

Chapter 7

Regularization

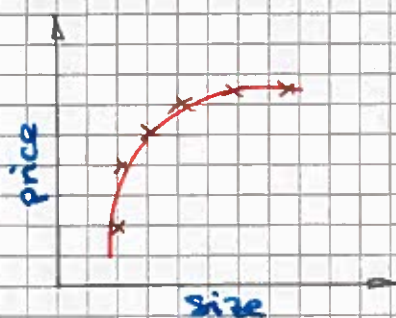
Regularization - The problem of ^{over}fitting - Chapter 7 [1]

Example: linear regression (house prices)



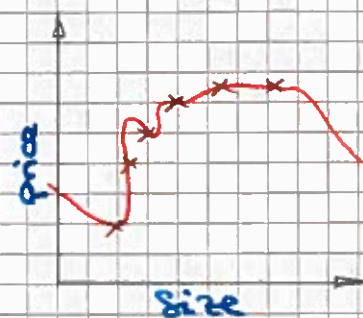
$$\theta_0 + \theta_1 x$$

"under-fit" or
"high bias"



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

"Just right"



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

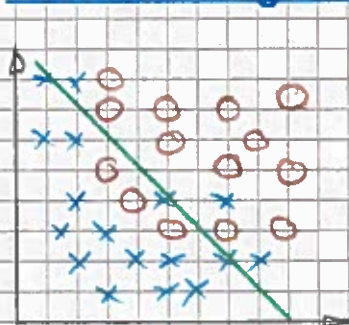
"overfitting"
"high variance"

Overfitting:

if we have too many features, the learned hypothesis may fit the training set very well:

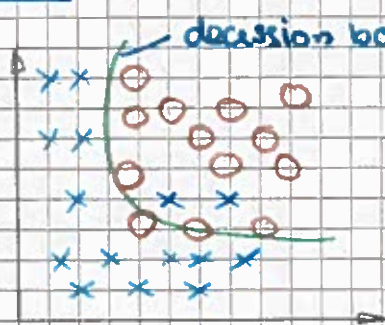
$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 \approx 0$, but fails to generalize the new examples (predict prices on new examples)

Example: logistic regression



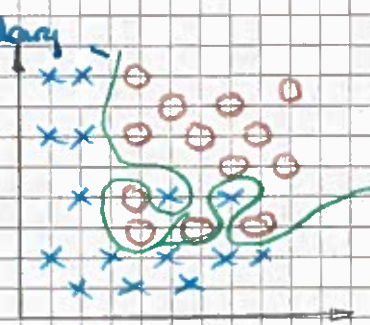
$$h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

"underfitting" or "high bias"



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

generalizes well



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

"overfitting" or
"high variance"
not generalizing well

- high variance, a high order polynomial can fit to any data, but lacks of generalization.
- sigmoid function is an underfit with logistic regression

Overfitting can become a problem, if we have a lot of features but just a little training sets.

Options:

1) Reduce the number of features

disadvantage:

- Manually select which features to keep.

throw away information

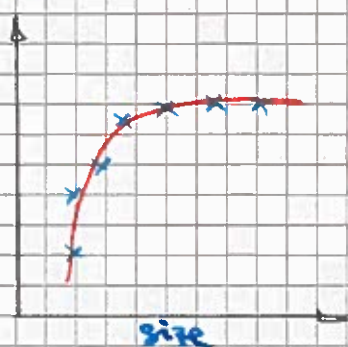
- Model selection algorithm (later in course), good for reducing features.

2) Regularization:

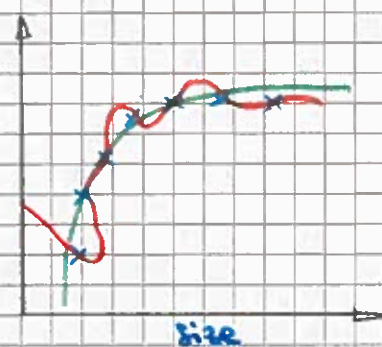
- Keep all the features, but reduce the magnitude/values of parameters θ_j

- Works well, when we have a lot of features, each of which contributes a bit to predict y .

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 ^{bestrafen} and make θ_3, θ_4 really small

$$\min_{\theta} \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + \underbrace{1000 \cdot \theta_3^2}_{\approx 0} + \underbrace{1000 \cdot \theta_4^2}_{\approx 0}$$

↳ if we use $\theta_3 \approx 0, \theta_4 \approx 0$ then we would end up with a quadratic function.

