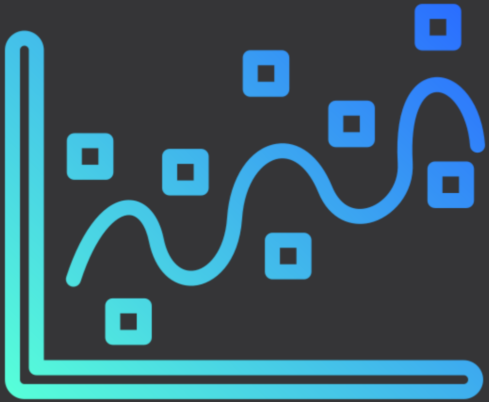


Regularization
The Problem of Overfitting

Lecture 11 – HCCDA-AI

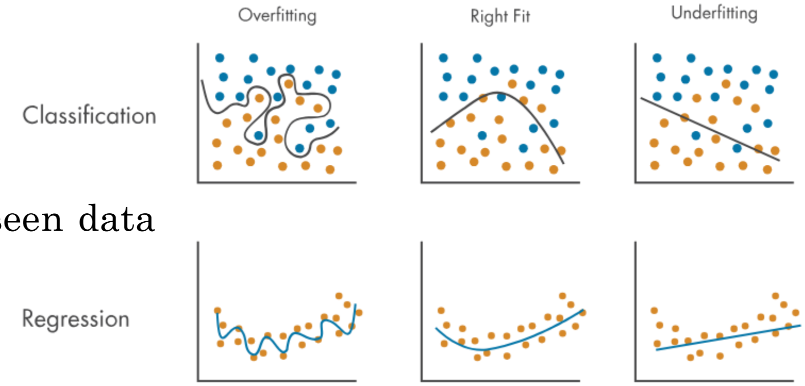
Imran Nawar

Regularization: The Problem of Overfitting

Overfitting occurs when a model performs exceptionally well on the training data but poorly on new, unseen data.

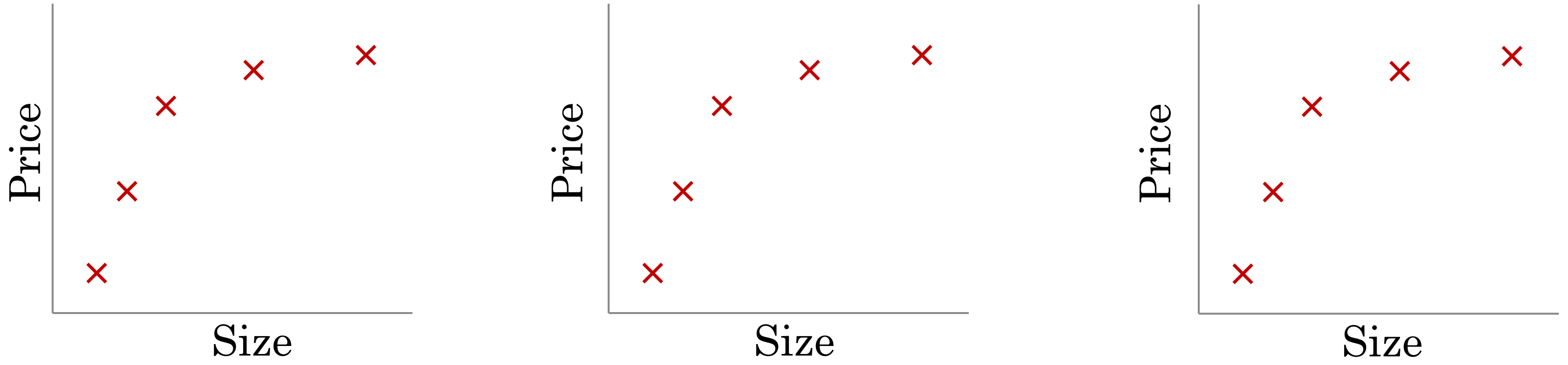
Model Fitting Scenarios:

- **Overfitting:**
  - The model learns noise and specific details of the training set, leading to poor generalization.
- **Just Right (Good Generalization):**
  - The model captures patterns well and generalizes effectively to new data.
- **Underfitting:**
  - The model does not capture patterns well, leading to poor performance on both training and validation data.



- **Goal:** Build a model that generalizes well to unseen data while avoiding overfitting.

