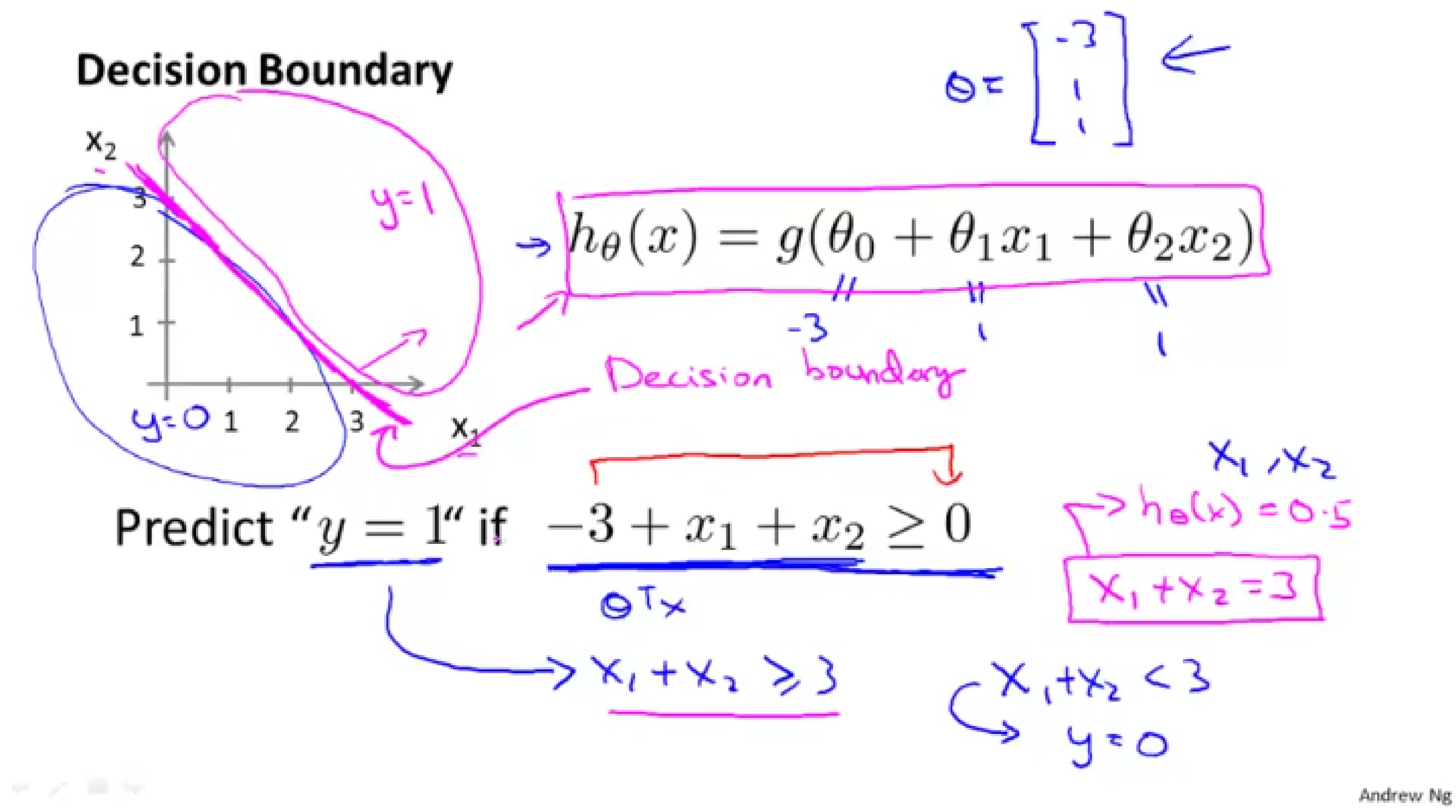


Decision Boundary

$\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$

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

$\theta_0 = -3$, $\theta_1 = 1$, $\theta_2 = 1$

Decision boundary

Predict "$y = 1$" if $\underbrace{-3 + x_1 + x_2 \geq 0}_{\theta^T x}$

$\rightarrow x_1 + x_2 \geq 3$

x_1, x_2

$\rightarrow h_\theta(x) = 0.5$

$\boxed{x_1 + x_2 = 3}$

$x_1 + x_2 < 3 \rightarrow y = 0$

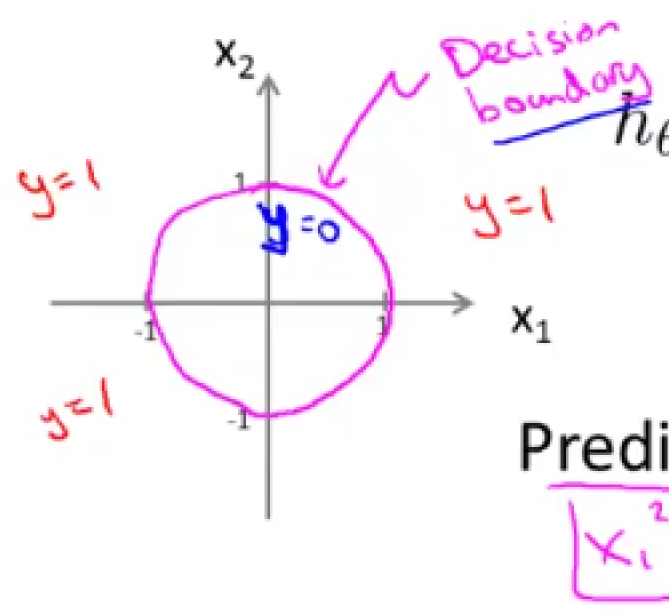
visualization.

