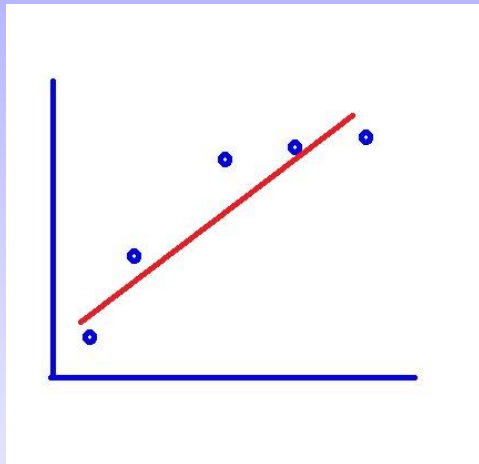


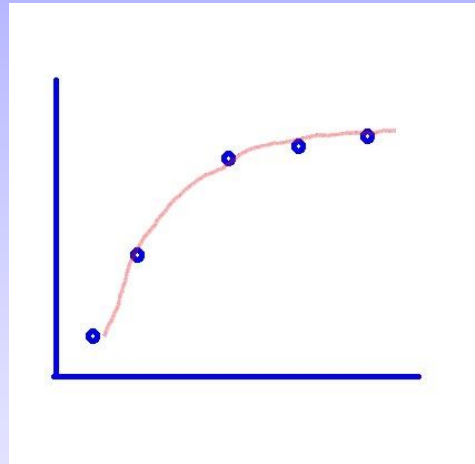
רגולריזציה

- בעיית התאמת היתר – overfitting
- דוגמא רגרסיה לינארית:



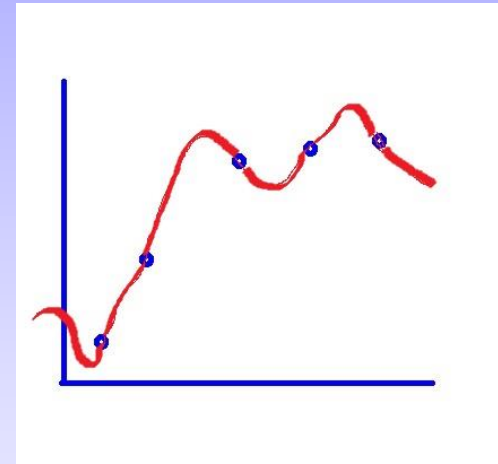
$$\theta_0 + \theta_1 x$$

התאמת חסר
underfitting
high bias



$$\theta_0 + \theta_1 x + \theta_2 x^2$$

מודל מתאים
right model



$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \dots$$

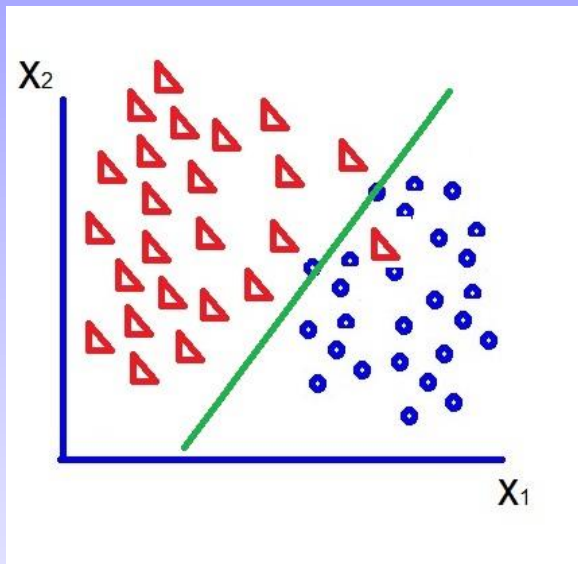
התאמת יתר
overfitting
high variance

רגולריזציה

- אם יש יותר מדי תכונות היפותזת הלמידה או פונקציית החיזוי עשויה להתאים לקבוצת האימון בצורה מושלמת, כלומר $J(\theta)=0$, אבל יכולת ההכללה לדוגמאות חדשות היא נמוכה.
- הכללה (generalization)

