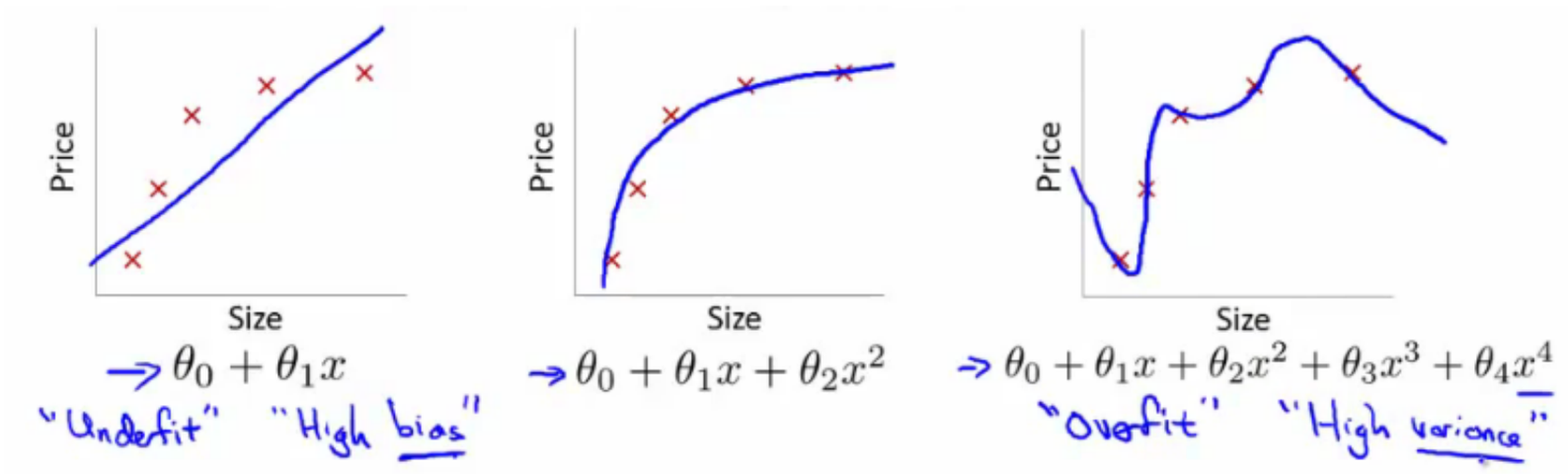
# Regularization


JP @ 2B3E

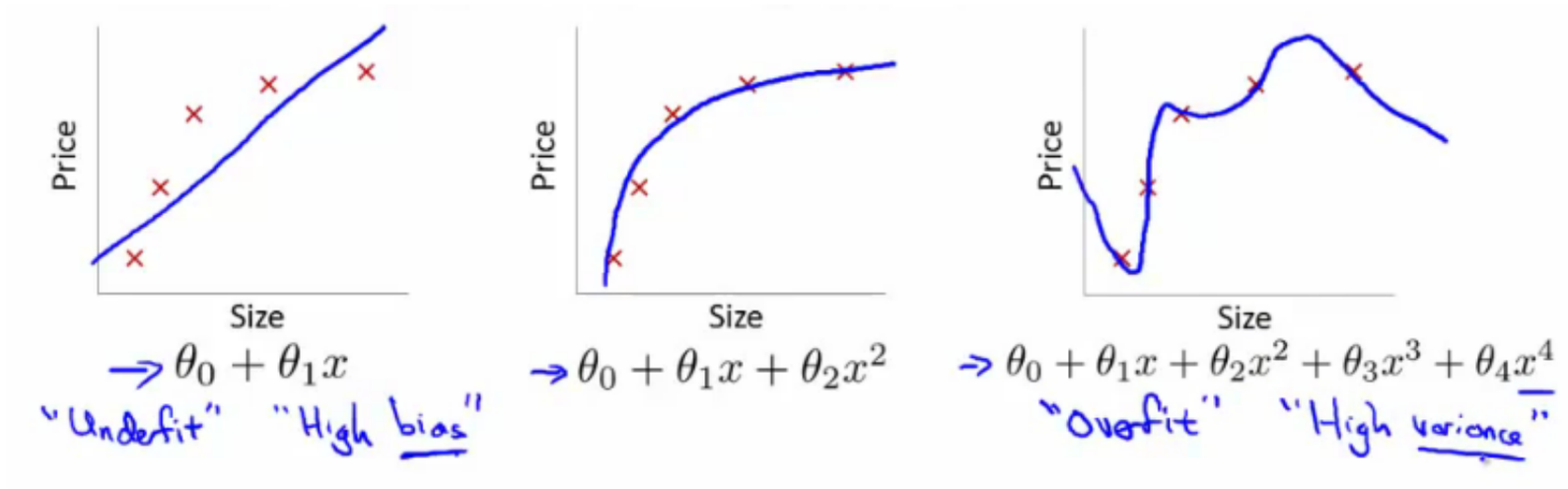
# The Problem of Overfitting

- house pricing example
  - Fit a linear function to the data - not a great model
    - **underfitting** - a.k.a. **high bias**
    - Bias
      - e.g., fitting a straight line to the data —> strong preconception that there should be a linear fit
