Regularization
AI4003-Applied Machine Learning

Dr. Mohsin Kamal

Department of Electrical Engineering
National University of Computer and Emerging Sciences, Lahore, Pakistan

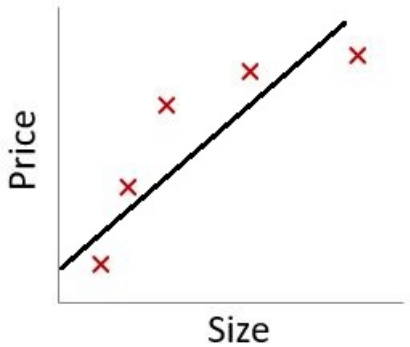
So far we have discussed following learning algorithms:

- Linear Regression
- Logistic Regression

These both work well for many problems but works very poorly in some cases.

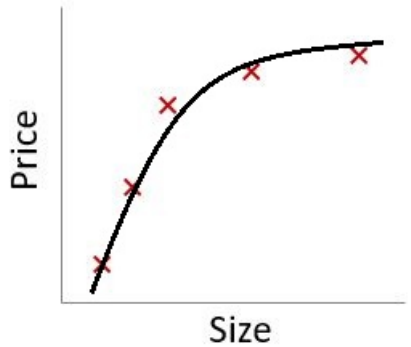
What is overfitting?
How can it be solved?

Example: Linear regression



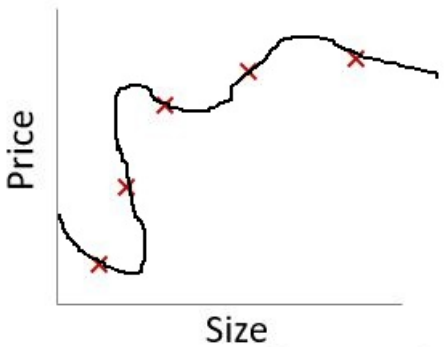
$\theta_0 + \theta_1 x$

"Underfit"



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

"Just right"

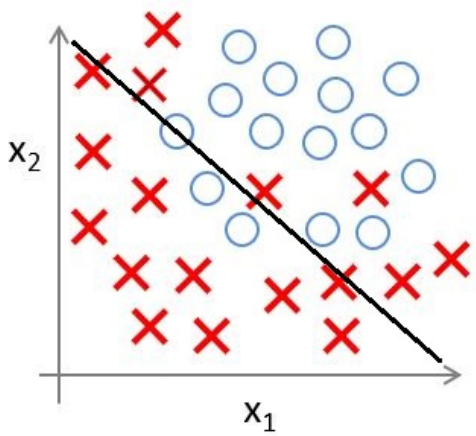


$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

"Overfit" having 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).

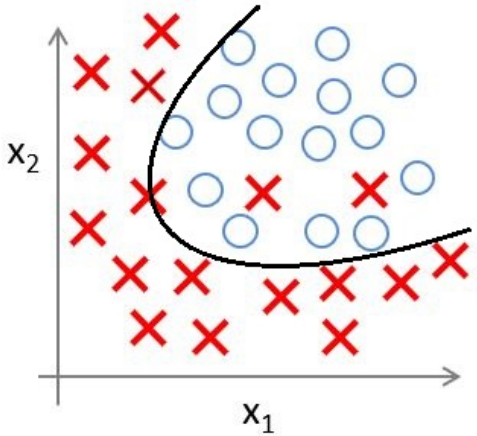
Example: Logistic regression



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

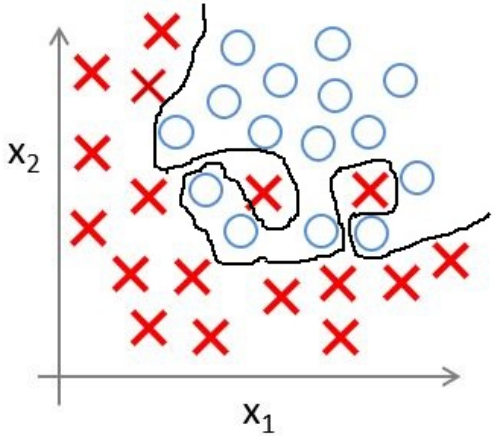
(g = sigmoid function)

"Underfit"



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

"Just Right"