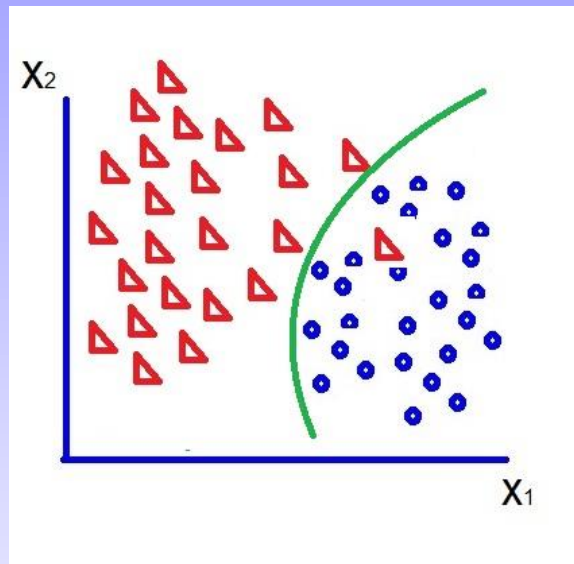
רגולריזציה

• דוגמא: רגרסיה לוגיסטית

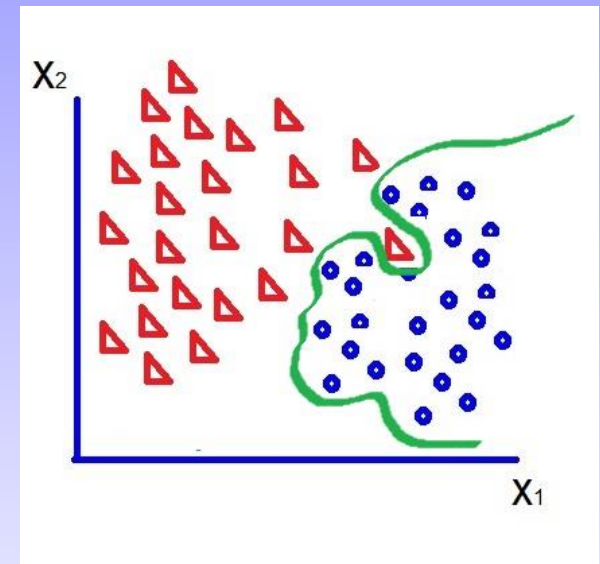


$$h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

underfitting



$$h_{\theta}(x) = g\left(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots\right)$$

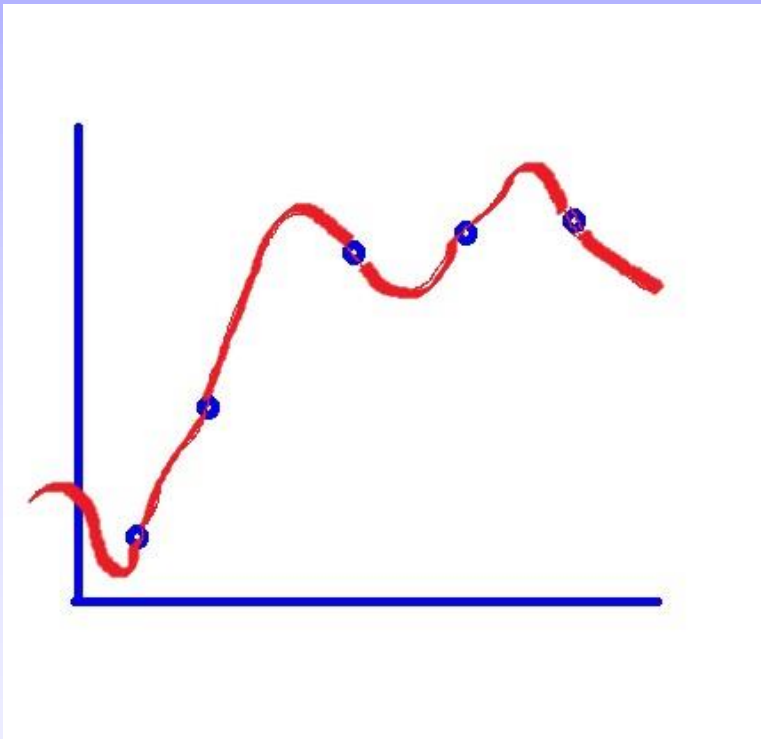


$$h_{\theta}(x) = g\left(\begin{array}{l} \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \dots \\ \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 \end{array}\right)$$

overfitting

רגולריזציה

- מה אפשר לעשות כדי להחליט מהי ההיפותזה הנכונה?
- אם יש רק שתי תכונות, אפשר לצייר את עקום הרגרסיה ולהחליט. לדוגמא עבור הרגרסיה הליניארית:



רגולריזציה

אבל עבור מספר רב של תכונות (לדוגמא עבור בעיית הגייזר):

x_1 – משך ההתפרצות

x_2 – עצמת ההתפרצות

x_3 – שטח הפיזור

x_4 – גובה ההתפרצות

x_5 – מרחק ההתפרצות

הרבה יותר קשה לראות מהו העקום אותו צריך להעביר.

תכונות רבות ומעט דוגמאות \leftarrow ייתכן שיתקבל **overfitting**

רגולריזציה

- מענה לבעיית התאמת היתר:

1. הפחתת מספר התכונות

א. החלטה באופן ידני איזה תכונות לשמור.

ב. הפעלת אלגוריתם אוטומטי לבחירת התכונות

חיסרון – הפחתת מספר התכונות עשויה לפגוע בביצועי המסווג, התכונות עשויות להכיל מידע חשוב לחיזוי (לדוגמא משך ההתפרצות בדוגמת הגייזר הנאמן או שטח הבית בדוגמת הבתים).

