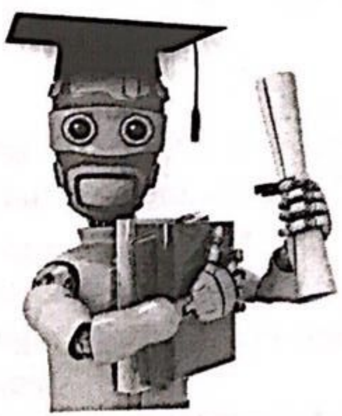
Addressing overfitting: Overfitting'i address etmek için 2 seçenek var:

Options:

1. Reduce number of features. → İşe yarayabilir ancak, useful info'yu da kaybetmiş oluyoruz.
   - Manually select which features to keep. : Gereksizleri atarız.
   - Model selection algorithm (later in course). : Algorithms automatically decide which feature to keep, which to throw out.
2. Regularization. → θ'ların değerini düşürürüz. Data'yı atmadan overfitting'den kurtulabiliriz.
   - 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.

Andrew Ng



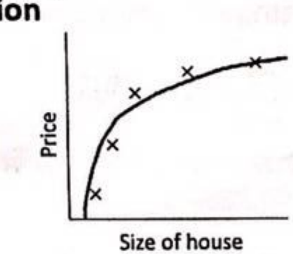
Machine Learning

Regularization

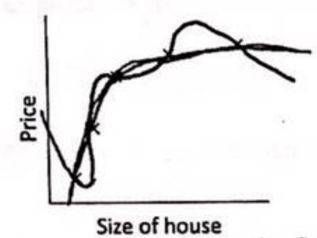
Cost function

→ Overfitting'e sebep olan terimler $\theta_3 x^3 + \theta_4 x^4$ ise ben bunları minimize edersem overfittingi çözerim, yani $\theta_3 \approx 0$ ve $\theta_4 \approx 0$ yaparsam problem çözülür! Çünkü hipotez yine 2. derece gibi olacak x^3 ve x^4'ten çok küçük bir kısım gelecek.

Intuition



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



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

Suppose we penalize and make θ_3, θ_4 really small.

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

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

θ_3 ve θ_4 parametrelerini (overfitting'in sebebi) cost function'a ekliyoruz böylece artık bu parametreleri de minimize edip hipoteze olan etkisini düşürüyoruz.

Andrew Ng

Geçen örnekte sadece Θ_3 ve $\Theta_4 \approx 0$ yaparak simpler hypothesis elde ettik.
More broadly: Θ'lar küçülürse simpler hypothesis ve less change for overfitting

Regularization.

Small values for parameters $\boxed{\theta_0, \theta_1, \ldots, \theta_n}$

- "Simpler" hypothesis ← Smoother
- Less prone to overfitting ← yatkın

→ $\boxed{\Theta_3, \Theta_4}$ ≈ 0

Housing:

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

Regularization Term

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

$\cancel{\Theta_0}$ $\Theta_1, \Theta_2, \Theta_3, \ldots, \Theta_{100}$ $\cancel{\Theta_0}$

Andrew Ng

100 tane Θ var hangi parametreler gereksiz, hangisi higher order bilmiyoruz. Regularization ile cost function for linear regression'ı alıp bütün Θ parametrelerini shrink etmek için, modify ederim. Reg. Term bütün Θ'ları shrink eder, Θ_0 haricinde! $\Theta_0 = 1$ hep! ..

λ: Controls the tradeoff b/w two goals: 1) Fitting the training set well. 2) Keeping the parameters small: Avoid overfitting

$\lambda \downarrow \; \Theta \uparrow$

Goal 2 $\lambda \uparrow \; \Theta \downarrow$

λ: Çok büyürse, underfitting olur! Çok küçülürse, overfitting!

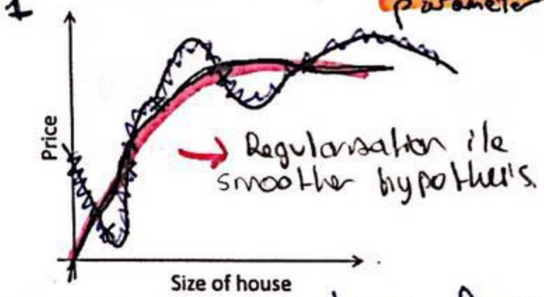
Regularization.