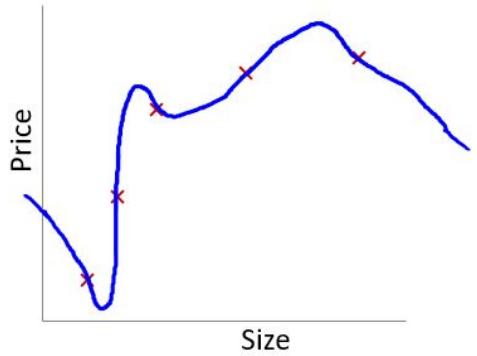


$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \dots)$

Overfit"

Addressing overfitting

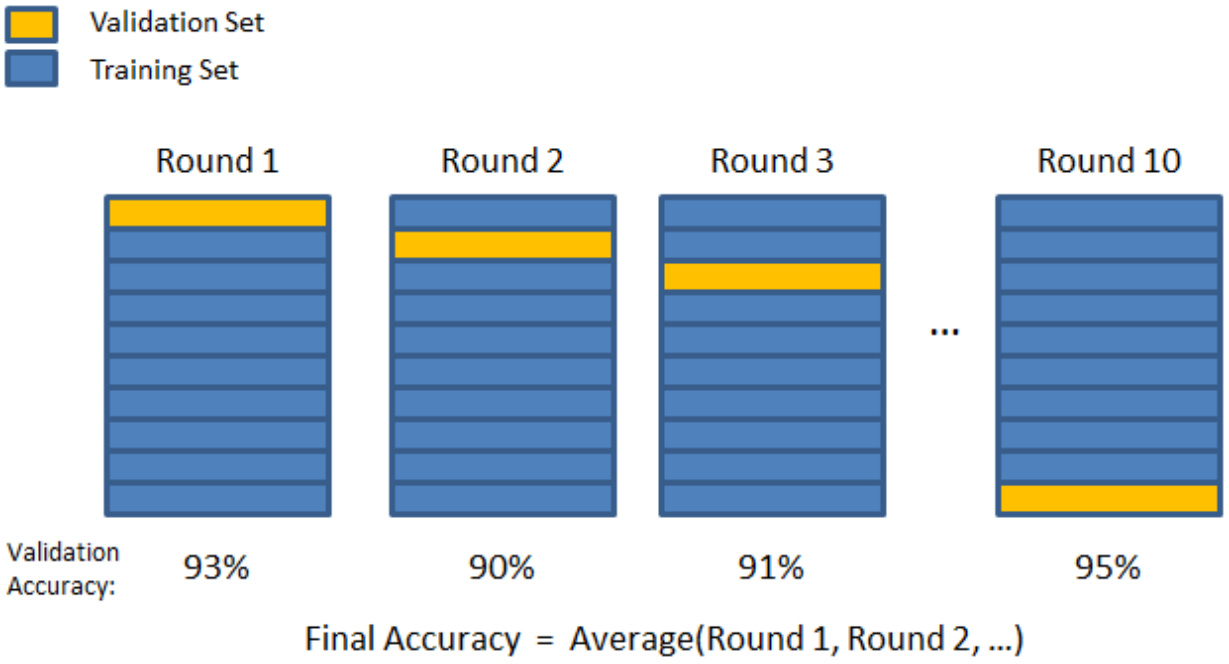
When we have more features and small training data then we may face overfitting.



x_1 = size of house
x_2 = no. of bedrooms
x_3 = no. of floors
x_4 = age of house
x_5 = average income of neighborhood
x_6 = kitchen size
⋮
x_{100} =

Training and Test Splits

- Splits can be 2/3 for training and 1/3 for test
  - must be random
  - Pandas has a DataFrame function 'sample' for this
- Can use K-Fold cross validation*