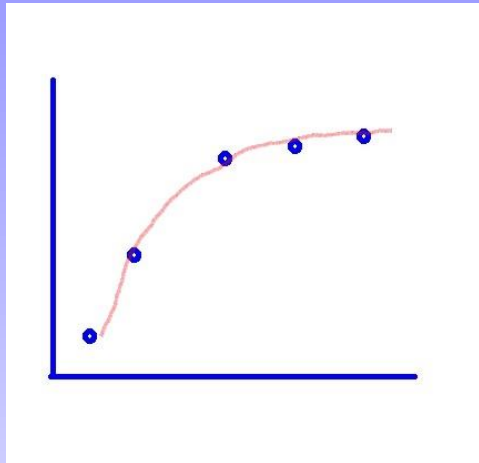
2. רגולריזציה (Regularization):

שומרים על כל התכונות, אבל מורידים את ערכי הפרמטרים θ .

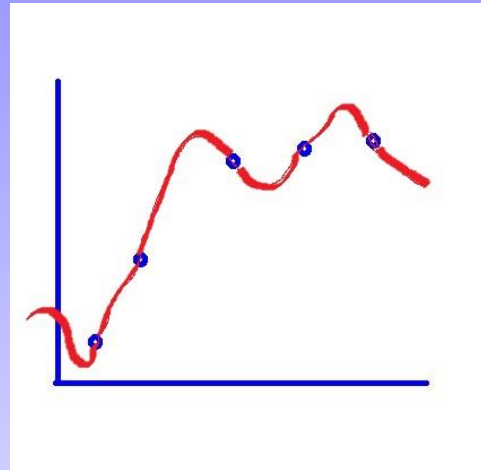
היתרון – שיטה זו עובדת היטב כאשר יש מספר רב של תכונות, וכל אחת מהן תורמת לחיזוי y .

רגולריזציה – פונקציית המחיר

- ראינו בדוגמאות הקודמות (רגרסיה לינארית):



$$\theta_0 + \theta_1 x + \theta_2 x^2$$



$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \dots$$

- בחירה בהיפותזה השמאלית – התאמה די טובה.
- בחירה בהיפותזה הימנית – התאמת יתר.
- נניח כי "מענישים" על-ידי הוספת גורם קנס לפונקציית המחיר כך שהערך של θ_3, θ_4 יהיה מאוד קטן.

רגולריזציה – פונקציית המחיר

- נתבונן בפונקציית המחיר:

$$\min_{\theta} \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

- נוסיף גורם עונש ל- θ_3, θ_4 :

$$\min_{\theta} \left(\frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + 10^6 \theta_3^2 + 10^6 \theta_4^2 \right)$$

מספר גדול

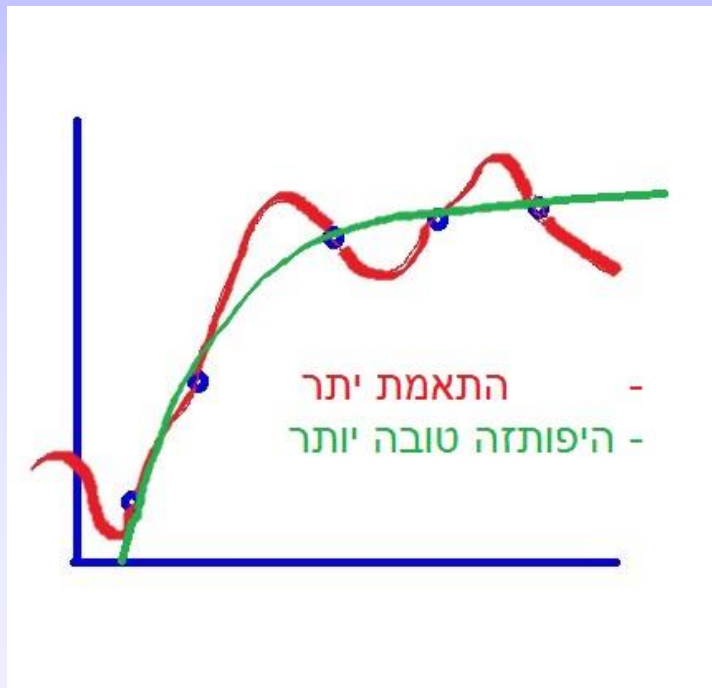
- מהו ערכם של θ_3, θ_4 לאחר מזעור פונקציית המחיר?

רגולריזציה – פונקציית המחיר

- לאחר מזעור פונקציית המחיר:
 $\theta_3 \approx 0, \quad \theta_4 \approx 0$

נקבל ביטוי קרוב לריבועי (קוודרטי)

$$\theta_0 + \theta_1 x + \theta_2 x^2 + \cancel{\theta_3 x^3} + \cancel{\theta_4 x^4}$$



- ערכים קטנים עבור כל הפרמטרים מביאים להיפותרזה פשוטה יותר ופונקציה חלקה יותר.
- ההיפותרזה פחות נוטה ל-overfitting

רגולריזציה – פונקציית המחיר

- אפשר להראות שערכים קטנים עבור כל ערכי הפרמטרים מביאה להיפותזה פשוטה ופונקציה חלקה יותר $\theta_0, \theta_1, \dots, \theta_n$
- דוגמא: מחירי הבתים: x_0, x_1, \dots, x_{99}
- איך נדע איזה פרמטרים צריך לבחור ואת איזה לכווץ? $\theta_0, \theta_1, \dots, \theta_{99}$
- הפתרון: כווץ כל הפרמטרים.

$$\min_{\theta} \left(\frac{1}{2m} \left(\sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n \theta_j^2 \right) \right)$$

- (לא מקובל לכלול את θ_0 . ההבדל מאוד קטן אם כוללים או לא).