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

regularization parameter

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

Goal 1



→ Regularization ile smoother hypothesis.

Nedenini anlamaman normal diyor. Sena gör çalışın, bunu böyle bil diyor!

Andrew Ng

λ büyürse Θ'lar ≈ 0 olacak underfitting occurs

In regularized linear regression, we choose θ to minimize

Sonuç:

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

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

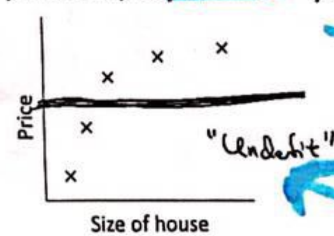
- ✗ Algorithm works fine; setting λ to be very large can't hurt it
- ✗ Algortihm fails to eliminate overfitting.
- ✓ Algorithm results in underfitting. (Fails to fit even training data well).
- ✗ Gradient descent will fail to converge.

Andrew Ng

In regularized linear regression, we choose θ to minimize

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

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



$\Theta_1, \Theta_2, \Theta_3, \Theta_4$

$\Theta_1 \approx 0$, $\Theta_2 \approx 0$, $\Theta_3 \approx 0$, $\Theta_4 \approx 0$

$h_\theta(x) = \Theta_0$

$h_\theta(x)$

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

diğer Θ'lar ≈ 0 olunca sonuç horizontal flat line olur!

Andrew Ng

Yani λ doğru seçilmeli!

→ Linear Regression için iki farklı algorithm öğrenmiştik, 1 based on gradient descent, 1 based on normal equation

→ Şimdi ikisini de alıp, generalise ederek, Regularized Linear Regression'ı oluşturucağız

Machine Learning

Regularization

Regularized linear regression

Gradient Descent için Θ_0'yu aynı şekilde minimize ettiğimizi görürüz çünkü, regularization term $J=1$ den başlar Θ_0'ı etkilemez.

Regularized linear regression

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

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

→ Amacımız regularized cost function'ı minimize etmek!

→ Bunun için Gradient Descent kullanıyordu

Gradient descent

Θ_0 $\Theta_1, \Theta_2, \dots, \Theta_n$

Repeat {

Θ_0 için ayrı yazdık → $$\theta_0 := \theta_0 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_0^{(i)} \qquad \frac{\partial}{\partial\Theta_0}J(\Theta)$$

kalanlar için ayrı!

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

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

}

same!

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

$1 - \alpha\frac{\lambda}{m} < 1$ 0.99 $\Theta_j \times 0.99$ Θ_j^2

Regularization etkisi ile Θ_J biraz küçülür $1 - \alpha\frac{\lambda}{m}$ genelde 1'den biraz küçüktür 0.99 gibi, α ve $\lambda > 0$ ve m'de > 0.

Son form gösteriyor ki regularization ile yapılan aslında her iterasyonda, Θ'yı 0.99 gibi bir rakamla çarpıp shrink etmek. Gerisi tamamen aynı!

Min Θ'ları bulmak için 2. algorithm Normal Equation idi. X ve y yi bulduktan sonra min $\Theta = (X^T . X)^{-1} X^T . y$ olarak bulunabilirdi

Normal equation

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

$\rightarrow \min_\theta J(\theta)$ $\qquad \frac{\partial}{\partial \theta_j} J(\theta) \overset{set}{=} 0$

$$\rightarrow \Theta = \Big(X^T X + \lambda \underbrace{\begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 1 & & \\ \vdots & & 1 & \\ 0 & & & 1 \end{bmatrix}}_{(n+1)\times(n+1)} \Big)^{-1} X^T y$$

E.g. $n=2$ $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

→ Kırmızı önceden uygulanan algorithm idi!
→ Mavi yeni eklenen term. Regularization ile gelen term. (Yeni cost function'dan bulunabilir $\frac{\partial}{\partial \theta_j} J(\theta) = 0$ yapan Θ'dan geliyor!

Non-invertibility (optional/advanced).

Θ $m \leq n$ ise $X^T X$ non invertible!