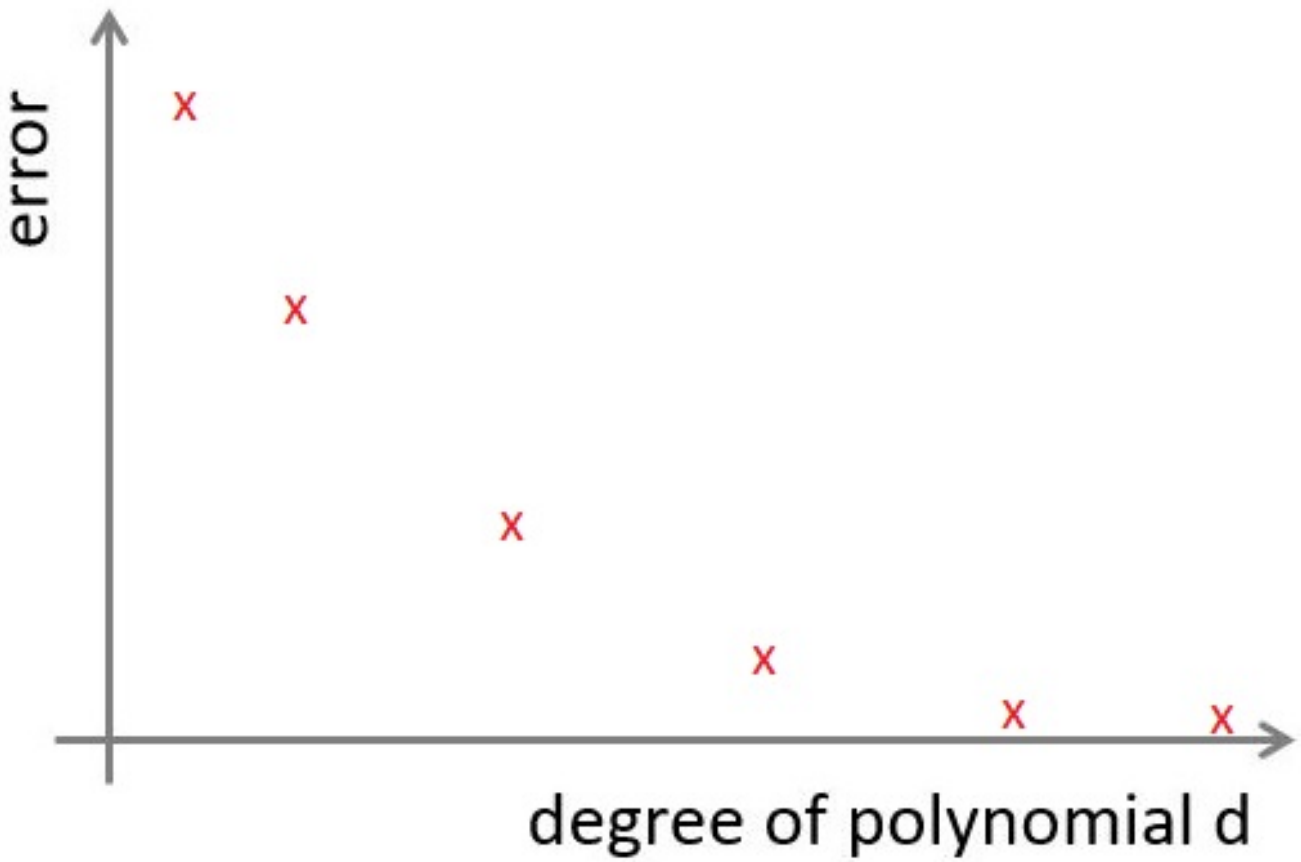


*source: https://chrisjmccormick.wordpress.com/2013/07/31/k-fold-cross-validation-with-matlab-code/

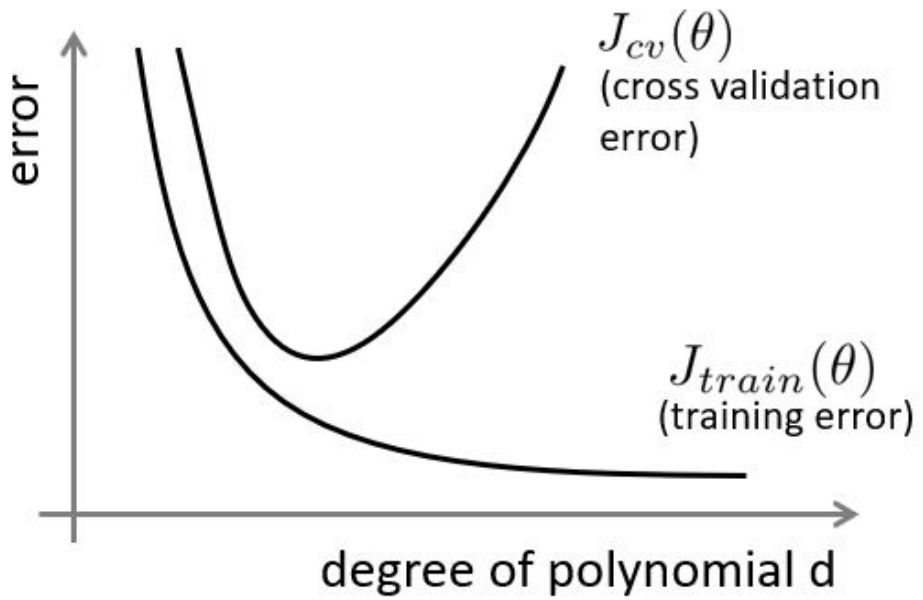
Training error vs. test error

- Reducing training error is easy
- Test error is a better measure
- Important that test data is not touched during training

