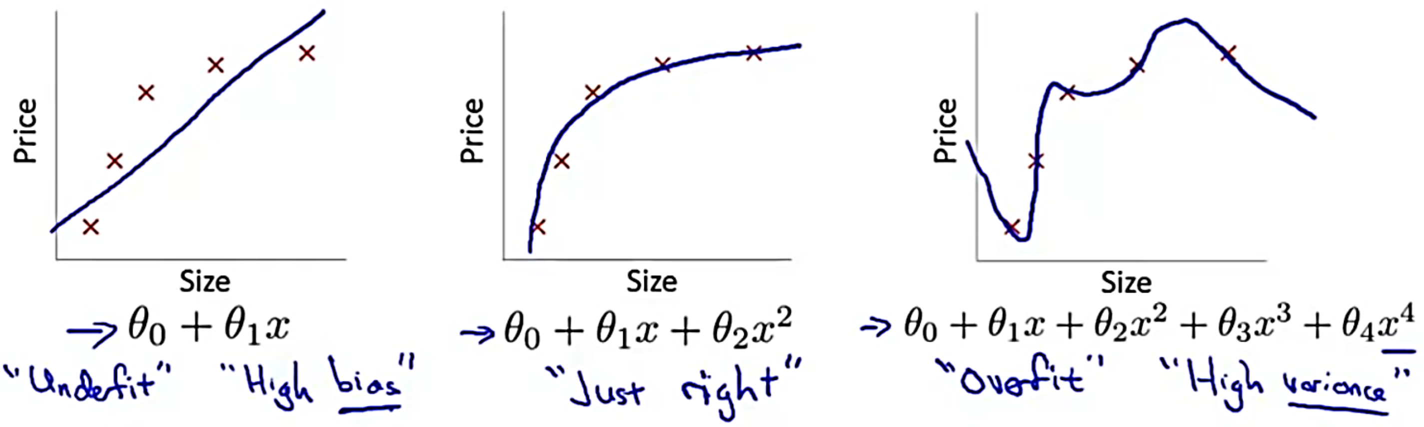
7. Regularization

THE PROBLEM OF OVERFITTING:

Example: Linear regression (housing prices)

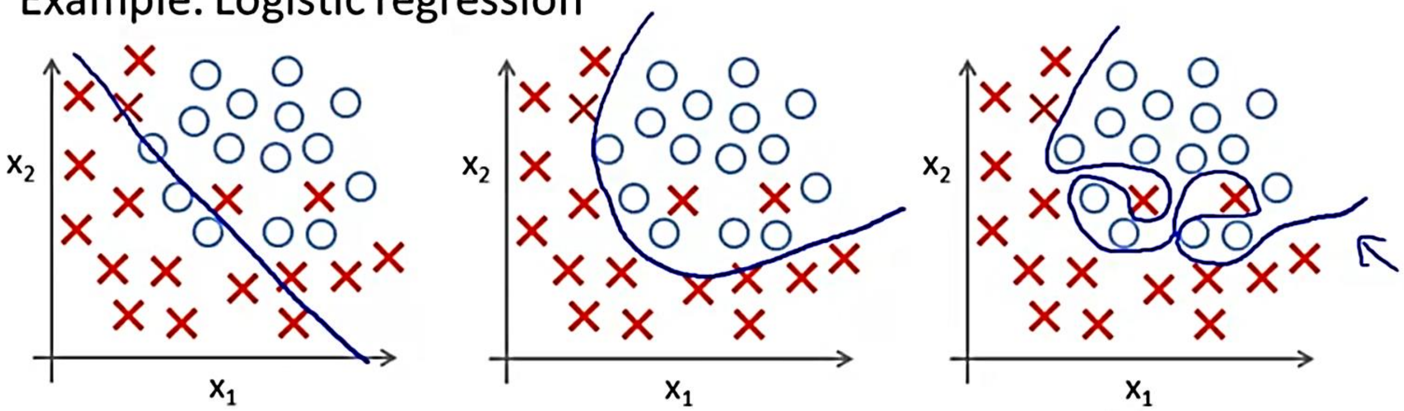


Overfitting: If we have too many features, the learned hypothesis may fit the training set very well ($J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 \approx 0$), but fail to generalize to new examples (predict prices on new examples).

Naively, it might seem that the more features we add, the better. However, there is also a danger in adding too many features: The rightmost figure is the result of 5th order polynomial $\mathbf{y = \Sigma^5_{j=0}\, \theta_j x^j}$

Overfitting or **high variance**, is caused by a hypothesis function that fits the available data but does not generalize well to predict new data. It is usually caused by a complicated function that creates a lot of unnecessary curves and angles unrelated to the data.