    - In this case, this is not correct, but a straight(곧은) line can't help being straight(올바르게)!



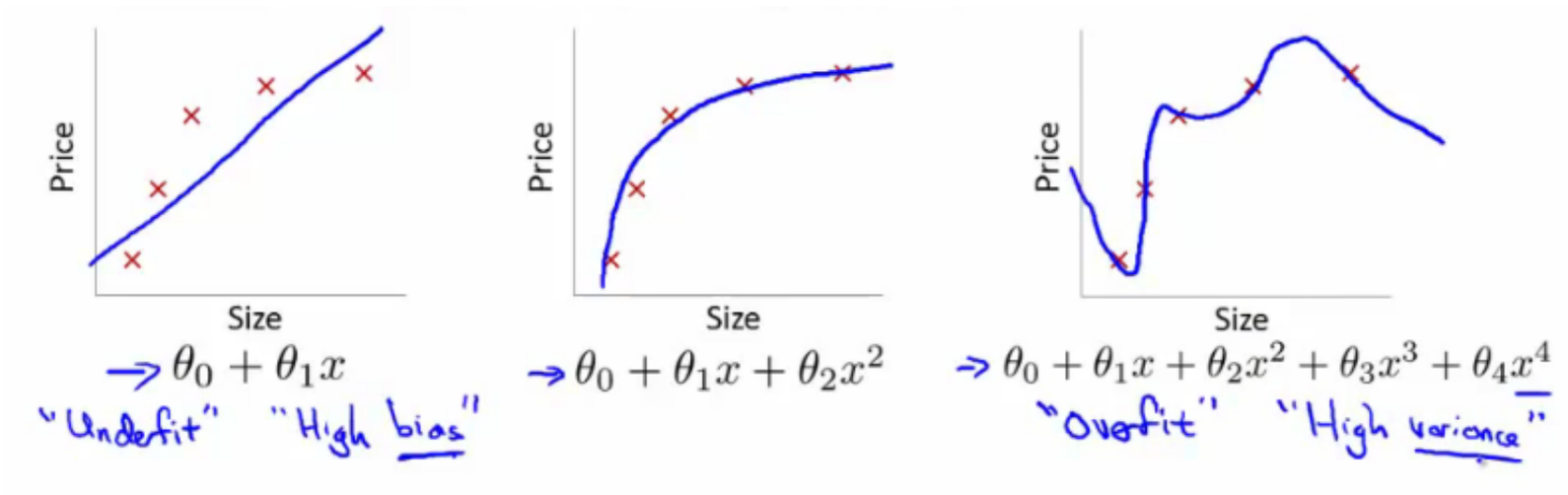
# The Problem of Overfitting

- Fit a quadratic function
  - Works well
- Fit a 4th order polynomial
  - curve fit's through all five examples
    - good job fitting the training set
    - But, this is actually not such a good model
- This is **overfitting** - also known as **high variance**



