Machine Learning

Regularization
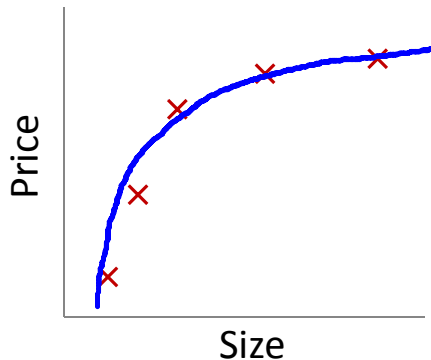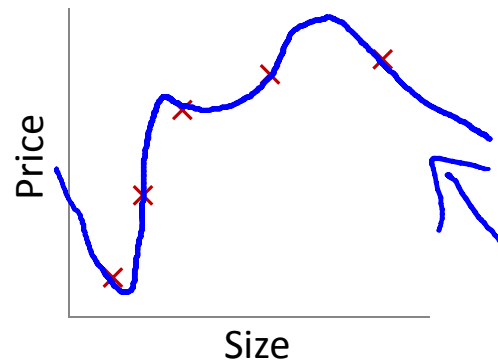
The problem of overfitting

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).
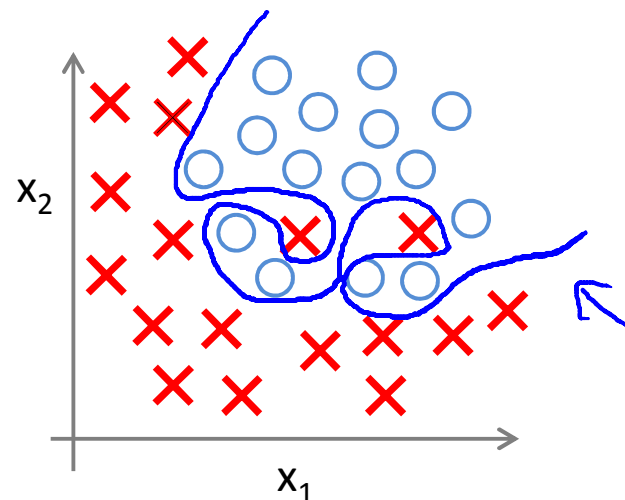
Example: Logistic regression

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

(g = sigmoid function)

"Underfit"

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

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

"Overfit"

Andrew Ng

Addressing overfitting:

$x_1 =$ size of house

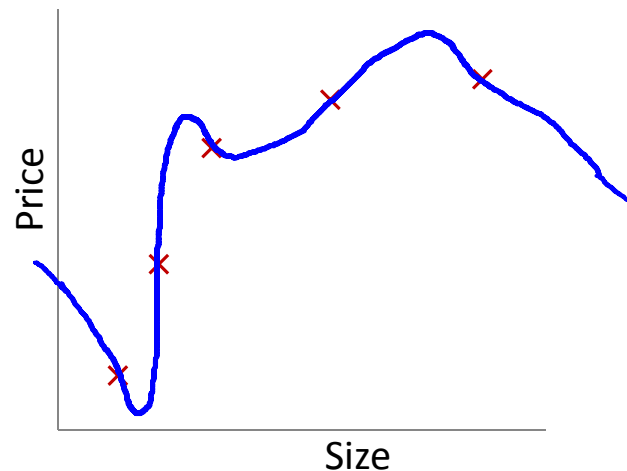$x_2 =$ no. of bedrooms

$x_3 =$ no. of floors

$x_4 =$ age of house

$x_5 =$ average income in neighborhood

$x_6 =$ kitchen size

$\vdots$

x_{100}

Addressing overfitting:

Options:

    1. <mark>Reduce number of features</mark>.

       — Manually select which features to keep.

       — Model selection algorithm (later in course).

    2. <mark>Regularization</mark>.

       — Keep all the features, but <mark>reduce</mark> magnitude/values of <mark>parameters θ_j</mark>.

