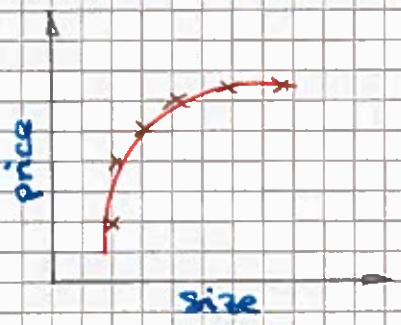
7. REGULARIZATION

Regularization - The problem of overfitting. - Chapter 7 [1]

Example: linear regression (house prices)

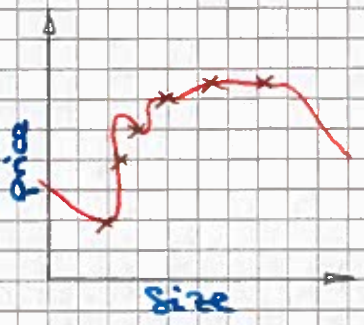
$\theta_0 + \theta_1 x$

"underfit" or "high bias"



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

"just right"



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

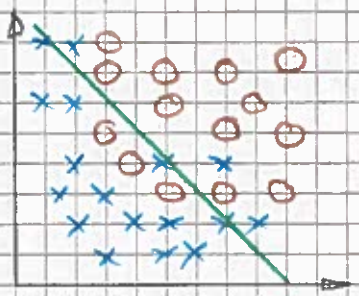
"overfitting"
"high variant"

Overfitting:

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

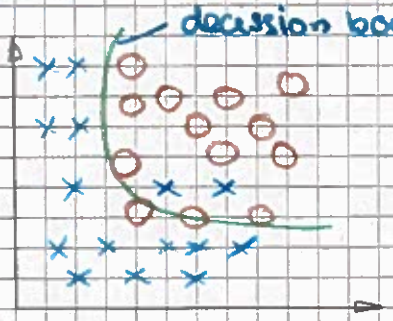
$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h\theta(x^{(i)}) - y^{(i)})^2 \approx 0$), but fails to generalize the new examples. (predict prices on new examples)

Example: logistic regression



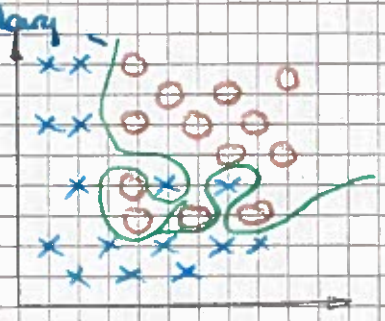
$h\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

"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



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

"overfitting" or "high variance"
not generalizing well

- high variance, a high order polynomio can fit to any data, but lags of generalization.
- Sigmoid function is an underfit with logistic regression

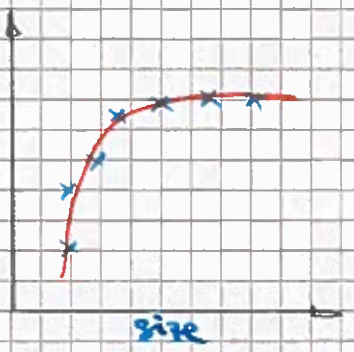
Overfitting can become a problem, if we have a lot of features but just a little training sets.

Options:

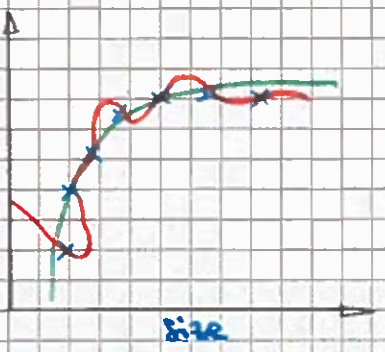
1) Reduce the number of features — disadvantage: throw away information
   - Manually select which features to keep.
   - Model selection algorithm (later in course), good for reducing features.