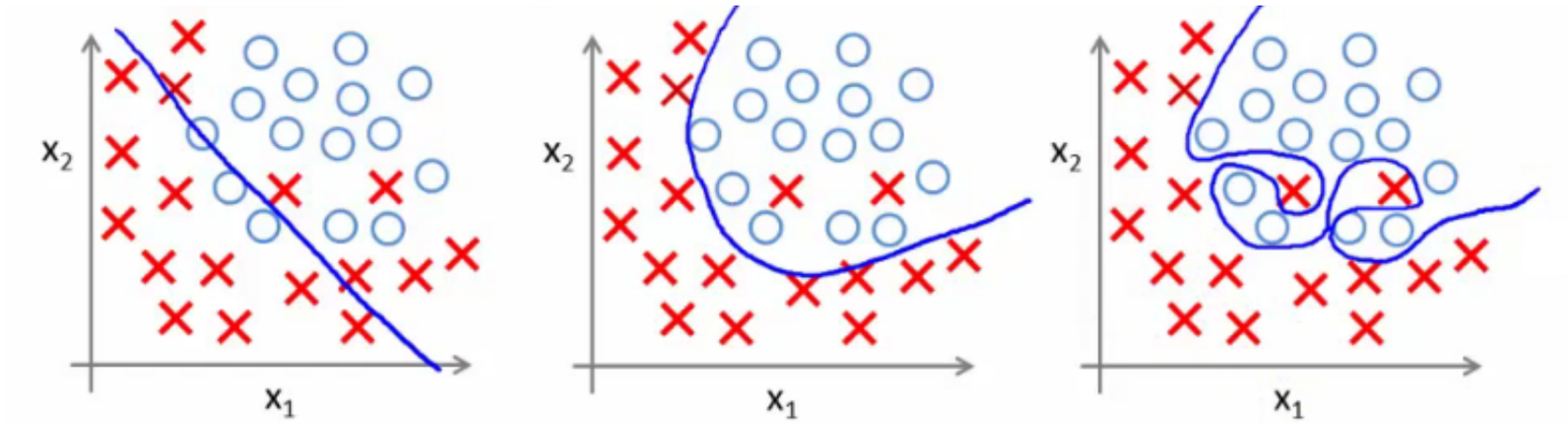
# The Problem of Overfitting

- Algorithm has high variance
  - High variance - if fitting high order polynomial then the hypothesis can basically fit any data
  - Space(차수) of hypothesis is too large
- To recap, if we have too many features then the learned hypothesis may give a cost function of exactly zero
- But this tries too hard to fit the training set
- Fails to provide a *general* solution - **unable to generalize** (apply to new examples)



# Overfitting with logistic regression

- Same thing can happen to logistic regression
  - Sigmoidal function is an underfit
  - But a high order polynomial gives and overfitting (high variance hypothesis)




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

( $g$ = sigmoid function)

**UNDERFITTING**
**(high bias)**

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

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

**OVERFITTING**
**(high variance)**

# Addressing Overfitting

- Plotting hypothesis is one way to decide, but doesn't always work
- Often have lots of a features
  - not just a case of selecting a degree polynomial
  - also harder to plot the data and visualize to decide what features to keep and which to drop

- If you have lots of features and little data - overfitting can be a problem
  - lots of data → lots of parameters

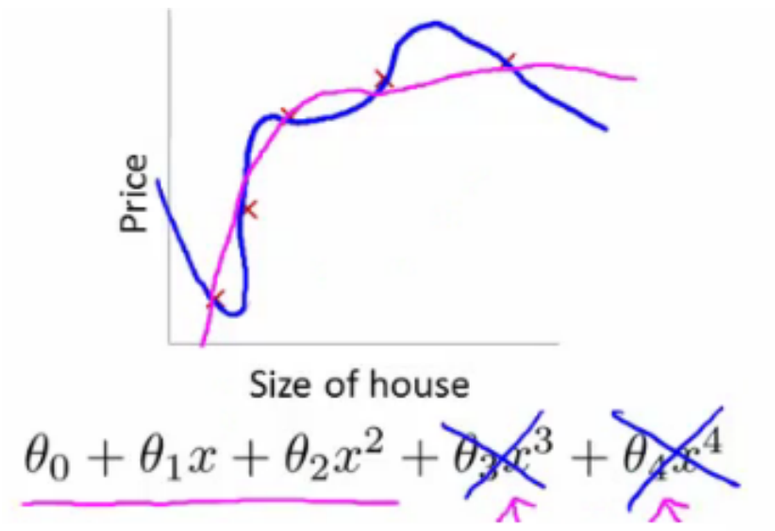
- How do we deal with this?
  - 1) **Reduce number of features**
    - Manually select which features to keep
    - Model selection algorithms are discussed later
      - good for reducing number of features
    - But, in reducing the number of features we lose some information
      - Ideally select those features which minimize data loss, but even so, some info is lost
  - 2) **Regularization**
    - Keep all features, but reduce magnitude of parameters θ
    - Works well when we have a lot of features, each of which contributes a bit to predicting y