רגולריזציה – פונקציית המחיר

- הפתרון: כוּוץ כל הפרמטרים.

$$\min_{\theta} \left(\frac{1}{2m} \left(\sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n \theta_j^2 \right) \right)$$

- (לא מקובל לכלול את θ_0) למרות שזו לא שגיאה "להעניש" גם את θ_0)
- ההבדל מאוד קטן אם כוללים או לא).

רגולריזציה – פונקציית המחיר

$$\min_{\theta} \left(\frac{1}{2m} \left(\sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n \theta_j^2 \right) \right)$$

λ - פרמטר הרגולציה – פשרה בין שתי מטרות שונות:

(1) התאמה לנתוני האימון

(2) מניעת התאמת יתר על-ידי הקטנת הפרמטרים.

- λ - שולטת על הפשרה.
- הרגולריזציה מאפשרת להשתמש בכל התכונות מבלי לגרום להתאמת יתר.
- אפשר לכלול פולינום מסדר גבוה.
- איך נראית הפונקציה הנוצרת? לא קוודרטית, אך עדיין חלקה ומתאימה לנתונים.

רגולריזציה – פונקציית המחיר

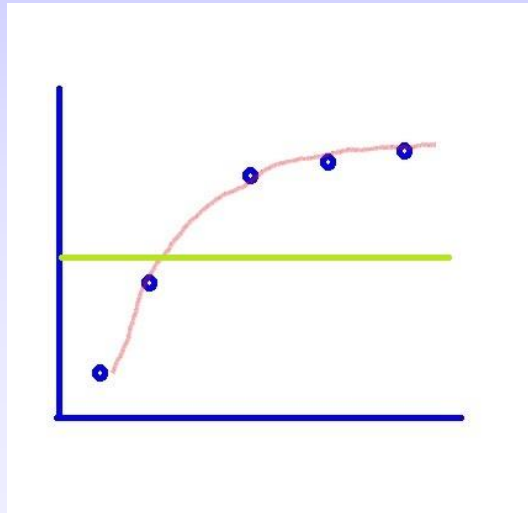
$$\min_{\theta} \left(\frac{1}{2m} \left(\sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n \theta_j^2 \right) \right)$$

מה יקרה אם נבחר λ מאוד גבוה? $\lambda = 10^{10}$

נניח ערך זה עבור המודל:

$$h_{\theta}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \theta_5 x^5$$

θ_0



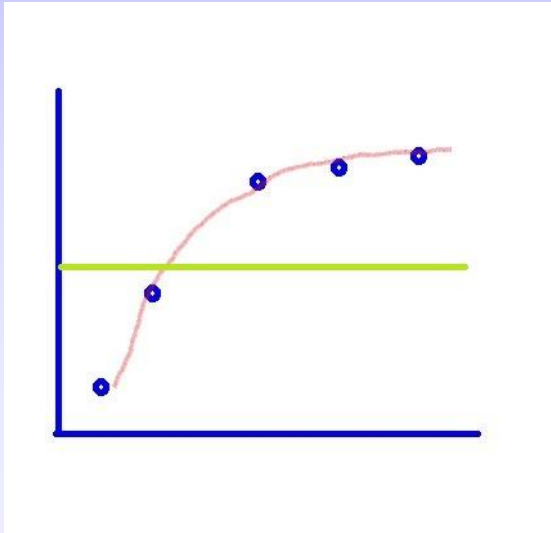
רגולריזציה – פונקציית המחיר

מה יקרה אם נבחר λ מאוד גבוה? $\lambda = 10^{10}$

$$\theta_1 \approx 0, \quad \theta_2 \approx 0, \quad \theta_3 \approx 0, \quad \theta_4 \approx 0, \quad \theta_5 \approx 0$$

$$h_{\theta}(x) = \theta_0 + \cancel{\theta_1 x} + \cancel{\theta_2 x^2} + \cancel{\theta_3 x^3} + \cancel{\theta_4 x^4} + \cancel{\theta_5 x^5}$$

הישר בשיפוע 0 יוצר התאמת חסר, לפיכך דרושה בחירה טובה יותר של פרמטר הרגולציה λ



רגרסיה ליניארית עם רגולריזציה

(Regularized Linear Regression)

נכליל את אלגוריתם ה-gd כך שיכלול גם ביטוי לרגולריזציה

$$J(\theta) = \frac{1}{2m} \left(\sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n \theta_j^2 \right)$$

Regularization term

רוצים למצוא פרמטרים למזעור פונקציית המחיר.

