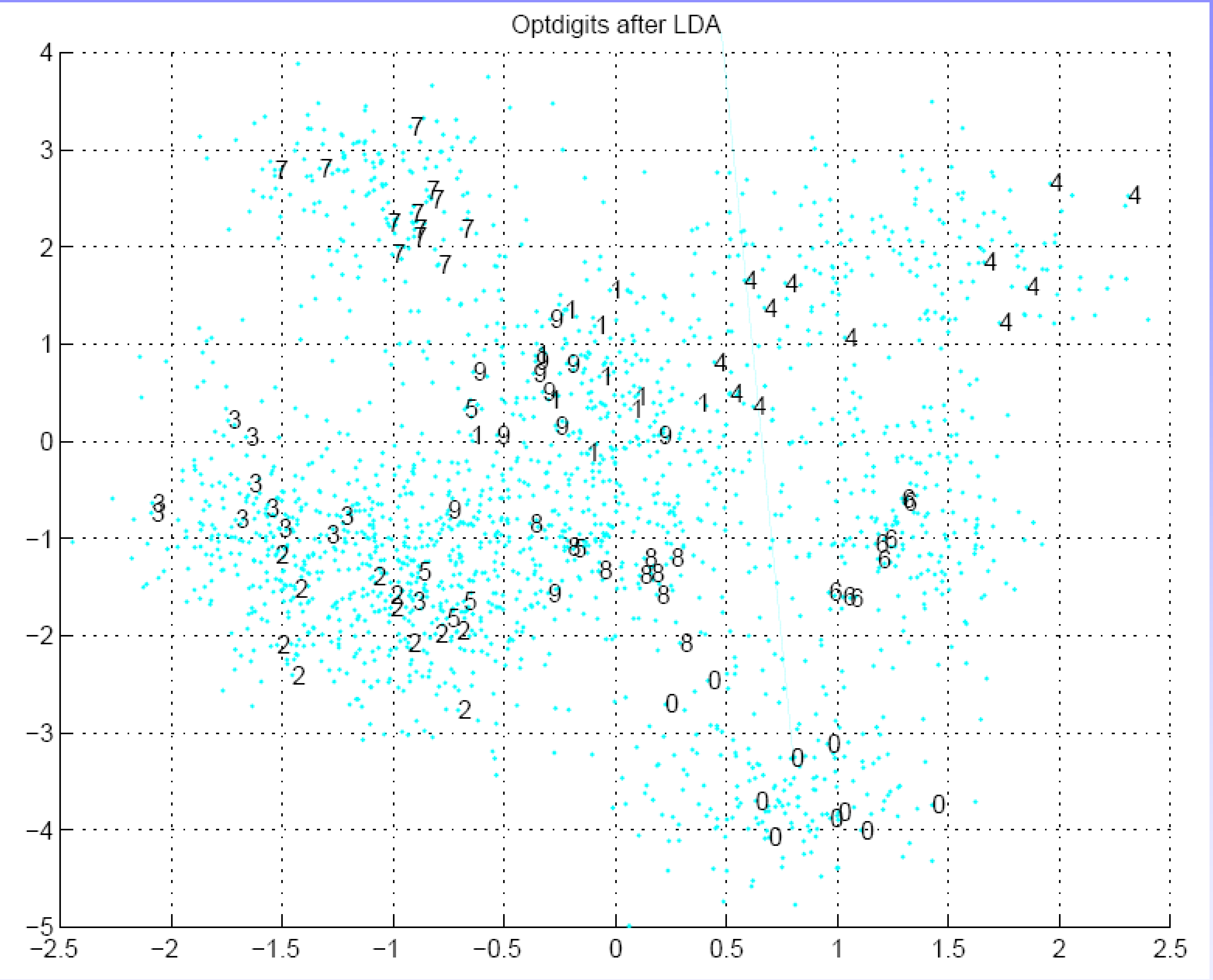
- Between-class scatter:

$$\mathbf{S}_B = \sum_{i=1}^{K} N_i \left(\mathbf{m}_i - \mathbf{m}\right)\left(\mathbf{m}_i - \mathbf{m}\right)^T \qquad \mathbf{m} = \frac{1}{K}\sum_{i=1}^{K} \mathbf{m}_i$$

- Find **W** that max

$$J(\mathbf{W}) = \frac{\left|\mathbf{W}^T \mathbf{S}_B \mathbf{W}\right|}{\left|\mathbf{W}^T \mathbf{S}_W \mathbf{W}\right|}$$

The largest eigenvectors of $\mathbf{S}_W^{-1}\mathbf{S}_B$
Maximum rank of K-1


Optdigits after LDA