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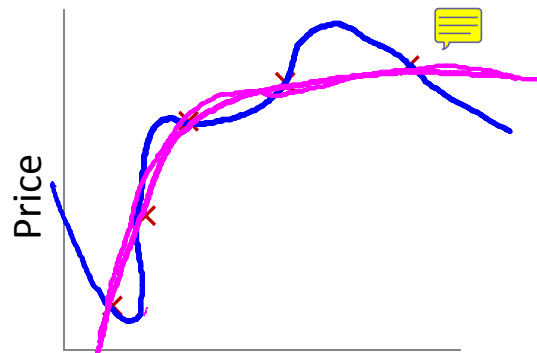
Regularization

Cost function

Machine Learning

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 θ_3, θ_4 really small.

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

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

Andrew Ng

Regularization.

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

— "Simpler" hypothesis
— Less prone to overfitting
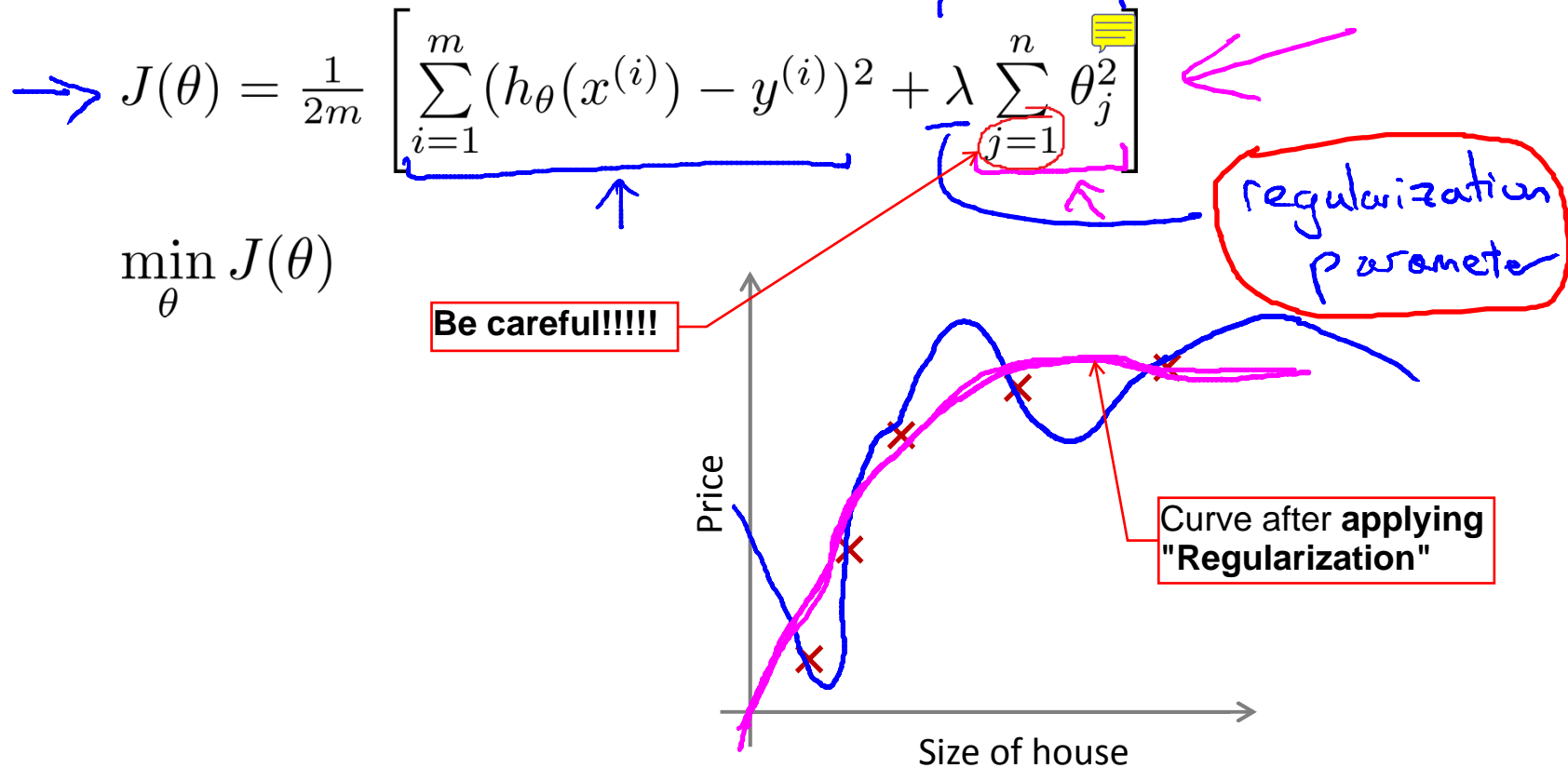
$\theta_3, \theta_4 \approx 0$