But even if we take away the data set this decision boundary and the region where we predict y =1 versus y = 0, that's a property of the hypothesis and of the parameters of the hypothesis and not a property of the data set.

Later on, of course, we'll talk about how to fit the parameters and there we'll end up using the training set, using our data. To determine the value of the parameters.

But once we have particular values for the parameters theta0, theta1,

Non-linear decision boundaries



$$h_\theta(x) = g(\overset{=-1}{\theta_0} + \overset{=0}{\theta_1} x_1 + \overset{=0}{\theta_2} x_2 + \underset{=1}{\theta_3} x_1^2 + \underset{=1}{\theta_4} x_2^2) \qquad \theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Predict "$y = 1$" if $\underline{-1 + x_1^2 + x_2^2 \geq 0}$

$\boxed{x_1^2 + x_2^2 = 1}$ $\qquad x_1^2 + x_2^2 \geq 1$

Andrew Ng

Once again, the decision boundary is a property, not of the trading set, but of the hypothesis under the parameters. So, so long as we're given my parameter vector theta, that defines the decision boundary, which is the circle. But the training set is not what we use to define the decision boundary.

The training set may be used to fit the parameters theta. We'll talk about how to do that later. But, once you have the parameters theta, that is what defines the decisions boundary. Let me put back the training set just for visualization. And finally let's look at a more complex example.

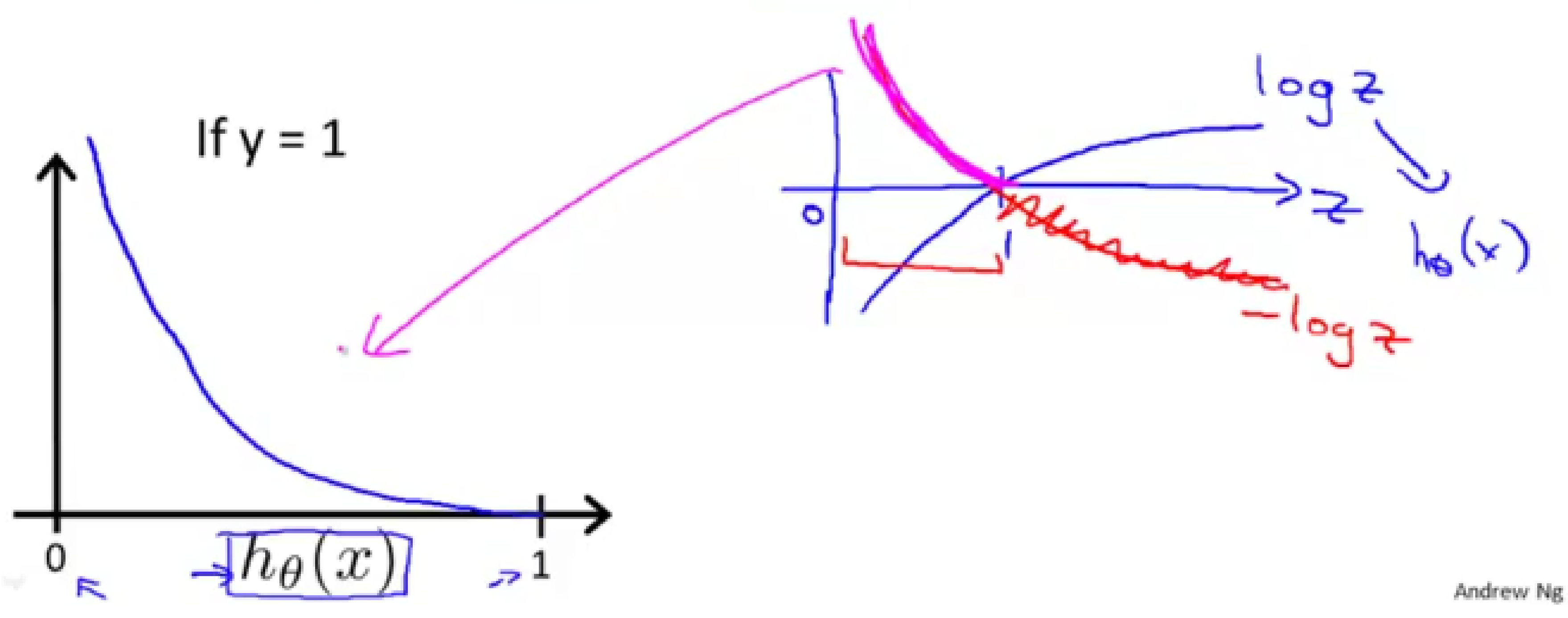
3:31

Cost Function

You're using Coursera offline

Logistic regression cost function

$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} \boxed{-\log(h_\theta(x))} & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$



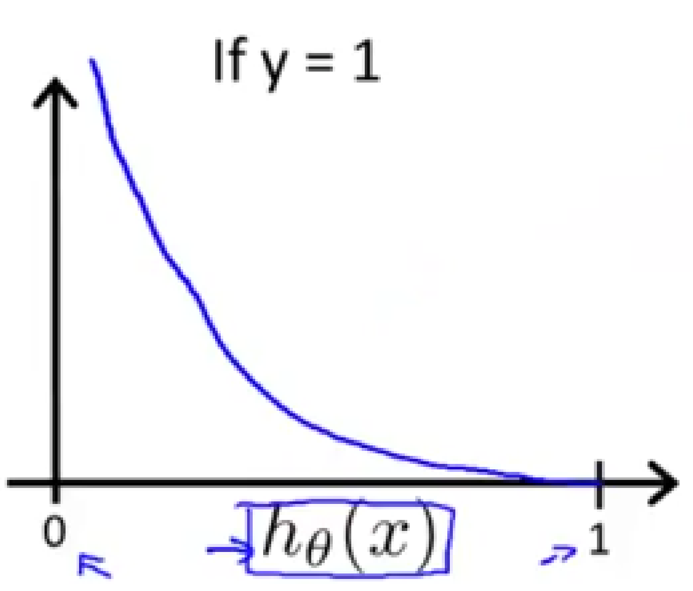
Now, this cost function has a few interesting and desirable properties. First, you notice that if y is equal to 1 and h(x) is equal to 1, in other words, if the hypothesis exactly predicts h equals 1 and y is exactly equal to what it predicted, then the cost = 0 right?