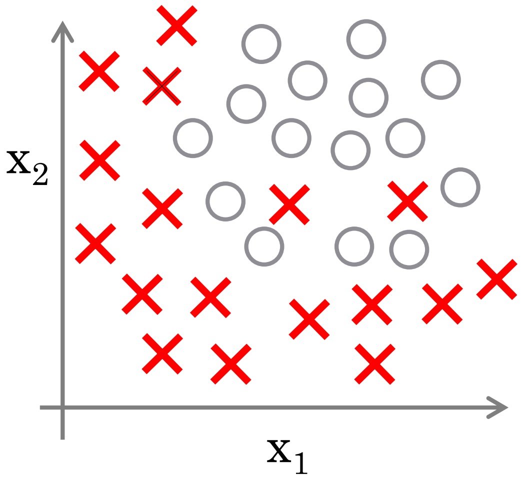
Example: Linear regression (housing prices)



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

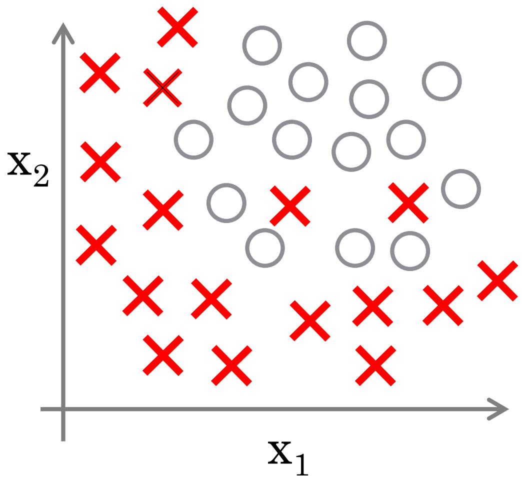
Overfitting: If we have too many features, the learned hypothesis may fit the training set very well ($J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 \approx 0$), but fail to generalize to new examples (predict prices on new examples).

Example: Logistic regression

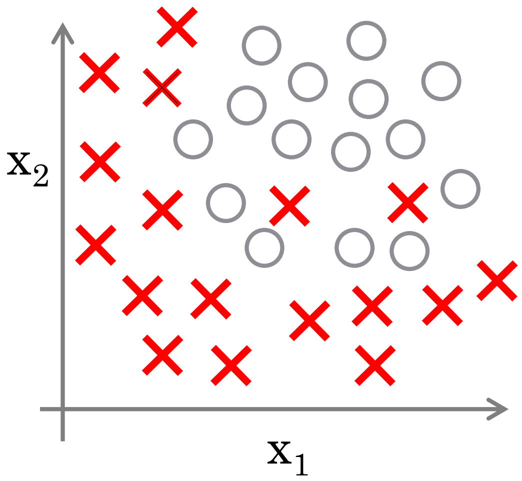


$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

(g = sigmoid function)



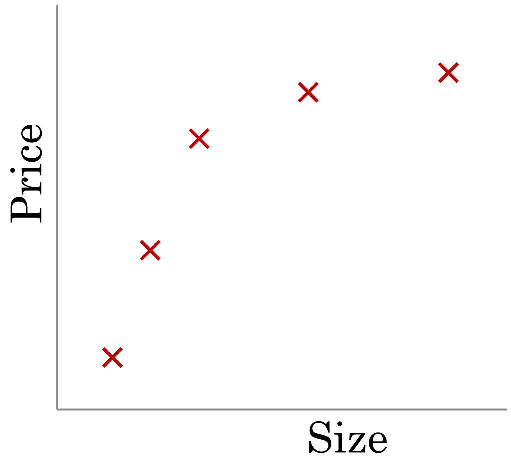
$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 \\ + \theta_3 x_1^2 + \theta_4 x_2^2 \\ + \theta_5 x_1 x_2)$$

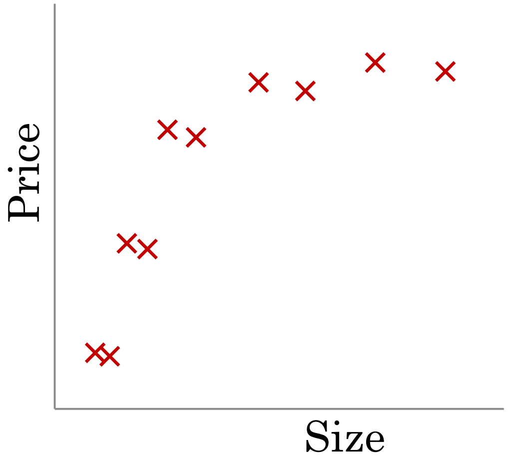


$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 \\ + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 \\ + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \dots)$$

Addressing Overfitting:

1. The number one tool to use against overfitting to get more training data.
   - **Increase Training Data:** More diverse and representative data helps the model generalize better and reduces overfitting.