רגרסיה ליניארית עם רגולריזציה

(Regularized Linear Regression)

הסבר:

$$\begin{aligned}\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \left(\frac{1}{2m} \left(\sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{k=1}^n \theta_k^2 \right) \right) = \\ &= \frac{1}{2m} \left(\sum_{i=1}^m 2(h_{\theta}(x^{(i)}) - y^{(i)}) \frac{\partial}{\partial \theta_j} h_{\theta}(x^{(i)}) + 2\lambda \theta_j \right) = \\ &= \frac{1}{m} \left(\sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)} \right) + \frac{\lambda}{m} \theta_j = \\ &= -\frac{1}{m} \left(\sum_{i=1}^m (y^{(i)} - h_{\theta}(x^{(i)})) x_j^{(i)} \right) + \frac{\lambda}{m} \theta_j =\end{aligned}$$

רגרסיה ליניארית ללא רגולריזציה

(Unregularized Linear Regression)

Gradient Descent : ללא רגולריזציה

{ Repeat (until convergence)

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)} \quad j = 0, 1, 2, \dots, n$$

}

רגרסיה ליניארית ללא רגולריזציה

(unregularized Linear Regression)

עתה נתייחס ל- θ_0 לחוד (מאחר ובגורם הרגולריזציה הוא לא מופיע)

{ Repeat (until convergence)

$$\theta_0 := \theta_0 + \alpha \frac{1}{m} \left(\sum_{i=1}^m (y^{(i)} - h_{\theta}(x^{(i)})) x_0^{(i)} \right)$$

$$\theta_j := \theta_j + \alpha \frac{1}{m} \left(\sum_{i=1}^m (y^{(i)} - h_{\theta}(x^{(i)})) x_j^{(i)} \right) \quad j = \cancel{0}, 1, 2, \dots, n$$

}

הסימן של α השתנה כי הפכנו את הסדר עבור ההפרש בין $y^{(i)}$ לפונקציית ההיפותזה

רגרסיה ליניארית עם רגולריזציה

(Regularized Linear Regression)


נוסיף את גורם הרגולריזציה:

{ Repeat (until convergence)

$$\theta_0 := \theta_0 + \alpha \frac{1}{m} \left(\sum_{i=1}^m (y^{(i)} - h_{\theta}(x^{(i)})) x_0^{(i)} \right)$$

$$\theta_j := \theta_j + \alpha \left(\frac{1}{m} \sum_{i=1}^m (y^{(i)} - h_{\theta}(x^{(i)})) x_j^{(i)} - \frac{\lambda}{m} \theta_j \right) \quad j = 1, 2, \dots, n$$

}


$$\frac{\partial}{\partial \theta_j} J(\theta)$$

רגרסיה ליניארית עם רגולריזציה

(Regularized Linear Regression)

או באופן אקוויולנטי :

{ Repeat (until convergence)

$$\theta_0 := \theta_0 + \alpha \frac{1}{m} \sum_{i=1}^m (y^{(i)} - h_{\theta}(x^{(i)})) x_0^{(i)}$$

$$\theta_j := \theta_j \left(1 - \alpha \frac{\lambda}{m} \right) + \alpha \frac{1}{m} \left(\sum_{i=1}^m (y^{(i)} - h_{\theta}(x^{(i)})) x_j^{(i)} \right) \quad j = 1, 2, \dots, n$$

}

רגרסיה ליניארית עם רגולריזציה

(Regularized Linear Regression)

$$\text{קל לראות: } \left(1 - \alpha \frac{\lambda}{m}\right) < 1$$

ולכן החלק השמאלי של הביטוי מתכווץ לאחר כל איטרציה.

מכווצים את הפרמטרים θ_j מעט לאחר כל איטרציה

{ Repeat (until convergence)

$$\theta_0 := \theta_0 + \alpha \frac{1}{m} \sum_{i=1}^m (y^{(i)} - h_{\theta}(x^{(i)})) x_0^{(i)}$$

$$\theta_j := \theta_j \left(1 - \alpha \frac{\lambda}{m}\right) + \alpha \left(\sum_{i=1}^m (y^{(i)} - h_{\theta}(x^{(i)})) x_j^{(i)} \right) \quad j = 1, 2, \dots, n$$

}

החלק השני – כמו קודם ללא הרגולריזציה.

* - מניחים כי α קטן ו- m יחסית גדול

רגרסיה לוגיסטית עם רגולריזציה

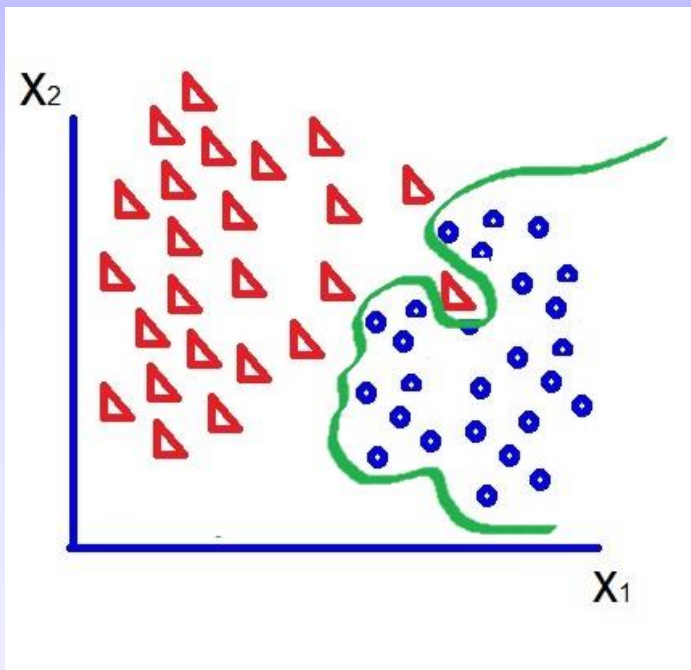
(Regularized Logistic Regression)