Model complexity basics



Bias vs. variance



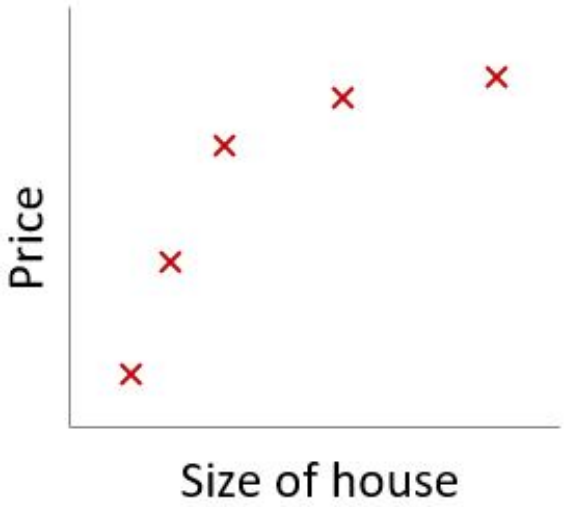
Bias (underfit):
$J_{train}(\theta)$ will be high
$J_{cv}(\theta) \approx J_{train}(\theta)$

Variance (overfit):
$J_{train}(\theta)$ will be low
$J_{cv}(\theta) >> J_{train}(\theta)$

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.



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

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

Suppose we penalize and make θ_3, θ_4 really small.

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

making $\theta_3 \approx 0, \theta_4 \approx 0$ will solve the problem

Ridge Regression/L2-Regularization

Small values for parameters $\theta_0, \theta_0, \cdots, \theta_n$

- "Simpler" hypothesis
- Less prone to overfitting

Housing:

- Features: $x_1, x_2, \cdots, x_{100}$
- Parameters: $\theta_0, \theta_0, \cdots, \theta_{100}$

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

$\lambda \sum_{i=1}^{n} \theta_j^2$ is called "regularization term" while λ is the "regularization parameter".

Ridge Regression/L2-Regularization

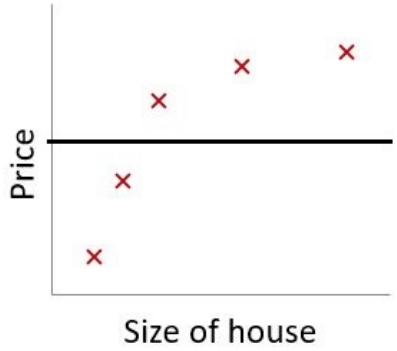
$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{i=1}^{n}\theta_j^2\right] \quad (1)$$

We have two goals in equation 2, i.e., the first term fits the training data well while the second term keeps the parameters values small hence performs regularization.
λ controls the trade-off between the two terms.

In regularized linear regression, we choose θ to minimize equation 2.