That corresponds to the curve doesn't actually flatten out. The curve is still going. First, notice that if h(x) = 1, if that hypothesis predicts that y = 1 and if indeed y = 1 then the cost = 0. That corresponds to this point down here, right?

If h(x) = 1 and we're only considering

Logistic regression cost function

$$\text{Cost}(h_\theta(x), y) = \begin{cases} \boxed{-\log(h_\theta(x))} & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$



$\text{Cost} = 0$ if $y = 1, h_\theta(x) = 1$
But as $h_\theta(x) \to 0$
$Cost \to \infty$

Captures intuition that if $h_\theta(x) = 0$, (predict $P(y = 1|x;\theta) = 0$), but $y = 1$, we'll penalize learning algorithm by a very large cost.

Now, this cost function has a few interesting and desirable properties. First, you notice that if y is equal to 1 and h(x) is equal to 1, in other words, if the hypothesis exactly predicts h equals 1 and y is exactly equal to what it predicted, then the cost = 0 right?

That corresponds to the curve doesn't actually flatten out. The curve is still going. First, notice that if h(x) = 1, if that hypothesis predicts that y = 1 and if indeed y = 1 then the cost = 0. That corresponds to this point down here, right?

If h(x) = 1 and we're only considering

Optimization algorithm

Given θ, we have code that can compute

$$\boxed{\begin{array}{l} - \ J(\theta) \\ - \ \frac{\partial}{\partial \theta_j} J(\theta) \end{array}} \begin{array}{l} \leftarrow \\ \leftarrow \quad (\text{for } j = 0, 1, \ldots, n) \end{array}$$

Optimization algorithms:

$\rightarrow$ - Gradient descent
- Conjugate gradient
- BFGS
- L-BFGS

Advantages:

- No need to manually pick α
- Often faster than gradient descent.

Disadvantages:

- More complex

of this algorithms you usually do

Andrew Ng

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

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} \quad \theta_1 = 5, \theta_2 = 5.$$

$$\boxed{J(\theta) = (\theta_1 - 5)^2 + (\theta_2 - 5)^2}$$

$$\frac{\partial}{\partial \theta_1} J(\theta) = 2(\theta_1 - 5)$$

$$\frac{\partial}{\partial \theta_2} J(\theta) = 2(\theta_2 - 5)$$

```
function [jVal, gradient]
           = costFunction(theta)
  jVal = (theta(1)-5)^2 + ...
         (theta(2)-5)^2;
  gradient = zeros(2,1);
  gradient(1) = 2*(theta(1)-5);
  gradient(2) = 2*(theta(2)-5);
```

```
options = optimset('GradObj', 'on', 'MaxIter', '100');
initialTheta = zeros(2,1);
[optTheta, functionVal, exitFlag] ...
     = fminunc(@costFunction, initialTheta, options);
```

-and the way you call this is as follows.

Andrew Ng

theta = $\begin{array}{l} \left[\begin{array}{c} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{array}\right] \end{array} \begin{array}{l} \text{— theta(1)} \\ \text{— theta(2)} \\ \\ \text{— theta(n+1)} \end{array}$

function [jVal, gradient] = costFunction(theta)

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

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

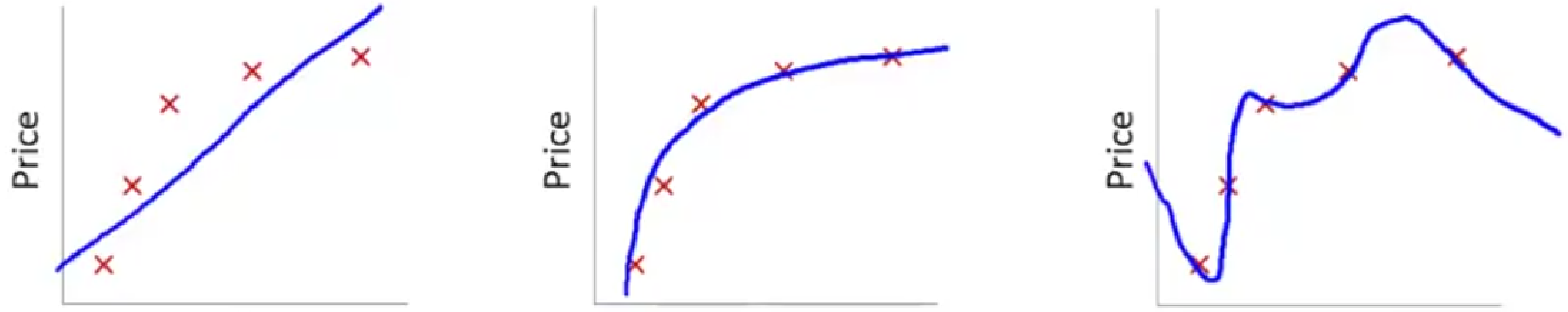
gradient(2) = [code to compute $\frac{\partial}{\partial\theta_1}J(\theta)$] ;

$\vdots$

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

on, rather than gradient 0, gradient

Example: Linear regression (housing prices)



$\rightarrow \theta_0 + \theta_1 x$ "Underfit" "High $\underline{\text{bias}}$"

