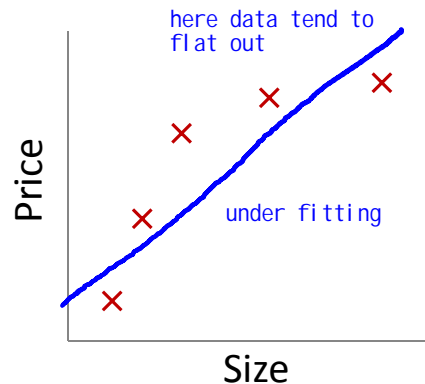


Machine Learning

Regularization

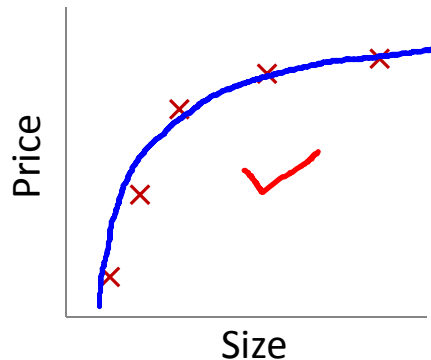
The problem of
overfitting

Example: Linear regression (housing prices)



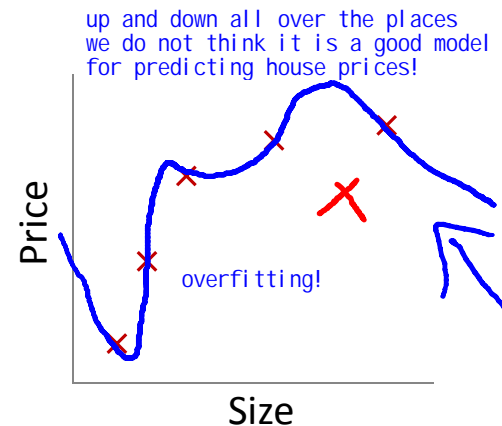
$$\rightarrow \theta_0 + \theta_1 x$$

"Underfit" "High bias"



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

"Just right"



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

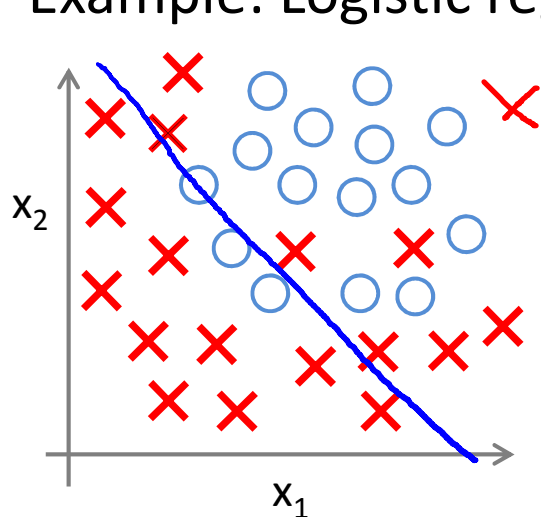
"Overfit" "High variance"

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).

 predict!

overfitting examples in logistic regression!