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

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

↳ 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 =0

Housing:

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

this term shrinks all parameters

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

~~θ_0~~, $\theta_1, \theta_2, \dots \theta_{100}$

by convention not used (penalized)

↳ Regularization term

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

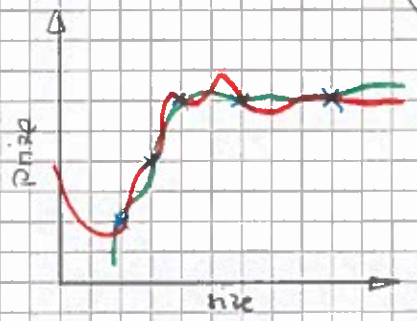
⇒ with Regularization we take cost function and modify it to shrink all parameters.

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

↳ used to keep parameters small

↳ Regularization parameter

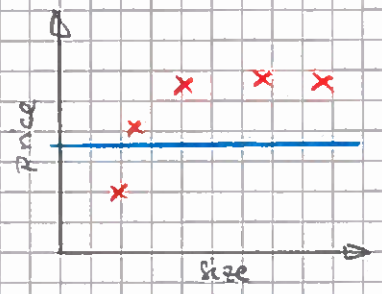
$\min_\theta J(\theta)$



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 $\lambda = 10^{10}$) ⇒ end up being close to 0



$\theta_1, \theta_2, \theta_3, \theta_4$
↳ ≈ 0
which means that they not exist anymore.

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

↳ we are left with θ_0 and this is a straight line ⇒ "underfit"

↳ $h_\theta(x) = \theta_0$ ⇒ hypothesis to biased ↳ to strong preconception

we should choose λ carefully - not too big.

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:

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

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

Gradient descent:

Repeat $\{$

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

regularization does not penalize θ_0

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

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

$(j = \cancel{0}\ 1,2,3,4,\dots n)$

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

$\Downarrow$

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

$\Rightarrow 0{,}99 \quad \Rightarrow \theta_j \cdot 0{,}99 \Rightarrow$ makes it smaller

slightly smaller than 1

Normal Equation:

$$X = \begin{bmatrix} (x^{(1)})^T \\ \vdots \\ (x^{(m)})^T \end{bmatrix}_{m\times(n+1)} \qquad y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix} \in \mathbb{R}^m$$

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

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

identity matrix ?

e.g. $n=2$

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

Suppose $m \leq n$,

(m: #examples, n: #features)

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

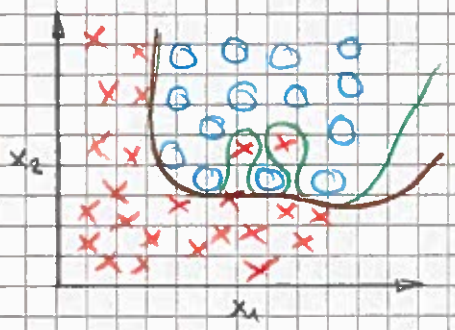
⇒ with Octaves pinv( ), it gives some reasonable values

If $\lambda > 0$, then

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

⇒ Take care of non invertibility of matrixes

Regularized Logistic Regression



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

Cost function:

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

$$+ \underbrace{\frac{\lambda}{2m}\sum_{j=1}^{n} \theta_j^2}_{\text{leads to the curve above}} \quad \text{penalizing} \Rightarrow \theta_1, \theta_2 \dots \theta_n$$

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

Gradient descent:

Repeat {

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

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

$(j = \cancel{0}, 1, 2, 3, 4, 5 \dots n)$

Separately managed

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

Advanced optimization

Function [jval, gradient] = costFunction(theta)

jval = code to compute $J(\theta)$ ⇒ 1)
gradient(1) = code to compute $\frac{\partial}{\partial \theta_0} J(\theta)$;
⋮
gradient(n+1) = code to compute $\frac{\partial}{\partial \theta_n} J(\theta)$;

$\theta = \begin{bmatrix} \theta_0 \\ \vdots \\ \theta_n \end{bmatrix}$ ⇓ fminunc(@costFunction .....)

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

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

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

⋮

Ensure that summation doesn't extend the lamda term! ⇒ it doesn't