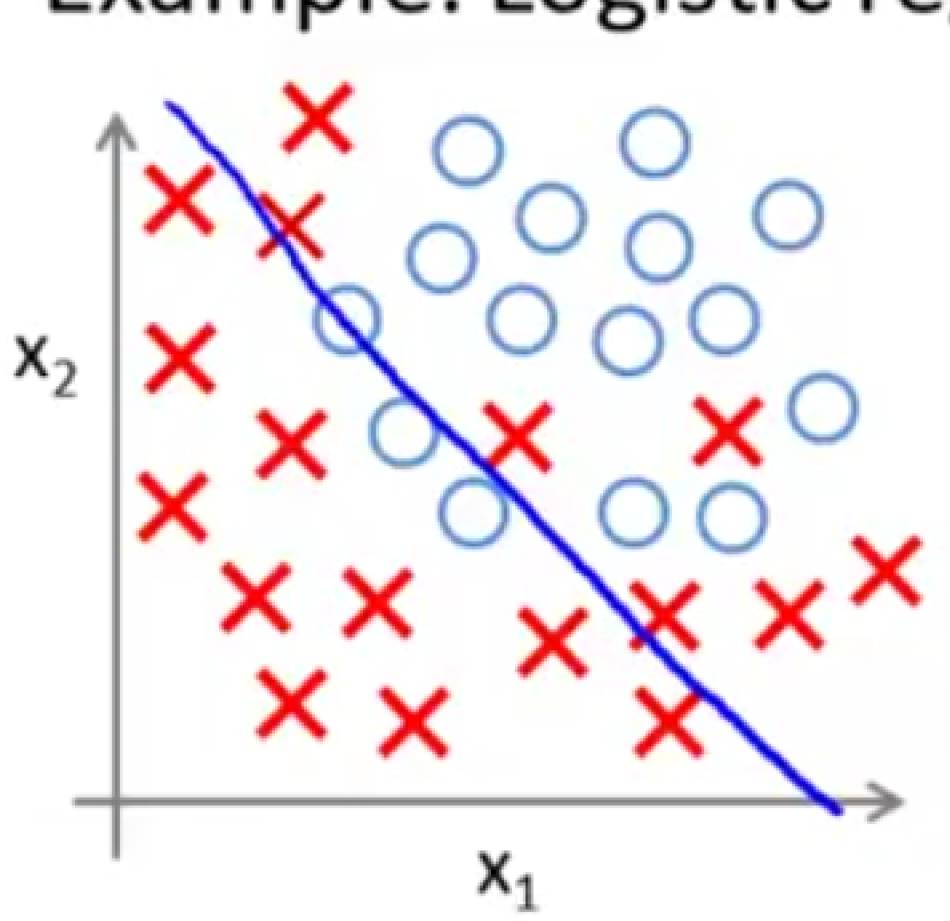
$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2$ "Just right"

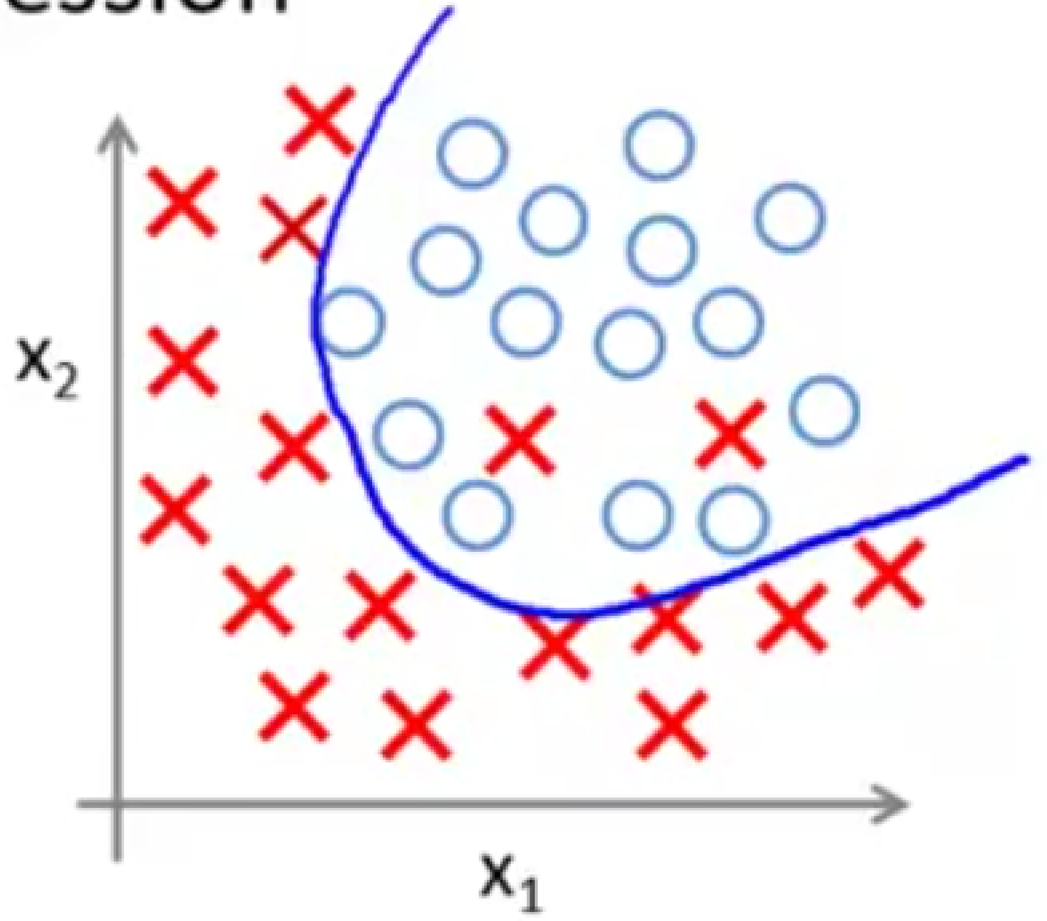
$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 \underline{x^4}$ "Overfit" "High $\underline{\text{variance}}$"