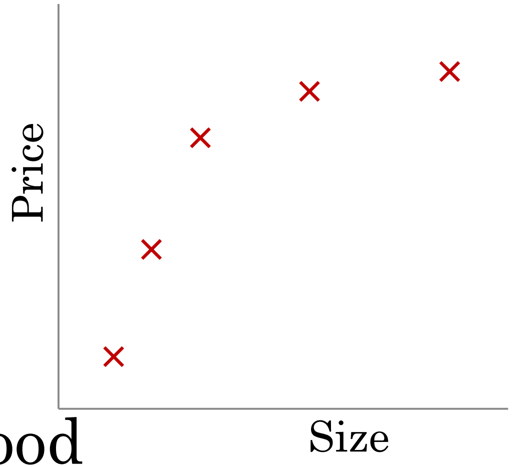




Addressing Overfitting:

2. A second option for addressing overfitting is to see if you can use fewer features.

$$
\begin{aligned}
x_1 &= \text{size of house} \\
x_2 &= \text{no. of bedrooms} \\
x_3 &= \text{no. of floors} \\
x_4 &= \text{age of house} \\
x_5 &= \text{average income in neighborhood} \\
x_6 &= \text{kitchen size} \\
&\vdots \\
x_{100}
\end{aligned}
$$



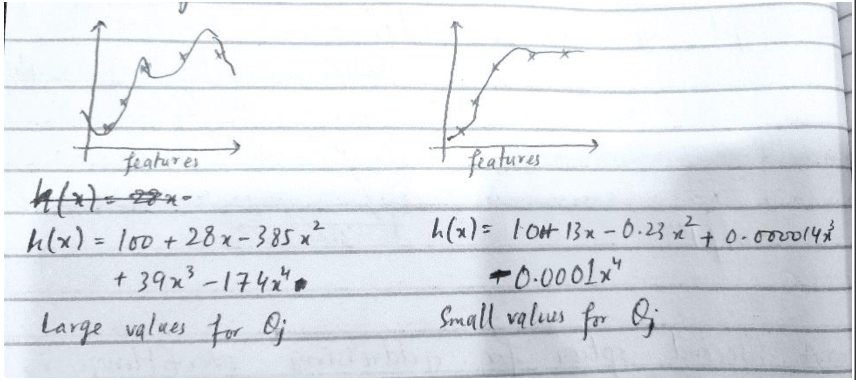
- Reduce number of features.
  - Manually select which features to keep.
  - Model selection algorithm.

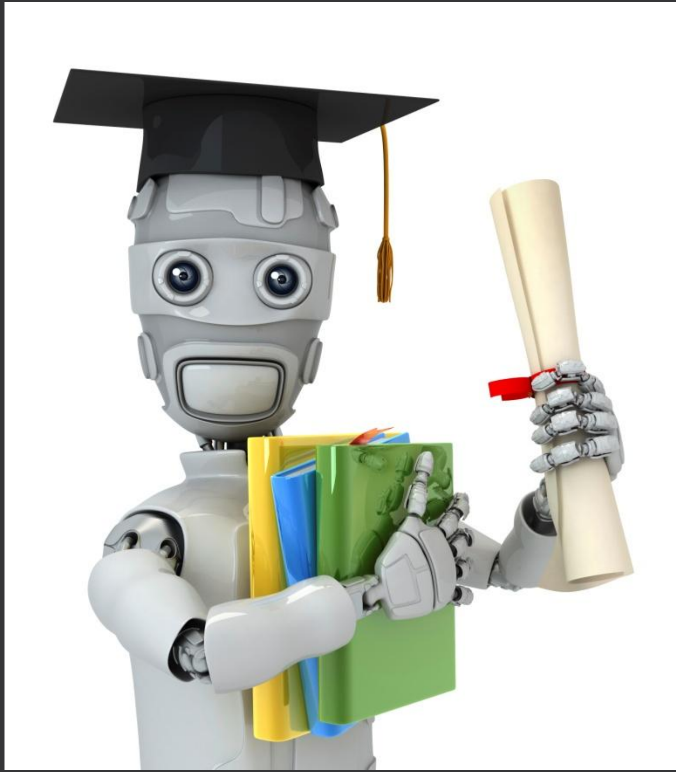
Addressing Overfitting:

3. The third option to reduce overfitting is to use regularization.

Regularization.