Regularization:

Small values for parameters ($\theta_0, \theta_1, \dots, \theta_n$) lead to:

- "Simpler" hypothesis is
- less prone to overfitting \Rightarrow

Housing:

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

this term shrinks all parameters

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

~~$\theta_0, \theta_1, \dots, \theta_{100}$~~

by convention not used (penalized)

Regularization terms

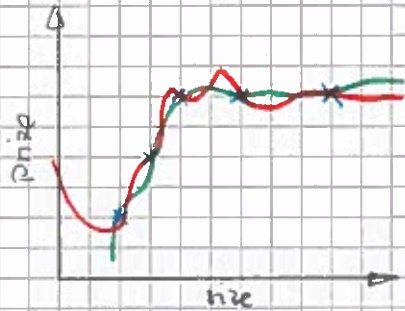
will shrink all parameters $\theta_1, \dots, \theta_n$

\Rightarrow with Regularization we take cost function and modify it to shrink all parameters.

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

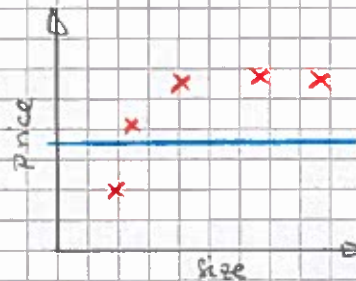
used to keep parameters small

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



Regularization parameter will lead to smooth the curve out. Might give better algorithm.

In regularized linear regression, we choose θ to minimize what if λ is set to an extremely large value (perhaps far too large for our problem, say $\lambda = 10^{10}$) \Rightarrow end up being close to 0



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

$$\rightarrow \approx 0$$

which means that they not exist anymore.

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

\rightarrow we are left with θ_0 and

this is a straight line \Rightarrow "underfit"

$\rightarrow h_{\theta}(x) = \theta_0 \Rightarrow$ hypothesis is too biased

we should choose λ carefully - not too big. \rightarrow too strong preconception

λ is the regularization parameter

- Controls a trade off between our goals

1. want to fit the training set

2. want to keep parameters small

Regularized Linear Regression

5

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:

Repeat \sum

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

$\frac{\partial}{\partial \theta_0} \cdot J(\theta)$

regularization does not penalize θ_0

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

(j = 1, 2, 3, 4, ..., n)

\Downarrow

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

$1 - \frac{\lambda}{m} < 1 \Rightarrow 0.99 \Rightarrow \theta_j \cdot 0.99 \Rightarrow$ makes it smaller slightly smaller than 1

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

Normal Equation:

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

$m \times (n+1)$

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

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

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

e.g. $n=2$

$$\Rightarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

identity matrix?

Suppose $m \leq n$,
#examples #features

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

=> with Octaves `pinv()`, it gives some reasonable values

If $\lambda > 0$, then

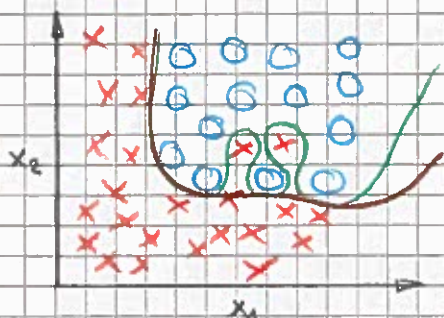
$$\Theta = \underbrace{\left(X^T X + \lambda \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ & \ddots \end{bmatrix} \right)^{-1}}_{\text{invertible}} X^T y$$

=> Take care of non invertibility of matrices

Regularized Logistic Regression

[7]

Regularized logistic Regression



$$h_{\theta}(x) = \frac{1}{1 + e^{-\theta_0 - \theta_1 x_1 - \theta_2 x_1^2 - \theta_3 x_1^2 x_2 - \theta_4 x_1^2 x_2^2 - \theta_5 x_1^2 x_2^3 - \dots}}$$

Cost function:

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

penalizing $\Rightarrow \theta_1, \theta_2, \dots, \theta_n$

leads to the curve above

Gradient descent:

→ this has the effect to penalize $\theta_1, \dots, \theta_n$
to get a better fitting lower order hypothesis

Repeat Σ

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

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

(j = ~~0~~ 1, 2, 3, 4, 5, ..., n)

separately managed

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

Advanced optimization

Function [jval, gradient] = costfunction(theta)

jval = code to compute $J(\theta) \Rightarrow 1)$

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

\vdots

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

$$\theta = \begin{bmatrix} \theta_0 \\ \vdots \\ \theta_n \end{bmatrix}$$

\Downarrow

fminunc (@costfunction, ...)

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

$$\text{gradient}(1) : \frac{1}{m} \sum_{i=1}^m (h\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\text{gradient}(2) : \frac{1}{m} \sum_{i=1}^m (h\theta(x^{(i)}) - y^{(i)}) x_1^{(i)} + \frac{\lambda}{m} \cdot \theta_1$$

\vdots

Ensure that summation doesn't extend the lambda term! \Rightarrow it doesn't