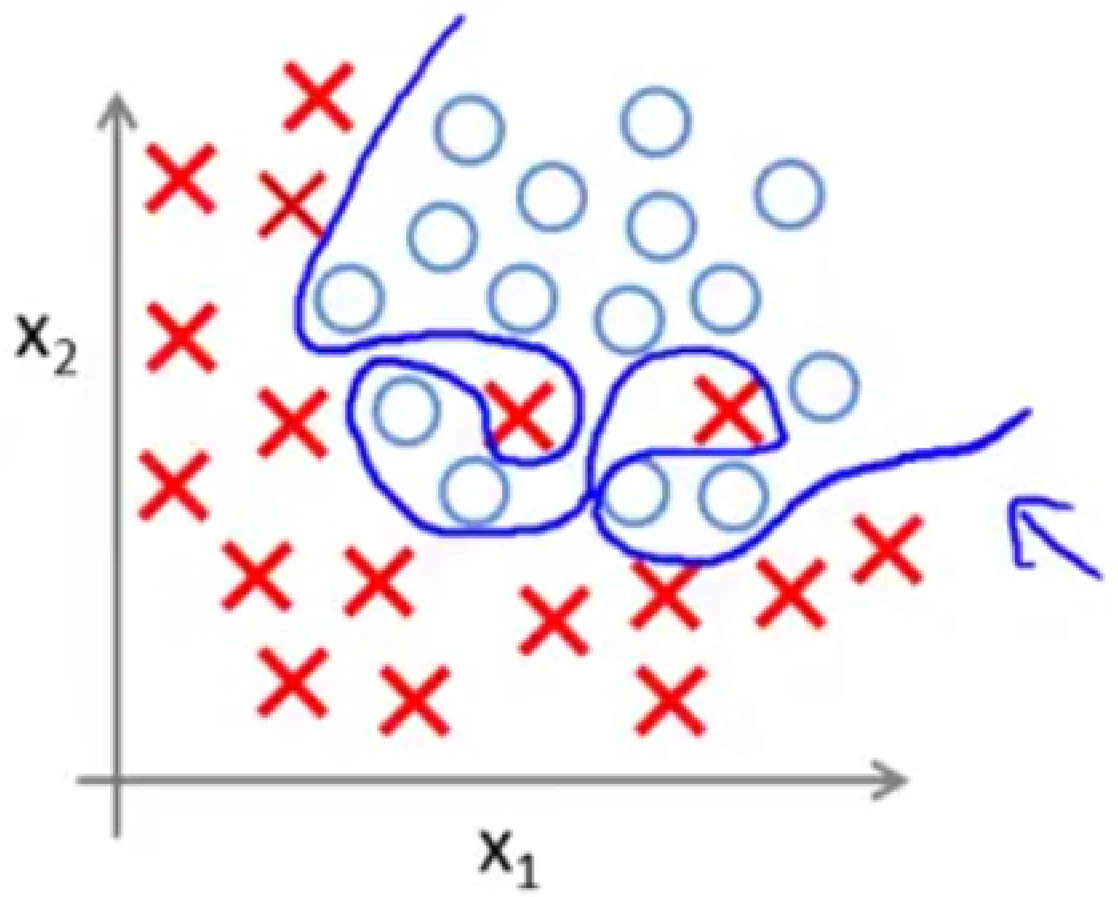
Overfitting: If we have too many $\underline{\text{features}}$, the learned hypothesis may fit the training set very well ($J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 \approx 0$), but fail to generalize to new examples (predict prices on new examples).

Andrew Ng

Example: Logistic regression







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

(g = sigmoid function)

"Underfit"

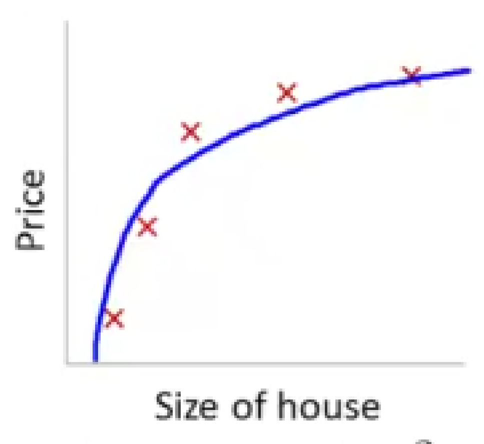
$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2$
$+\theta_3 x_1^2 + \theta_4 \underline{x_2^2}$
$+\theta_5 \underline{x_1 x_2})$

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

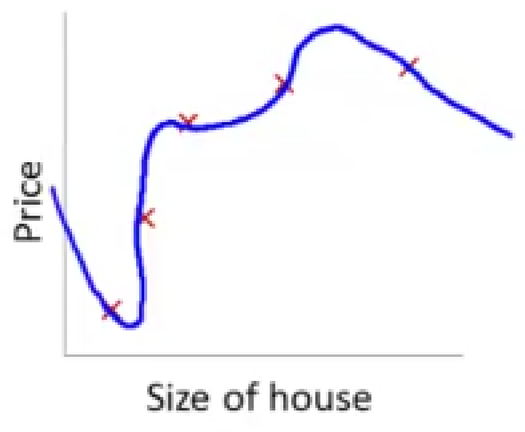
"Overfit"

Andrew Ng

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.