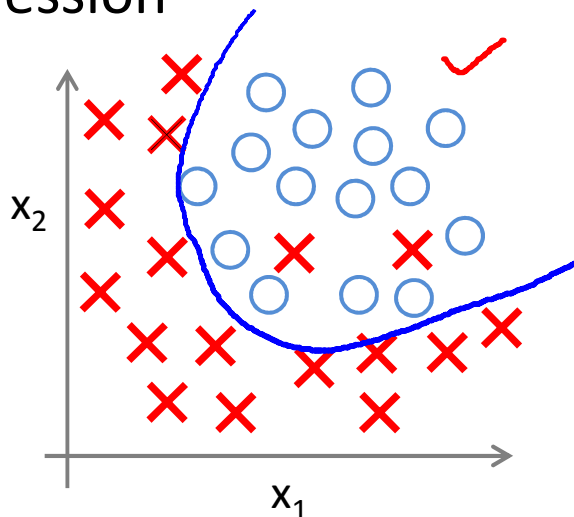
Example: Logistic regression



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

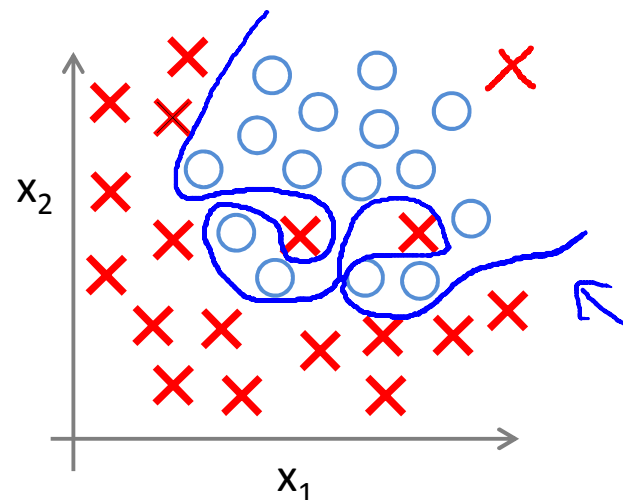
(g = sigmoid function)

"Underfit"



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

this is just right



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

"Overfit"

hypothesis has high variance!

Addressing overfitting: what do we do when overfitting happens

x_1 = size of house

x_2 = no. of bedrooms

x_3 = no. of floors

x_4 = age of house

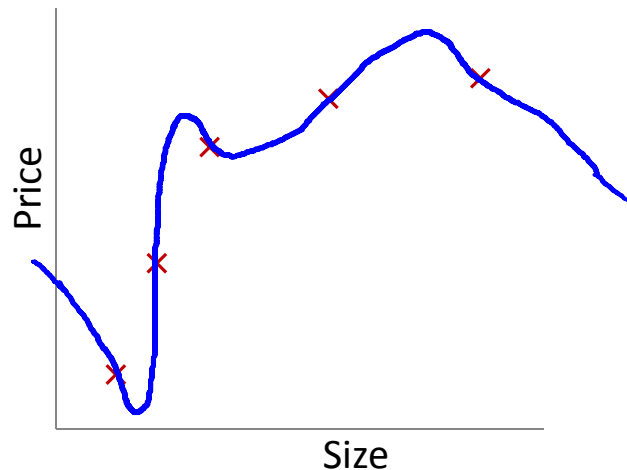
x_5 = average income in neighborhood

x_6 = kitchen size

\vdots

x_{100}

However, when we have so many features, it becomes harder to plot the data and visualize them!



one way to address the overfitting is to plot the hypothesis function and look at its shape!

Addressing overfitting: two main options to address overfitting problem

Options:

1. Reduce number of features.

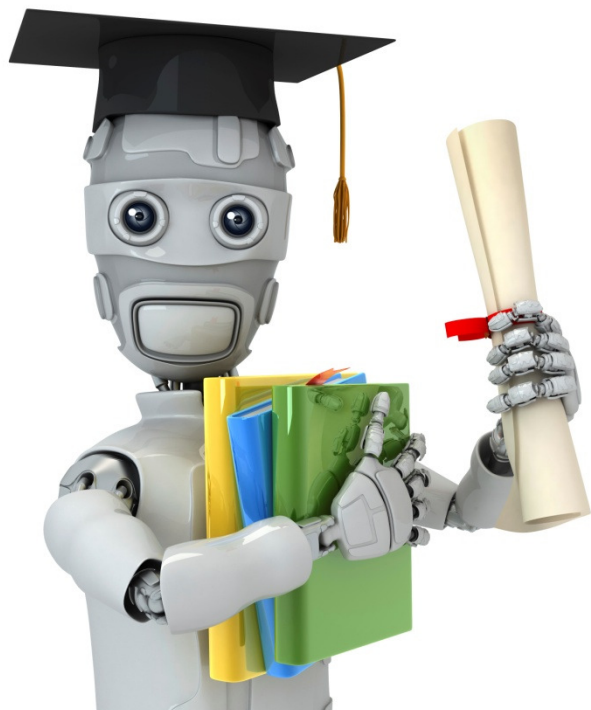
→ — Manually select which features to keep.

→ — Model selection algorithm (later in course).

2. Regularization. 正则化

→ — Keep all the features, but reduce magnitude/values of parameters θ_j .

— Works well when we have a lot of features, each of which contributes a bit to predicting y .



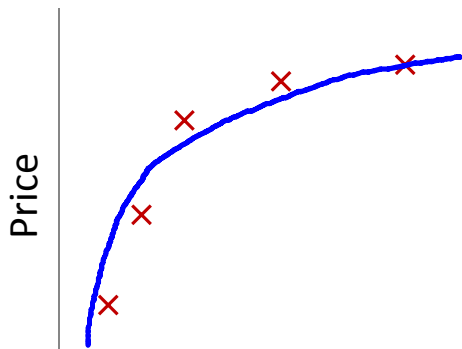
Machine Learning

for dealing with overfitting!