- Helps minimize overfitting and improves the generalization of the learning algorithm.
- Keeps all features but reduces the magnitude of model parameters θ_j (coefficients) to prevent excessive reliance on any single feature.
- Works well when there are many features, each contributing partially to predictions y.
- Provides a controlled way to reduce feature impact without completely eliminating them, unlike feature selection.



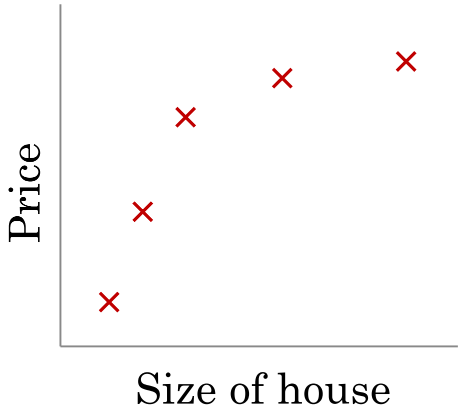


Machine Learning

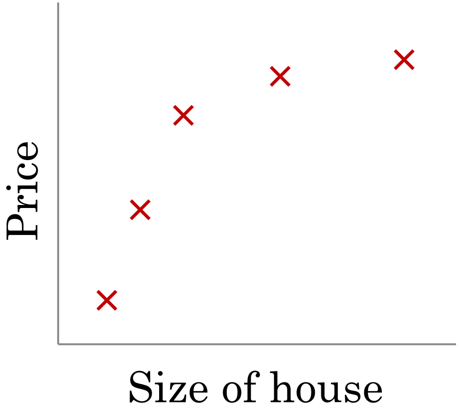
Regularization

Cost Function

Intuition



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



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

Suppose we penalize and make $\theta_3, \ \theta_4$ really small.

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

Regularization.

Small values for parameters $\theta_0, \theta_1, \ldots, \theta_n$

- "Simpler" hypothesis
- Less prone to overfitting

Housing:

- Features: $x_1, x_2, \ldots, x_{100}$
- Parameters: $\theta_0, \theta_1, \theta_2, \ldots, \theta_{100}$

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

Regularization.

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

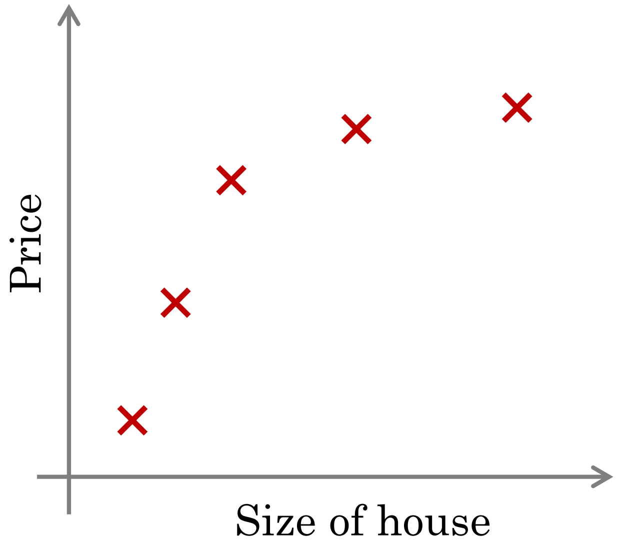
$$\min_\theta J(\theta)$$

$\lambda\sum_{j=1}^{n}\theta_j^2 \longrightarrow$ Regularization term

Add this regularization term at the end to shrink every single parameter, $\theta_1, \theta_2, \theta_3, \ldots$

$\lambda \longrightarrow$ "Lambda" Regularization parameter.

- λ is divide by 2m so that both the 1st and 2nd terms here are scaled 1/2m. By scaling both terms the same way it becomes a little bit easier to choose a good value of λ.
- By convention we are not going to penalize the parameter θ_0,



Regularization.