# Cost function optimization for regularization

- Penalize and make some of the θ parameters really small: e.g. here $\theta_3$ and $\theta_4$

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

- The addition in blue is a modification of our cost function to help penalize $\theta_3$ and $\theta_4$
  - So here we end up with $\theta_3$ and $\theta_4$ being close to zero (because the constants are massive)
  - So we're basically left with a quadratic function



# Cost function optimization for regularization

- In this example, 2 parameters were penalized
  - More generally, regularization is as follows
- Regularization
  - Small values for parameters corresponds to a simpler hypothesis (you effectively get rid of some of the terms)
  - A simpler hypothesis is less prone to overfitting
- Another example
  - Have 100 features $x_1$, $x_2$, …, $x_{100}$
  - Unlike the polynomial example, we don't know what are the high order terms
    - How do we pick the ones to pick to shrink?
  - With regularization, take cost function and modify it to shrink all the parameters
    - Add a term at the end
      - This regularization term shrinks every parameter
      - By convention you don't penalize $\theta_0$ - minimization is from $\theta_1$ onwards
      - In practice, if you include $\theta_0$ has little impact

# Cost function optimization for regularization

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

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

- **λ** is the **regularization parameter**
  - Controls a trade off between our two goals
    - 1) Want to fit the training set well
    - 2) Want to keep parameters small

# Cost function optimization for regularization

- using the **regularized objective** (i.e. the cost function with the regularization term), we get a much smoother curve which fits the data and gives a much better hypothesis
  - If **λ** is very large we end up penalizing ALL the parameters ($\theta_1$, $\theta_2$ etc.) so all the parameters end up being close to zero
    - If this happens, it's like we got rid of all the terms in the hypothesis
      - This results here is then underfitting
    - So this hypothesis is too biased because of the absence of any parameters (effectively)
- So, **λ** should be chosen carefully - not too big...
  - Need some automatic ways to select **λ**

# Regularized linear regression

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

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

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

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

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

for regularization we don't penalize $\theta_0$

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

# Regularized linear regression

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

$$(1 - \alpha\frac{\lambda}{m})$$

- is going to be a number less than 1 usually
- usually learning rate is small and m is large
  - So this typically evaluates to (1 - a small number)
  - So the term is often around 0.99 to 0.95
- This in effect means $\theta_j$ gets multiplied by 0.99
  - means “reduce” $\theta_j$ a little
  - the second term is exactly the same as the original gradient descent

# Regularization for logistic regression

- We saw earlier that logistic regression can be prone to overfitting with lots of features
- Logistic regression cost function is as follows;

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

- To modify it we have to add an extra term

$$+ \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

- This has the effect of penalizing the parameters $\theta_1$, $\theta_2$ up to $\theta_n$
- Means, like with linear regression, we can get what appears to be a better fitting lower order hypothesis

# Regularization for logistic regression

- How do we implement this?
  - Original logistic regression with gradient descent function was as follows

$$\theta_j := \theta_j - \alpha \quad \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$
$$(j = 0, 1, 2, 3, \ldots, n)$$

- Again, to modify the algorithm we simply need to modify the update rule for $\theta_1$, onwards
  - Looks cosmetically the same as linear regression, except obviously the hypothesis is very different

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

# Normal Equation

- Normal equation is the other linear regression model

normal equation : $\theta=(X^TX)^{-1}X^Ty$

- Analytically find global optimum
- No feature scaling needed
- Complexity is high: $O(n^3)$

| Gradient Descent | Normal Equation |
| --- | --- |
| alpha를 선택해야함 | alpha를 선택할 필요 없음 |
| 반복 연산 | 반복 연산 없음 |
| n이 클 때 잘 작동함 | n이 크면 느림 |

# Regularization with the normal equation

- Minimize the J(θ) using the normal equation
- To use regularization we add a term (+ λ [n+1 x n+1]) to the equation
  - [n+1 x n+1] is the n+1 identity matrix

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

**e.g. if n = 2** $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

# Other Regularization Methods

- Dropout
- Early Stopping
- Batch Normalization