ראינו כבר כי שימוש בפונקציית היפותזה מורכבת

כמו:

$$h_{\theta}(x) = g \left(\begin{aligned} &\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \dots \\ &\theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 \end{aligned} \right)$$

עשוי לגרום ל- overfitting



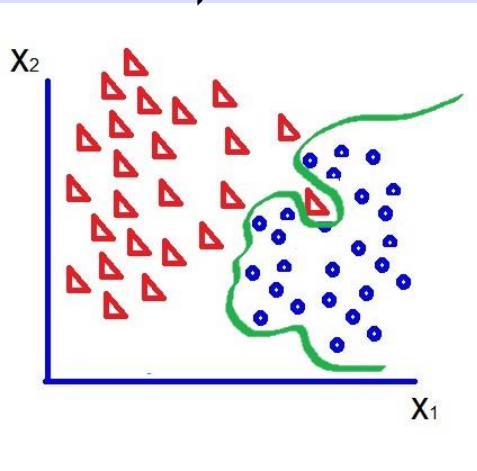
רגרסיה לוגיסטית עם רגולריזציה

(Regularized Logistic Regression)

פונקציית המחיר:

$$J(\theta) = \left[\frac{1}{m} \left(\sum_{i=1}^m -y^{(i)} \log(h_{\theta}(x^{(i)})) - (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)})) \right) \right]$$

- שימוש בתכונות רבות (לאו דווקא פולינומיליות) עשוי לגרום למצב של התאמת יתר.
- כדי להימנע ממצב זה נוסיף איבר לפונקציית המחיר שיגרום לכך שהפרמטרים לא יהיו יותר מדי גדולים.

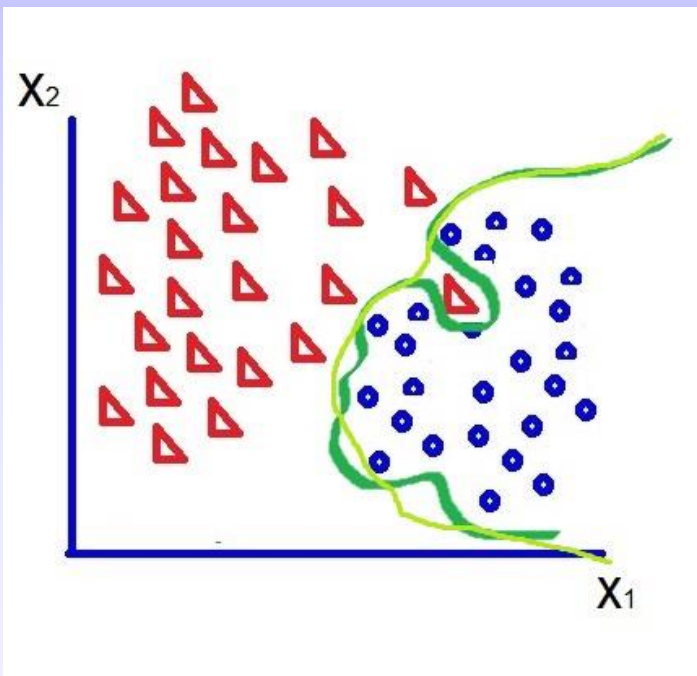


רגרסיה לוגיסטית עם רגולריזציה

(Regularized Logistic Regression)

נוסיף איבר לפונקציית המחיר:

$$J(\theta) = - \left[\frac{1}{m} \sum_{i=1}^m y^{(i)} \log(h_{\theta}(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)})) \right] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