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

$$J(\theta) = \underbrace{\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2}_{\text{Mean squared error}} + \underbrace{\frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2}_{\text{regularization term}}$$

This new cost function trades off two goals:

- **1st:** Fitting the training data well.
- **2nd:** Keeping the parameters small.

Impact of λ in Regularization

- **If $\lambda = 0$:** it means we are not using regularization term.
- **If λ *is too big* (*enormous*):** under fit.
  - The model applies excessive regularization, making it too simple and leading to underfitting.
- **If λ is in between:** Balance "just right"

In regularized linear regression, we choose θ to minimize

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

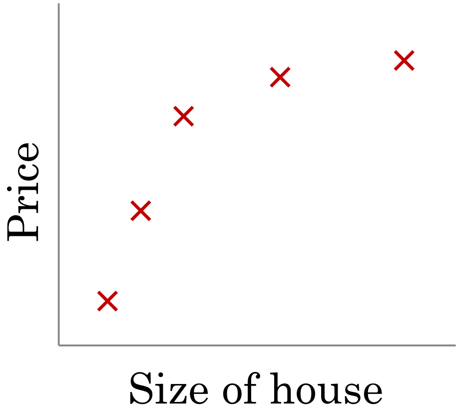
What if λ is set to an extremely large value (perhaps for too large for our problem, say $\lambda = 10^{10}$)?

- Algorithm works fine; setting λ to be very large can't hurt it.
- Algorithm fails to eliminate overfitting.
- Algorithm results in underfitting. (Fails to fit even training data well).
- Gradient descent will fail to converge.

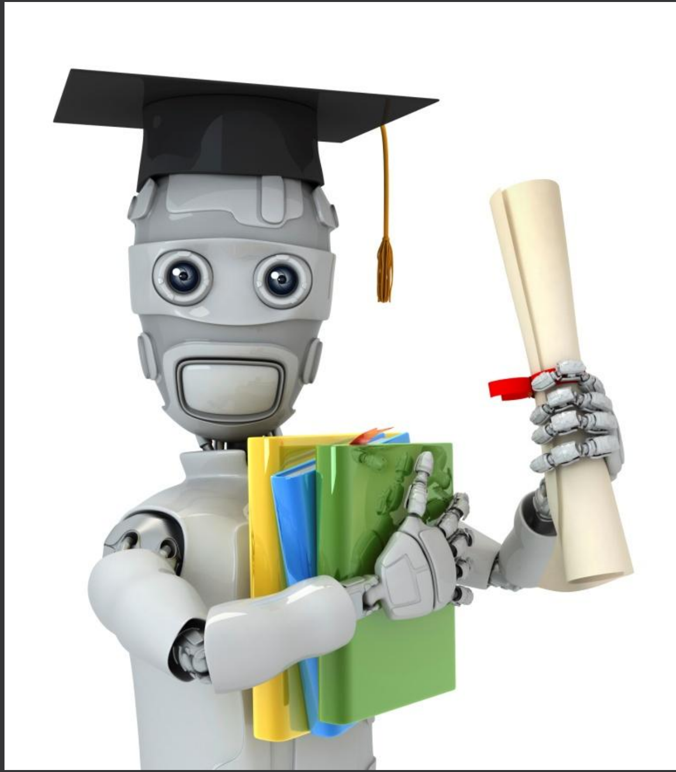
In regularized linear regression, we choose θ to minimize

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

What if λ is set to an extremely large value (perhaps for too large for our problem, say $\lambda = 10^{10}$)?



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



Machine Learning

Regularization

Regularized Linear Regression

Regularized linear regression

Cost Function: (regularized)

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

$$\min_\theta J(\theta)$$

Regularized linear regression

Gradient Descent:

repeat {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

} simultaneous update

repeat {

$$\theta_0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta)$$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

} simultaneous update $(j = 1, 2, \cdots n)$

Regularized linear regression

Gradient Descent:

$$\frac{\partial}{\partial\theta_0} J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)$$

→ don't have to penalize θ_0

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

repeat {

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

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

$j = (1, 2, 3, \cdots n)$

} simultaneous update

Regularized linear regression

Gradient Descent:

Let's take a look for update rule θ_j and rewrite it in another way.