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

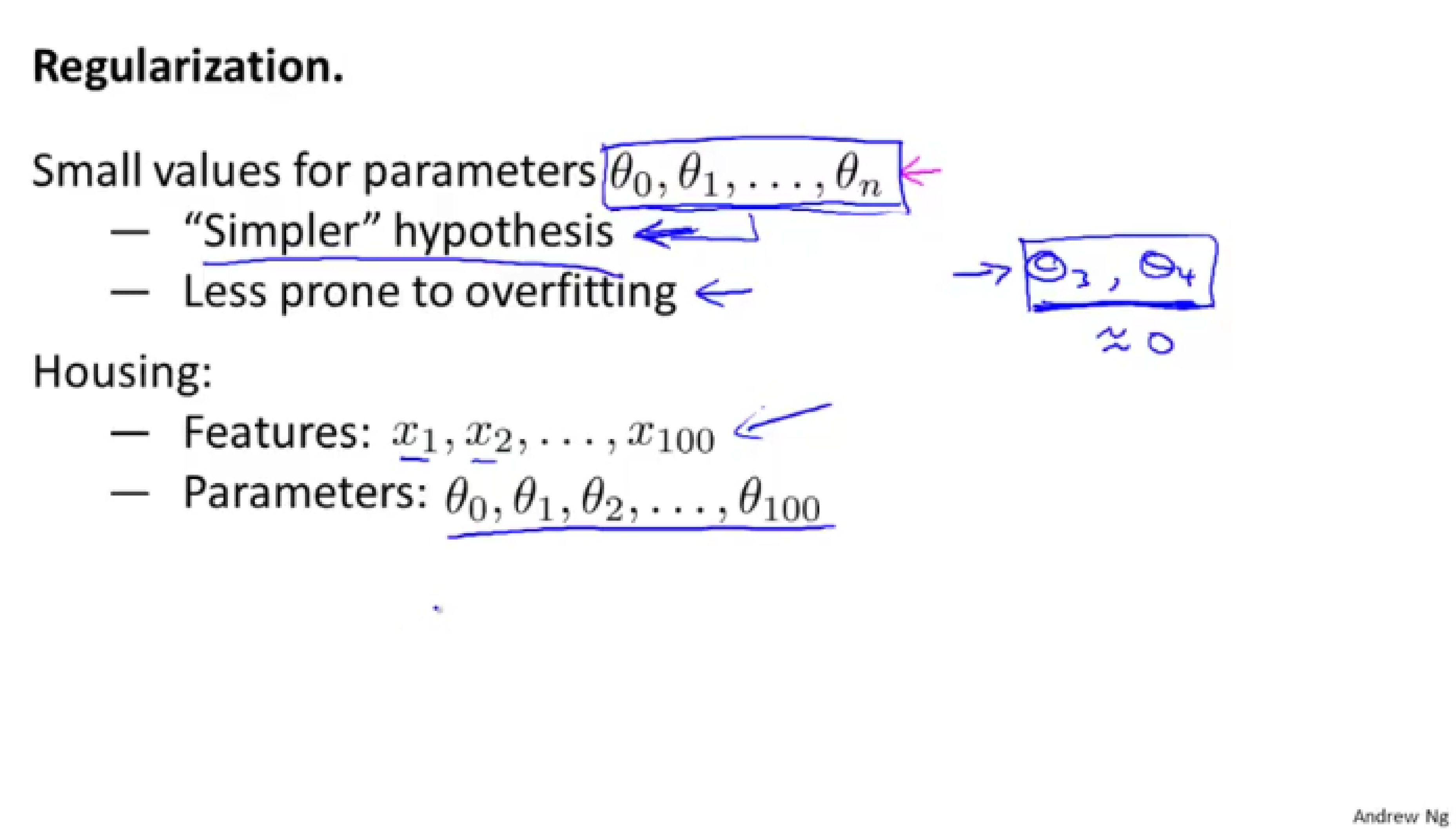
$\theta_3 \approx$

Andrew Ng

Let's say I take this objective and modify it and add to it, plus 1000 theta 3 squared, plus 1000 theta 4 squared. 1000 I am just writing down as some huge number. Now, if we were to minimize this function, the only way to make this new cost function small is if theta 3 and data 4 are small, right?

Because otherwise, if you have a thousand times theta 3, this new cost functions gonna be big. So when we minimize this new function we are going to end up with theta 3 close to 0 and theta 4 close to 0, and as if we're getting rid of these two terms over there.

And if we do that, well then, if theta 3



Regularization.

Small values for parameters $\boxed{\theta_0, \theta_1, \dots, \theta_n}$
- "Simpler" hypothesis
- Less prone to overfitting