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

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

$\theta_1, \theta_2, \theta_3, \ldots, \theta_{100}$

Andrew Ng

Regularization.

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

regularization parameter

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

Be careful!!!!!

Curve after **applying "Regularization"**

Price

Size of house

Andrew Ng

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
- Algortihm fails to eliminate overfitting.
- Algorithm results in underfitting. (Fails to fit even training data well).
- Gradient descent will fail to converge.

Summary, **Do NOT** set **"alpha"** to be *too large* value

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

Price

"Underfit"

Size of house

$h_\theta(x)$

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

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

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

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

$h_\theta(x) = \theta_0$

Regularization

Regularized linear regression

Machine Learning

Regularized linear regression

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

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

Gradient descent

Θ_0 $\Theta_1, \Theta_2, \ldots, \Theta_n$

$\frac{\partial}{\partial \Theta_0} J(\Theta)$

Repeat {

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

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

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

}

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

$\rightarrow J(\Theta)$

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

0.99

$\Theta_j \times 0.99$

Θ_j

Andrew Ng

Normal equation

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

$$m \times (n+1)$$

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

$$?\,?$$

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

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

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

$$(n+1) \times (n+1)$$

$$\text{E.g.} \quad n = 2 \quad \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Non-invertibility (optional/advanced).

Suppose $m \leq n$, ⬅

   (#examples) (#features)

$$\theta = \underline{(X^T X)^{-1}} X^T y$$

non-invertible / singular

pinv   inv

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

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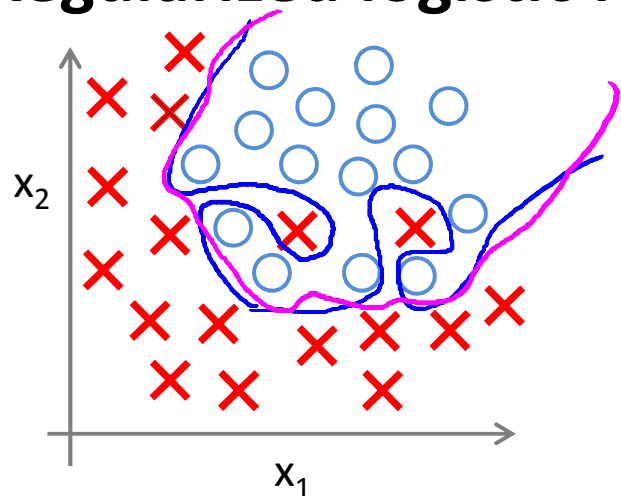
invertible.

Regularization

Regularized logistic regression

Machine Learning

Regularized logistic regression.

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

For **Logistic regression**

Cost function:

$$J(\theta) = -\left[\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))\right]$$
$$+ \frac{\lambda}{2m}\sum_{j=1}^{n} \theta_j^2 \qquad \boxed{\theta_1, \theta_2, \dots, \theta_n}$$

Andrew Ng

Gradient descent

Repeat {

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

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

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

$$\theta_1 \ldots \theta_n$$

}

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

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

Andrew Ng

Advanced optimization

fminunc (@ costFunction)

$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$ — theta(1) ← theta(2)

theta(n+1)

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

    jVal = [ code to compute J(θ)];
```

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

```
    gradient(1) = [code to compute ∂/∂θ₀ J(θ)];
```
$$\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)} \leftarrow$$

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

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

```
    gradient(3) = [code to compute ∂/∂θ₂ J(θ)];
```
$$\vdots \quad \left(\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)}\right) - \frac{\lambda}{m} \theta_2$$

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

Andrew Ng