Example: Logistic regression



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

(g = sigmoid function)

"Underfit"

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

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

"Overfit"

Addressing overfitting:

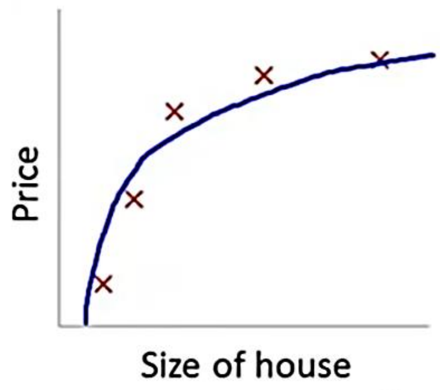
Options:

1. Reduce number of features.
   - Manually select which features to keep.
   - Model selection algorithm (later in course).
2. Regularization.
   - 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.

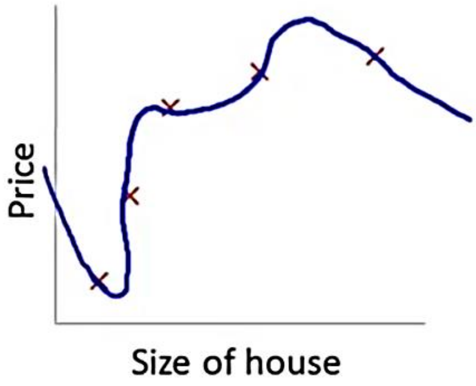
Model selection algos decide automatically which features to use and which don't, but they may also lead to deleting of important features

Cost function:

Intuition



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



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

Our old error function:

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

Suppose we penalize and make θ_3, θ_4 really small.

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

$$\theta_3 \approx 0 \qquad \theta_4 \approx 0$$

Therefore:

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

If we have overfitting from our hypothesis function, we can reduce the weight that some of the terms in our function carry, by increasing their cost.

Regularization.

Small values for parameters $\boxed{\theta_0, \theta_1, \ldots, \theta_n}$

- "Simpler" hypothesis
- Less prone to overfitting

Example:

Housing:

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

Modified error fxn:

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

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

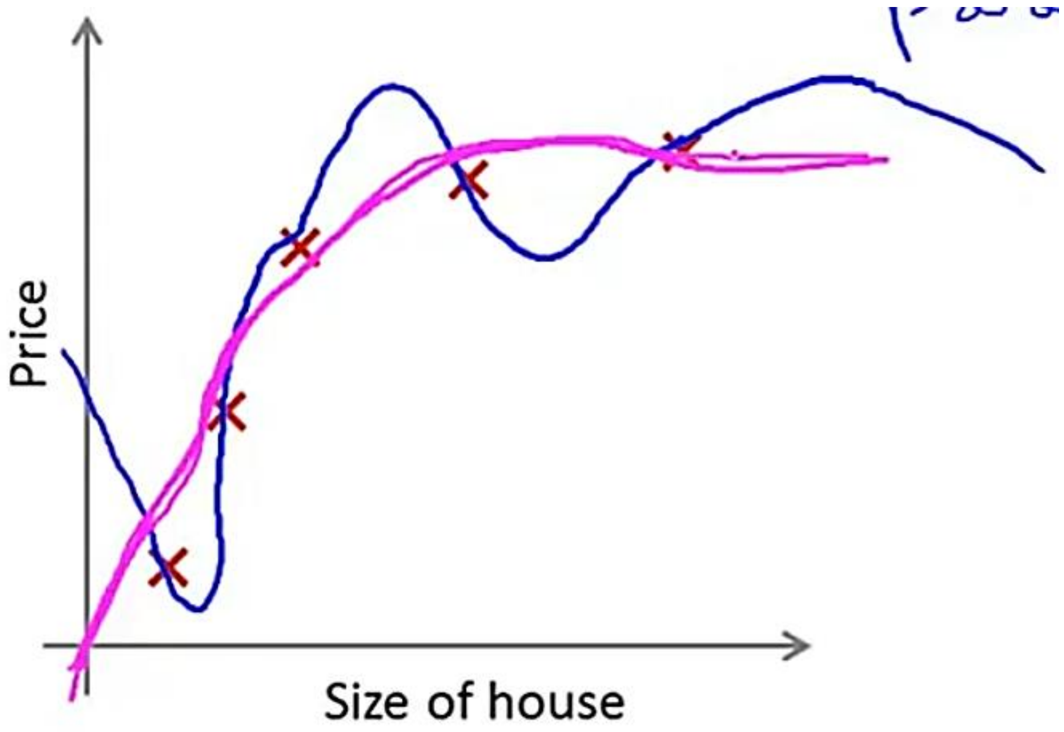
Here summation starts at 1: ***Θ$_0$ is not penalized***.