Regularization

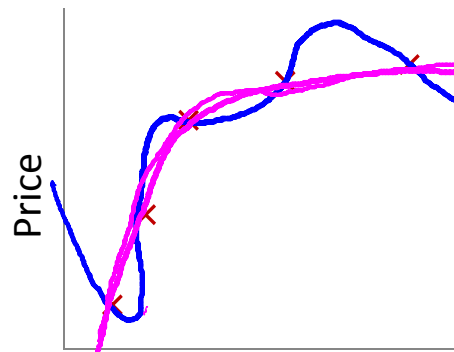
Cost function

Intuition



Size of house

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



Size of house

a simpler hypothesis
with small theta 3/4

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

Suppose we **penalize** and make θ_3, θ_4 really small.

$$\min_{\theta} \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + 1000 \theta_3^2 + 1000 \theta_4^2$$

we minimize this new cost function
so that we can make theta 3,4 very small

$$\theta_3 \approx 0$$

$$\theta_4 \approx 0$$

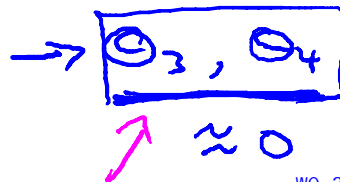
Regularization.

main ideas behind regularization!

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

- “Simpler” hypothesis
- Less prone to overfitting

e.g. we penalize these two parameters to make them very small!



an example!

Housing:

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

we do not know which to pick to make small parameters

we add extra regularization term to shrink all the parameters

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

Diagram illustrating the regularization term in the cost function $J(\theta)$. The term $\lambda \sum_{j=1}^n \theta_j^2$ is highlighted in a red box. A red arrow points from the text 'we add extra regularization term to shrink all the parameters' to this box. Below the box, the parameters $\theta_1, \theta_2, \theta_3, \dots, \theta_{100}$ are listed, with θ_0 crossed out and underlined. A red arrow points from the text 'by convention we do not regularize theta_0' to θ_0 . Another red arrow points from the box to the list of parameters.

by convention we do not regularize theta_0

Regularization.

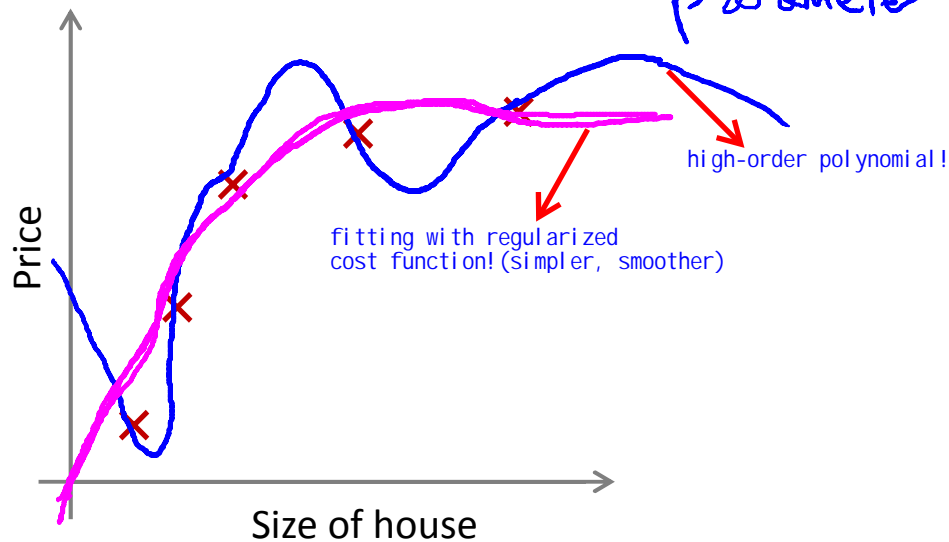
regularized cost function

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

controls a tradeoff b/c 2 diff goals:
1. fitting the training data well
2. want to keep parameter small, therefore keep the hypothesis simple to avoid overfitting!

$$\min_{\theta} J(\theta)$$





regularization parameter



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 far 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 far too large for our problem, say $\lambda = 10^{10}$)?