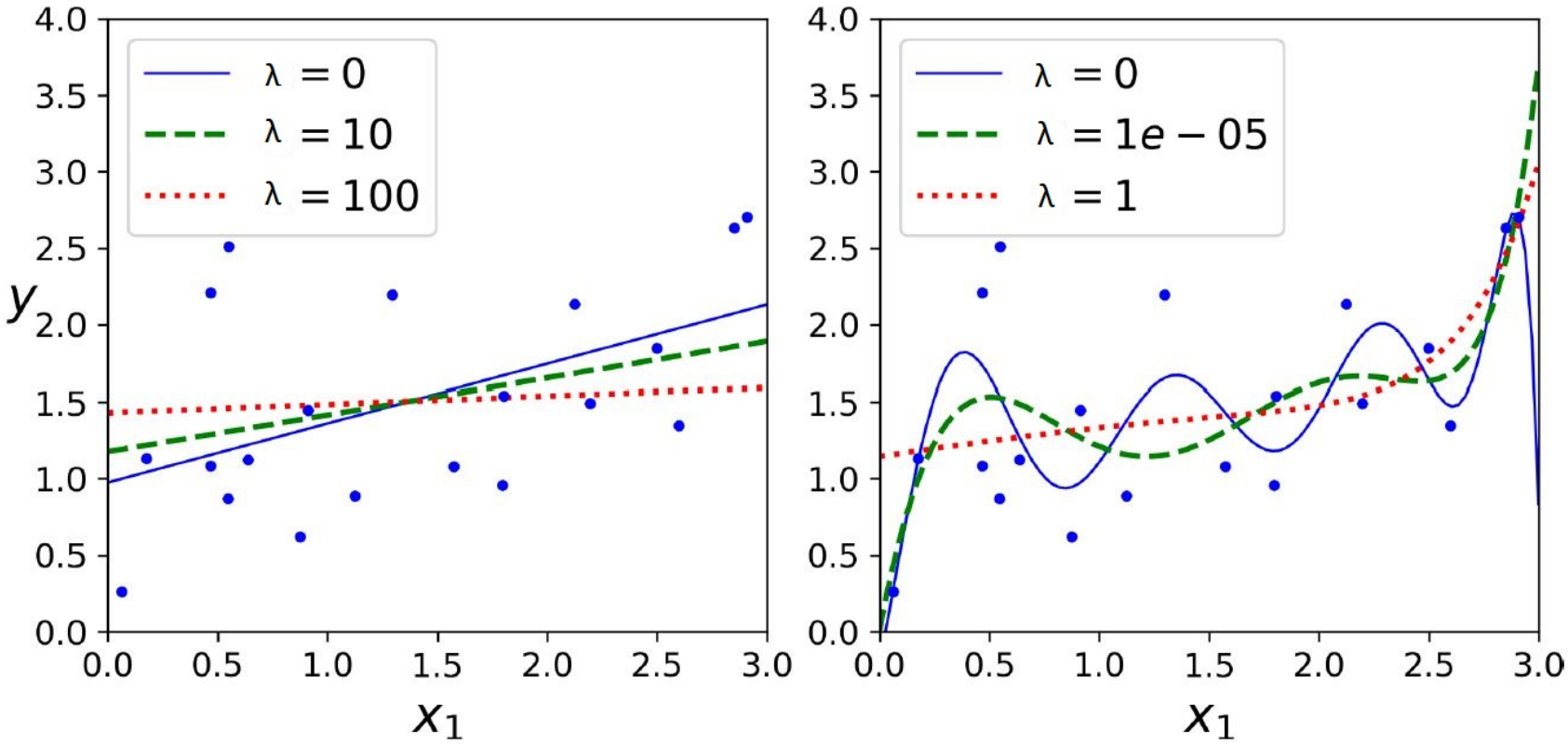
What if λ is set to an extremely large value (perhaps for too large for our problem, say $\lambda = 10^{10}$)?

- We will penalize the parameters associated with slope in our hypothesis
- These parameters will become approximately equal to zero
- This will result in underfitting
- Gradient descent will fail to converge.