The **λ**, or lambda, is the **regularization parameter**. It determines how much the costs of our theta parameters are **inflated**

After minimizing J(Θ) and obtaining values of optimized Θs:

Blue = **without** regularization parameters

Pink = **with** regularized parameters term



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 $\lambda = 10^{10}$)?

Then:

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

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

$\underline{h_\theta(x) = \theta_0}$

The regularization parameter should be chosen carefully

MINIMIZINIG J(Θ): to find the optimal values of Θ

We can apply regularization to both linear regression and logistic regression.

For linear regression:

Regularized linear regression

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

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

<u>There are 2 methods for using regularization in LINEAR REGRESSION:</u>

- Gradient descent
- Normal equations

1.Gradient Descent

We do not want to penalize so we write it separately

Gradient descent

$\underline{\Theta_0}$ $\Theta_1, \Theta_2, \ldots, \Theta_n$

Repeat {

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

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

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

}

Here:

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

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

Intuitively you can see it as reducing the value of Θ_j by some amount on every update

2.Normal Equations

Normal equation

$$X = \begin{bmatrix} (x^{(1)})^T \\ \vdots \\ (x^{(m)})^T \end{bmatrix} \qquad y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix} \quad \mathbb{R}^m$$

$m \times (n+1)$

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

Find Θ from this:

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

It turns out to be:

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

Here this blue matrix is **(n+1) x (n+1)** matrix with all **diagonal elements =1** ***except first, which is 0.*** . and all others are also 0

This matrix is denoted by **L.**

Intuitively, this is the identity matrix (though we are not Including x_0), multiplied with a single real number λ.

- There is **no issue of non-invertibility** with **normal equations** in regularization:

Non-invertibility (optional/advanced).

Suppose $m \leq n, \Leftarrow$

(#examples) (#features)

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

If **m<< n** : $X^T X$ is non invertible..

But in regularization:

If $\underline{\lambda > 0}$,

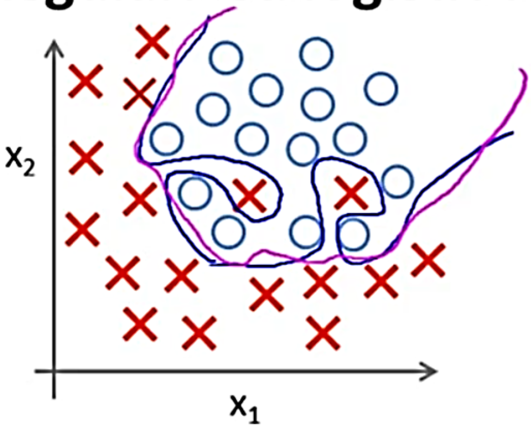
$$\theta = \underbrace{\left(X^T X + \lambda \begin{bmatrix} 0 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix} \right)^{-1}}_{\text{invertible.}} X^T y$$

This matrix is invertible. So, no problem of non-invertibility

Recall that if m < n, then $X^T X$ is non-invertible. However, when we add the term λ·L, then $X^T X$ + λ·L becomes invertible.

For logistic regression:

Regularized logistic regression.



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

Old cost error fxn:

Cost function:

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

New cost error fxn:

Cost function:

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

$\boxed{\theta_1, \theta_2, \dots, \theta_n}$

Gradient descent

Repeat $\{$

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

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

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

$$\theta_1 \ldots \theta_n$$

$\}$

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

Implementation of cost fxn: which can then be passed to **fminunc** fxn to minimize the cost and find respective Θ:

fminunc (@costFunction...)

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \begin{matrix} \text{theta(1)} \\ \text{theta(2)} \\ \\ \text{theta(n+1)} \end{matrix}$$

Advanced optimization

```
function [jVal, gradient] = costFunction(theta)
    jVal = [ code to compute J(θ)];
```

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

```
    gradient(1) = [code to compute ∂/∂θ0 J(θ)];
```

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

```
    gradient(2) = [code to compute ∂/∂θ1 J(θ)];
```

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

```
    gradient(3) = [code to compute ∂/∂θ2 J(θ)];
```

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

```
        ⋮
    gradient(n+1) = [code to compute ∂/∂θn J(θ)];
```