$\rightarrow \boxed{\theta_3, \theta_4} \approx 0$

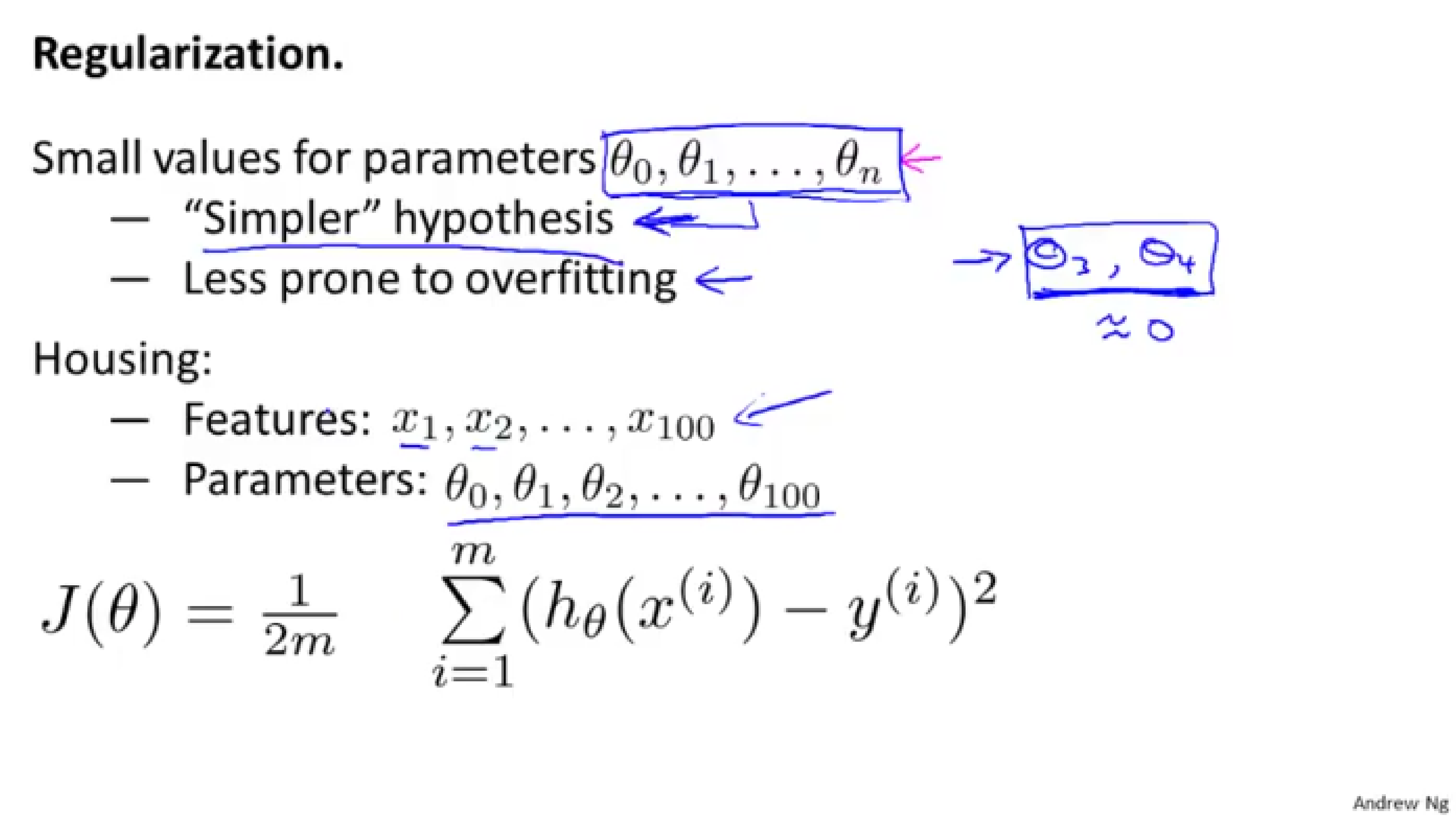
Housing:
- Features: $x_1, x_2, \dots, x_{100}$
- Parameters: $\theta_0, \theta_1, \theta_2, \dots, \theta_{100}$

Andrew Ng

So we have a hundred or a hundred one parameters. And we don't know which ones to pick, we don't know which parameters to try to pick, to try to shrink. So, in regularization, what we're going to do, is take our cost function, here's my cost function for linear regression.

And what I'm going to do is, modify this cost function to shrink all of my parameters, because, you know, I don't know which one or two to try to shrink. So I am going to modify my cost function to add a term at the end. Like so we have square brackets here as well.

When I add an extra regularization



Regularization.

Small values for parameters $\boxed{\theta_0, \theta_1, \ldots, \theta_n} \leftarrow$
— "Simpler" hypothesis $\leftarrow$
— Less prone to overfitting $\leftarrow$ $\rightarrow \boxed{\theta_3, \theta_4}$ ≈ 0

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

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

Andrew Ng

So we have a hundred or a hundred one parameters. And we don't know which ones to pick, we don't know which parameters to try to pick, to try to shrink. So, in regularization, what we're going to do, is take our cost function, here's my cost function for linear regression.

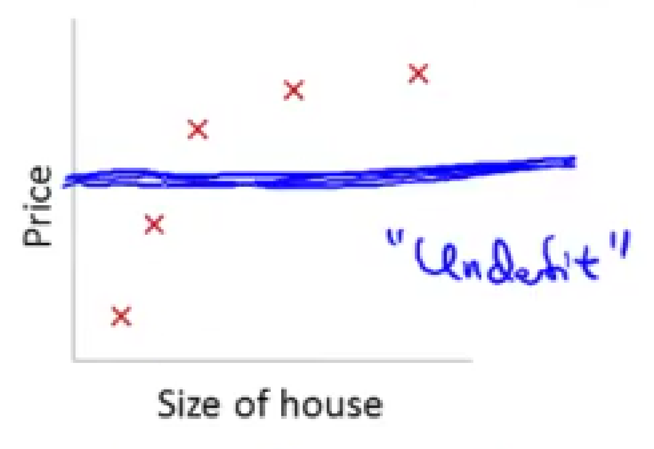
And what I'm going to do is, modify this cost function to shrink all of my parameters, because, you know, I don't know which one or two to try to shrink. So I am going to modify my cost function to add a term at the end. Like so we have square brackets here as well.