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

→ just re arraged the term

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

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

→ number slightly less than 1 →usual gradient descent update for un-regulari Linear Regression

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

Regularized linear regression

Gradient Descent:

$\alpha \frac{\lambda}{m} = 0.01 \frac{1}{50} = 0.0002$ let's suppose

$\Rightarrow 1 - \alpha \frac{\lambda}{m} < 1$

The effect of this term is that on every single iteration of gradient descent, you are taking θ_j and multiplying it with number slightly less than ~~0~~1. (0.99) etc.

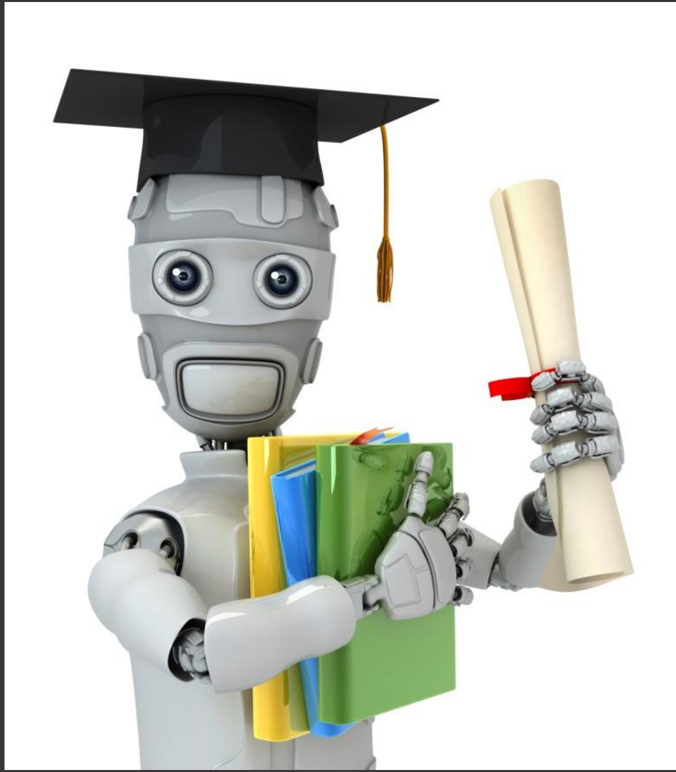
→ What regularization is doing on every single iteration is multiplying θ_j by a number slightly less than one and that has the effect of shrinking the value of θ_j a little bit.

How we get the derivative term:

$$\frac{\partial}{\partial \theta_j} J(\theta_0, \vec{\theta}) = \frac{d}{d\theta_j}\left[\frac{1}{2m}\sum_{i=1}^{m}\left(\underbrace{h_\theta(\vec{x}^{(i)})}_{\theta_0 + \vec{\theta}\cdot\vec{x}} - y^{(i)}\right)^2 + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2\right]$$

$$= \frac{1}{2m}\sum_{i=1}^{m}\left[\left(\theta_0 + \vec{\theta}\cdot\vec{x} - y^{(i)}\right) 2 x_j^{(i)}\right] + \frac{\lambda}{2m} 2\theta_j \qquad \text{No } \sum_{j=1}^{n}$$

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

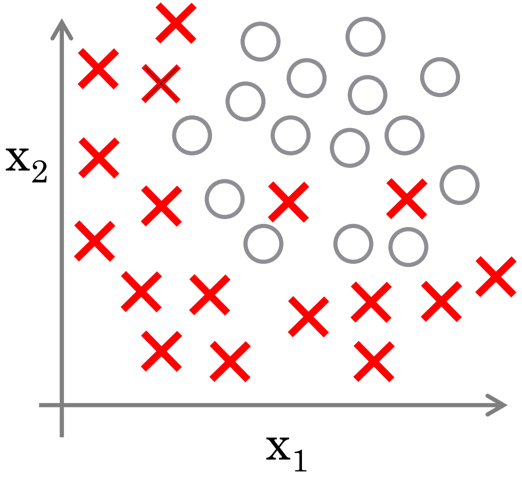


Machine Learning

Regularization

Regularized Logistic Regression

Regularized Logistic Regression.



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \dots)$$

Cost function:

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

Gradient Descent

Repeat $\{$

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

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

$$(j = \cancel{0}, 1, 2, 3, \ldots, n)$$

$\}$

Andrew Ng

Thank You