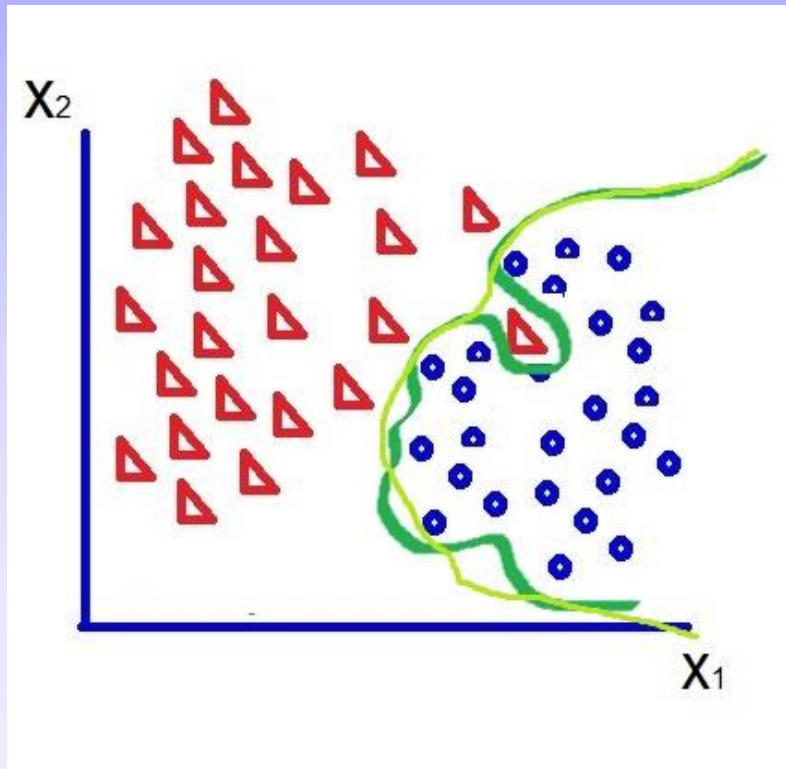


- במקרה זה נמנע מהתאמת היתר גם אם יש הרבה מאוד תכונות, וגבול ההחלטה יהיה חלק יותר.

רגרסיה לוגיסטית עם רגולריזציה

(Regularized Logistic Regression)

- לסיכום: הרגולריזציה מאפשרת שימוש במספר רב של תכונות ומונעת התאמת יתר.



רגרסיה לוגיסטית עם רגולריזציה

(Regularized Logistic Regression)

- באמצעות Gradient Descent כאשר את θ_0 נעדכן לחוד, ונוסיף את גורם הרגולריזציה.

Repeat {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m \left(h_{\theta}(x^{(i)}) - y^{(i)} \right) x_j^{(i)}$$

}

רגרסיה לוגיסטית עם רגולריזציה

(Regularized Logistic Regression)

• כלומר:

Repeat {

$$\theta_0 := \theta_0 - \alpha \left[\frac{1}{m} \sum_{i=1}^m \left(h_{\theta}(x^{(i)}) - y^{(i)} \right) x_0^{(i)} \right]$$

$$\theta_j := \theta_j - \alpha \left[\frac{1}{m} \sum_{i=1}^m \left(h_{\theta}(x^{(i)}) - y^{(i)} \right) x_j^{(i)} + \frac{\lambda}{m} \theta_j \right]$$

$$j = \cancel{0} \quad 1, 2, \dots, n$$

}

רגרסיה לוגיסטית עם רגולריזציה

(Regularized Logistic Regression)

Repeat {

$$\theta_0 := \theta_0 - \alpha \left[\frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_0^{(i)} \right]$$

$$\theta_j := \theta_j - \alpha \left[\frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)} + \frac{\lambda}{m} \theta_j \right]$$

$$j = 1, 2, \dots, n$$

}

• הערה: הביטוי בסוגריים הוא הנגזרת לפי θ_j של $J(\theta)$

$$h_{\theta}(x) = \frac{1}{1 + e^{-\theta^T x}}$$

רגרסיה לוגיסטית עם רגולריזציה

(Regularized Logistic Regression)

Repeat {

$$\theta_0 := \theta_0 - \alpha \left[\frac{1}{m} \sum_{i=1}^m \left(h_{\theta}(x^{(i)}) - y^{(i)} \right) x_0^{(i)} \right]$$

$$\theta_j := \theta_j - \alpha \left[\frac{1}{m} \sum_{i=1}^m \left(h_{\theta}(x^{(i)}) - y^{(i)} \right) x_j^{(i)} + \frac{\lambda}{m} \theta_j \right]$$

$$j = \cancel{0} \quad 1, 2, \dots, n$$

}

- כלל זה נראה זהה לכלל העדכון עם הרגולריזציה אותו בצענו עבור רגרסיה ליניארית, אבל, אבל כמובן זה לא אותו כלל, כיון שפונקציית ההיפותיזה היא פונקציה לוגיסטית.

$$h_{\theta}(x) = \frac{1}{1 + e^{-\theta^T x}}$$

רגרסיה לוגיסטית עם רגולריזציה

(Regularized Logistic Regression)

- נשתמש בפונקציית ה-Matlab לביצוע אופטימיזציה (מינימיזציה) ללא אילוצים:

fminunc

פונקצייה זו מוצאת את המינימום של $f(x)$: $\min_x f(x)$

כאשר $f(x)$ היא פונקציה המחזירה סקלר, ו- x הוא וקטור או מטריצה.

הפונקציה מבצעת **אופטימיזציה לא ליניארית ללא אילוצים** (unconstained nonlinear optimization).

(הערה: אופטימיזציה עם אילוצים – הכוונה מציאה של אופטימום כאשר יש איזשהו אילוץ של הפרמטרים, כמו חסם על ערכי הפרמטרים, לדוגמא $\theta > 2$ או $\|\theta\|=1$)

רגרסיה לוגיסטית עם רגולריזציה

(Regularized Logistic Regression)

במקרה של רגרסיה לוגיסטית נרצה למזער את פונקציית המחיר $J(\theta)$ עם הפרמטרים θ , המזעור יתבצע עם קבוצת דוגמאות האימון X, y - קבוצת דוגמאות קבועה.