When I add an extra regularization

In regularized linear regression, we choose θ to minimize

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

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



$\underline{\theta_1}, \underline{\theta_2}, \underline{\theta_3}, \underline{\theta_4}$

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

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

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

$h_\theta(x)$

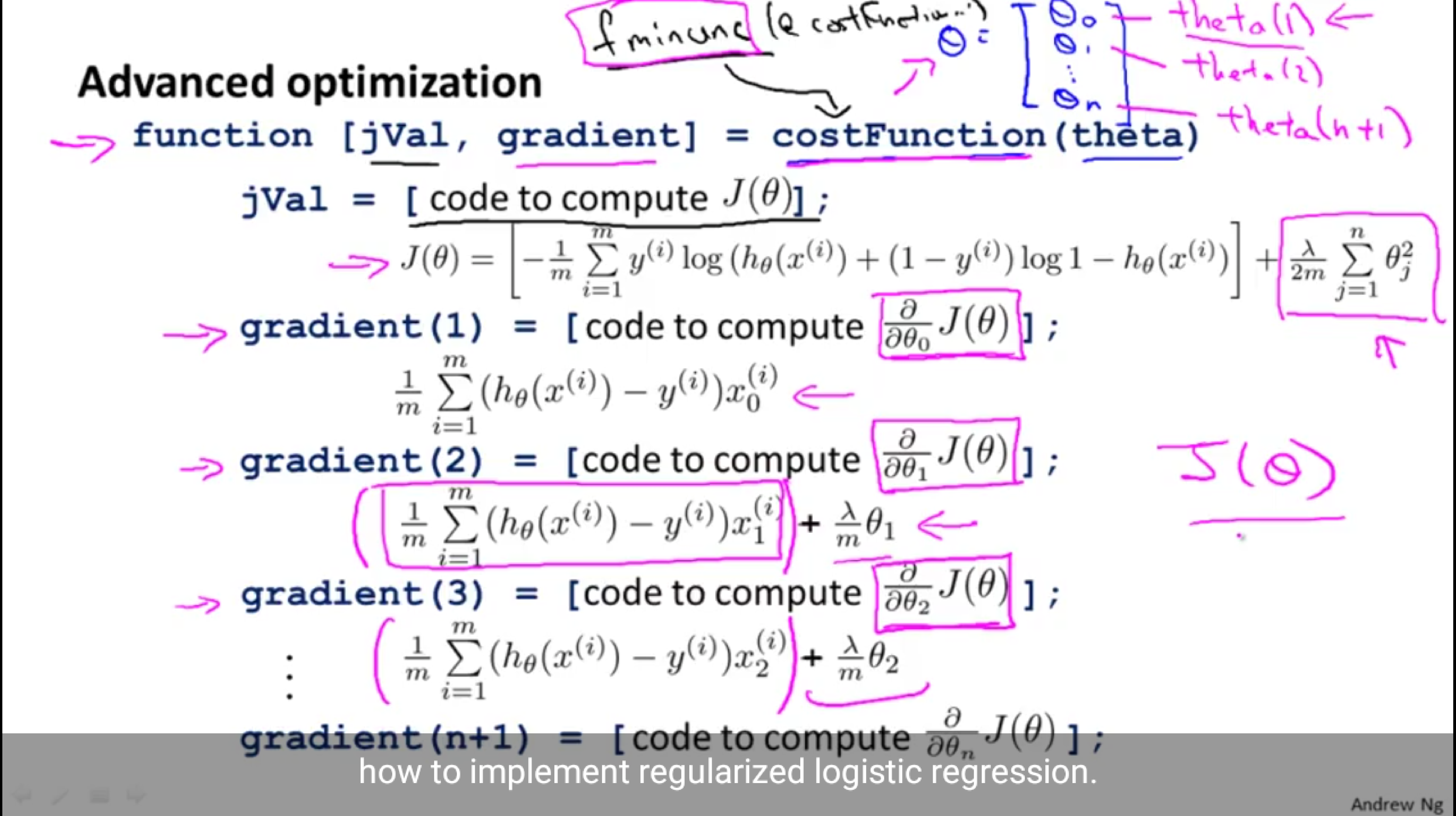
$\theta_0 + \cancel{\theta_1 x} + \cancel{\theta_2 x^2} + \cancel{\theta_3 x^3} + \cancel{\theta_4 x^4}$

Andrew Ng

In regularized linear regression, if the regularization parameter monitor is set to be very large, then what will happen is we will end up penalizing the parameters theta 1, theta 2, theta 3, theta 4 very highly. That is, if our hypothesis is this is one down at the bottom.

And if we end up penalizing theta 1, theta 2, theta 3, theta 4 very heavily, then we end up with all of these parameters close to zero, right? Theta 1 will be close to zero; theta 2 will be close to zero. Theta three and theta four will end up being close to zero.

And if we do that, it's as if we're



fminunc(@costFunction...)

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

Advanced optimization

```
function [jVal, gradient] = costFunction(theta)
    jVal = [ code to compute $J(\theta)$];
        $J(\theta) = \left[-\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log\left(h_\theta(x^{(i)}) + (1-y^{(i)}) \log 1 - h_\theta(x^{(i)})\right] + \boxed{\frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2}$
    gradient(1) = [code to compute $\boxed{\frac{\partial}{\partial\theta_0}J(\theta)}$];
        $\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_0^{(i)}$
    gradient(2) = [code to compute $\boxed{\frac{\partial}{\partial\theta_1}J(\theta)}$];
        $\left(\boxed{\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_1^{(i)}}\right) + \frac{\lambda}{m}\theta_1$
    gradient(3) = [code to compute $\boxed{\frac{\partial}{\partial\theta_2}J(\theta)}$];
        $\left(\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_2^{(i)}\right) + \underline{\frac{\lambda}{m}\theta_2}$
    ⋮
    gradient(n+1) = [code to compute $\frac{\partial}{\partial\theta_n}J(\theta)$];
```

$J(\theta)$

how to implement regularized logistic regression.

Andrew Ng