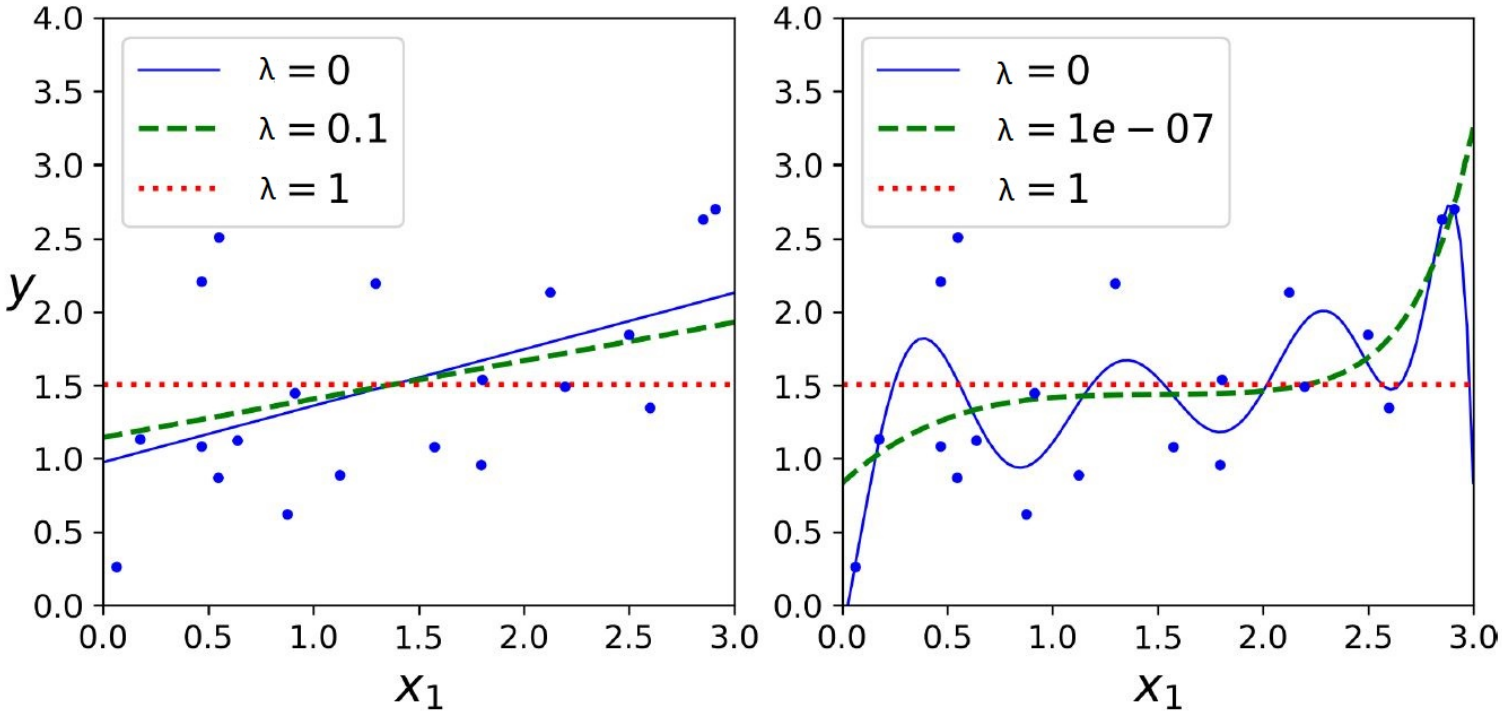
$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

Ridge Regression (left: linear model, right: order-10 polynomial)



Lasso Regression/L1-Regularization

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



Lasso Regression/L1-Regularization

An important characteristic of Lasso Regression is that it tends to completely eliminate the weights of the least important features (i.e., set them to zero). For example, the dashed line in the right plot on Figure in previous slide (with $\lambda = 10^{-7}$) looks quadratic, almost linear: all the weights for the high-degree polynomial features are equal to zero. In other words, Lasso Regression automatically performs feature selection and outputs a sparse model (i.e., with few nonzero feature weights).

Gradient descent

As we know

$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$

so,

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

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

The algorithm will become:

Repeat {

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

$\theta_j := \theta_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} + \frac{\lambda}{m} \theta_j \right] \quad (j = 1, 2, 3, \cdots, n)$

}

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

Normal equation/closed-form equation

As we know

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

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

The matrix will be $(n+1) \times (n+1)$

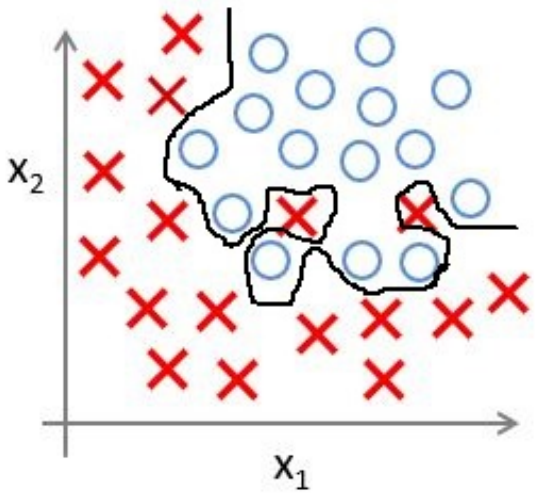
Non-invertibility

Supose $m \leq n$,
where $m =$ no. of examples and $n =$ No. of features then,
$\theta = (X^T)^{-1} X^T y$ is no-invertable or singular

If $\lambda > 0$,

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

is invertable



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

Cost function:

$$J(\theta) =$$

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

$$\forall \quad \theta_1, \theta_2, \cdots, \theta_n$$

Gradient descent

Repeat {

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

$$\theta_j := \theta_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} + \frac{\lambda}{m} \theta_j \right] \quad (j = 1, 2, 3, \cdots, n)$$

}

It looks like the gradient descent of linear regression but the only difference is the equation of $h_\theta(x)$

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