הסינטקס של הפונקציה `fminunc`:




רגרסיה לוגיסטית עם רגולריזציה

(Regularized Logistic Regression)

עבור רגרסיה לוגיסטית נשתמש בפונקציה הנקראת `costF_log` המחשבת בכל איטרציה את פונקציית המחיר J ואת הגרדיאנט לכל θ .

```
options = optimset ( 'GradObj' , 'on' , 'MaxIter' , 1000 ) ;
```



מודיע לפונקציה שהפונקציה
שלנו (`costF_log`) מחזירה
גם את המחיר וגם את
הגרדיאנט



מספר מקסימלי של
איטרציות

```
[theta, J]= fminunc ( @(t) (costF_log(t,X,y)), theta0, options) ;
```


דוגמא

```
% Initialize fitting parameters
initial_theta = zeros(size(X, 2), 1);
% Set regularization
lambda = 10;

% Set Options
options = optimset('GradObj', 'on', 'MaxIter', 5000);

% Optimize
[theta, J, exit_flag] = ...
    fminunc(@(t)(costFunctionReg(t, X, y, lambda)), initial_theta, options);
```

פונקציות אנונימיות

Anonymous Functions

What Are Anonymous Functions?

- A function that is *not* stored in a file, but is associated with a variable whose data type is `function_handle`.
- Anonymous functions can accept inputs and return outputs, just as standard functions.
- However, they can contain only a single executable statement.

Example: create a handle to an anonymous function

```
sqr = @(x) x.^2;
```

פונקציות אנונימיות

- Variable **sqr** is a function handle.
- The @ operator - creates the handle
- the parentheses () after the @ operator - include the function input arguments.
- The anonymous function in this case accepts a single input x, and implicitly returns a single output, an array the same size as x that contains the squared values.

What Is a Function Handle?

- A function handle is a MATLAB® **data type** that stores an **association** to a function. Indirectly calling a function enables you to invoke the function regardless of where you call it from.

Typical uses of function handles include:

- Pass a function to another function (often called *function functions*). For example, passing a function to integration and optimization functions, such as `integral` and `fzero`.
- Specify callback functions. For example, a callback that responds to a UI event or interacts with data acquisition hardware.
- Construct handles to functions defined inline instead of stored in a program file (anonymous functions).
- Call local functions from outside the main function.

Creating Function Handles

- You can see if a variable, `h`, is a function handle using `isa(h,'function_handle')`.
- To create a handle for a function, precede the function name with an `@` sign. For example, if you have a function called `myfunction`, create a handle named `f` as follows:
 - `f = @myfunction;`

Calling a function handle

- You call a function using a handle the same way you call the function directly.
- For example, suppose that you have a function named `computeSquare`, defined as:
- `function y = computeSquare(x) y = x.^2; end`
- Create a handle and call the function to compute the square of four.
- `f = @computeSquare; a = 4; b = f(a)`
- `b = 16`

פונקציות אנונימיות

- Find the square of a particular value (5) by passing the value to the function handle, just as you would pass an input argument to a standard function.

```
a = sqr(5)
```

```
>> a = 25
```

Example:

- find the integral of the sqr function from 0 to 1 by passing the function handle to the integral function:

```
q = integral(sqr,0,1);
```


פונקציות אנונימיות

- No need to create a variable in the workspace to store an anonymous function.
- Instead, you can create a temporary function handle within an expression, such as this call to the integral function:

```
>> q = integral(@(x) x.^2,0,1);
```

פונקציות אנונימיות

Variables in the Expression

- Function handles can store not only an expression, but also variables that the expression requires for evaluation.
- For example, create a function handle to an anonymous function that requires coefficients a, b, and c.

```
>> a = 1.3; b = .2; c = 30;
```

```
>> parabola = @(x) a*x.^2 + b*x + c;
```

פונקציות אנונימיות

- **Functions with Multiple Inputs or Outputs**
- Anonymous functions require that you explicitly specify the input arguments as you would for a standard function, separating multiple inputs with commas.
- Example: a function with two inputs, x and y:

```
>> myfunction = @(x,y) (x^2 + y^2 + x*y); x = 1; y = 10; z =  
>> myfunction(x,y)
```



```
>> Z=111
```

פונקציות אנונימיות

- However, you do not explicitly define output arguments when you create an anonymous function. If the expression in the function returns multiple outputs, then you can request them when you call the function. Enclose multiple output variables in square brackets.
- Example: the `ndgrid` function can return as many outputs as the number of input vectors.
- This anonymous function that calls `ndgrid` can also return multiple outputs:

```
>> c = 10; mygrid = @(x,y) ndgrid((-x:x/c:x),(-y:y/c:y));  
>> [x,y] = mygrid(pi,2*pi);
```

פונקציות אנונימיות

- The output from mygrid can be used to create a mesh or surface plot:

```
>> z = sin(x) + cos(y);
```

```
>> mesh(x,y,z)
```