then we will have

$$\theta_1, \theta_2, \theta_3, \theta_4$$

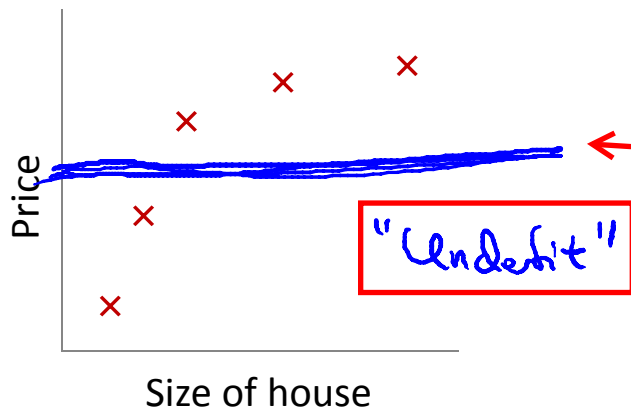
$$\theta_1 \approx 0, \theta_2 \approx 0$$

$$\theta_3 \approx 0, \theta_4 \approx 0$$

as a result

$$h_{\theta}(x) \approx \theta_0$$

almost flat line!



$h_{\theta}(x)$

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



Machine Learning

Regularization

Regularized linear
regression

Regularized linear regression

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

$$\min_{\theta} \underline{J(\theta)}$$

Gradient descent

$$\theta_0$$

$$\theta_1, \theta_2, \dots, \theta_n$$

Repeat {

$$\frac{\partial}{\partial \theta_0} J(\theta)$$

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

$$\theta_j := \theta_j - \alpha$$

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

$$+ \frac{\lambda}{m} \theta_j$$

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

}

$$\rightarrow J(\theta)$$

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

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

$$0.99$$

$$\theta_j \times 0.99$$

$$\theta_j^2$$

Normal equation

$$\underline{X} = \begin{bmatrix} (x^{(1)})^T \\ \vdots \\ (x^{(m)})^T \end{bmatrix} \leftarrow$$

$m \times (n+1)$

$$\underset{\uparrow}{y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix} \quad \mathbb{R}^m$$

$$\rightarrow \min_{\theta} \underline{J(\theta)}$$

$$\Rightarrow \Theta = \left(X^T X + \lambda \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{(n+1) \times (n+1)} \right)^{-1} X^T y$$

$\frac{\partial}{\partial \theta_j} J(\theta) \stackrel{\text{set}}{=} 0$

$\in \mathbb{R}^n \quad n=2$

Non-invertibility (optional/advanced).

Suppose $m \leq n$, \leftarrow
(#examples) (#features)

$$\theta = \underbrace{(X^T X)^{-1}}_{\text{non-invertible / singular}} X^T y$$

pinv

inv
 \nearrow

If $\lambda > 0$,

$$\theta = \left(X^T X + \lambda \begin{bmatrix} 0 & & & \\ & 1 & & \\ & & 1 & \\ & & & \ddots \\ & & & & 1 \end{bmatrix} \right)^{-1} X^T y$$

invertible.



Machine Learning

Regularization

Regularized
logistic regression

Regularized logistic regression.



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

Handwritten blue arrows point to $\theta_1 x_1$ and $\theta_2 x_1^2$. A handwritten pink arrow points to the $\theta_5 x_1^2 x_2^3$ term.

Cost function:

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

The term $\theta_1, \theta_2, \dots, \theta_n$ is enclosed in a pink box.

Gradient descent

Repeat {

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

$$\rightarrow \theta_j := \theta_j - \alpha \left[\underbrace{\frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}}_{\substack{(j = \cancel{0}, 1, 2, 3, \dots, n) \\ \theta_1, \dots, \theta_n}} + \frac{1}{n} \theta_j \right] \leftarrow$$

}

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

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

Advanced optimization

→ `function [jVal, gradient] = costFunction(theta)`

`jVal = [code to compute $J(\theta)$];`

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

→ `gradient(1) = [code to compute $\frac{\partial}{\partial \theta_0} J(\theta)$];`

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

→ `gradient(2) = [code to compute $\frac{\partial}{\partial \theta_1} J(\theta)$];`

$$\left(\frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_1^{(i)} \right) - \frac{\lambda}{m} \theta_1 \leftarrow$$

→ `gradient(3) = [code to compute $\frac{\partial}{\partial \theta_2} J(\theta)$];`

$$\vdots \left(\frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_2^{(i)} \right) - \frac{\lambda}{m} \theta_2$$

`gradient(n+1) = [code to compute $\frac{\partial}{\partial \theta_n} J(\theta)$];`

$f_{\text{minunc}}(\theta \text{ costFunction})$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

$\theta_0 \leftarrow \text{theta}(1)$
 $\theta_1 \leftarrow \text{theta}(2)$
 $\theta_n \leftarrow \text{theta}(n+1)$

$J(\theta)$