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

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

If $\lambda > 0$,

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

Andrew Ng

Regularization ile bu non-invertibility problemi çözülür!

Andrew Ng

Machine Learning

Regularization

Regularized logistic regression

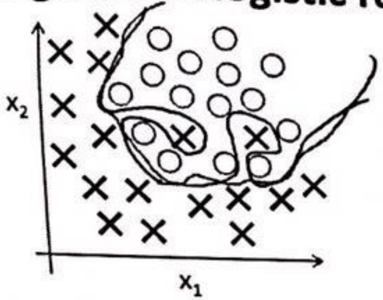
Logistic Regression modelinde $J(\Theta)$'yı minimize etmek için 2 farklı algorithmden bahsetmiştik.

1) Gradient Descent
2) Advanced Optimization Methods

Şimdi bunları Regularization ile adapt edeceğiz!

Bu overfitting'e neden olabilir.

Regularized logistic regression.



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

Cost function:

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

$\boxed{\theta_1, \theta_2, \dots, \theta_n}$

Andrew Ng

Gradient descent

Repeat {

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

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

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

$\theta_1 \dots \theta_n$

}

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

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

cosmetically same

Andrew Ng

Biraz önceki ile aynı şey çıkar!

Sadece $h_\theta(x)$ farklı!

Hypothesis is different!

* Eski J(θ)'ya yeni terimi ekleriz olay bu!
böylece hipotez smoothened ve overfitting'den kurtulunabilir.

Advanced optimization

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

```
function [jVal, gradient] = costFunction(theta)
    jVal = [ code to compute J(θ) ];
```

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

```
    gradient(1) = [ code to compute ∂/∂θ0 J(θ) ];
```

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

```
    gradient(2) = [ code to compute ∂/∂θ1 J(θ) ];
```

$$\left(\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_1^{(i)}\right) - \frac{\lambda}{m}\theta_1$$

```
    gradient(3) = [ code to compute ∂/∂θ2 J(θ) ];
```

$$\left(\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_2^{(i)}\right) - \frac{\lambda}{m}\theta_2$$

```
    ⋮
    gradient(n+1) = [ code to compute ∂/∂θn J(θ) ];
```

J(θ)

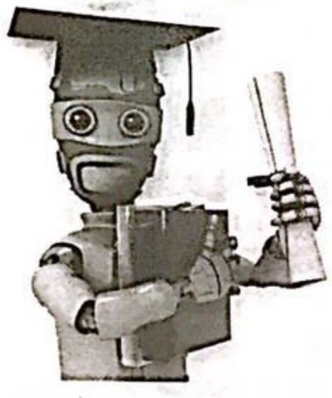
Kod aynı yalnızca J(θ) değişiyor buna bağlı gradients'de değişiyor

Andrew Ng

Sonra da yine bu alg. ile logistic regression için regularized parameters? bulunur ve overfitting olmadan J(θ)'yı min yapan θ'lar elde edilir 6

Linear ve Logistic regression works well for many machine learning problems but when you apply them to certain machine learning applications, they can run into a problem called OVERFITTING

Overfitting performansı düşürür.



Machine Learning

→ Overfitting'in çözümü: Regularization

Regularization

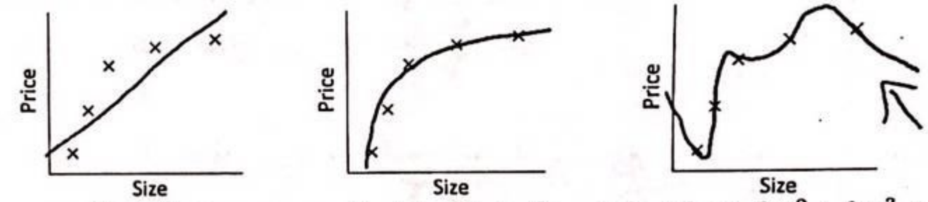
The problem of overfitting

→ Overfitting Logistic Regression için de gerçekleşebilir.

what is Overfitting?

(?) 2. ve 3. Linear Regression olmuyor sanki?

Example: Linear regression (housing prices)



→ $\theta_0 + \theta_1 x$ "Underfit" "High bias"

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

→ $\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$ "Overfit" "High variance" has

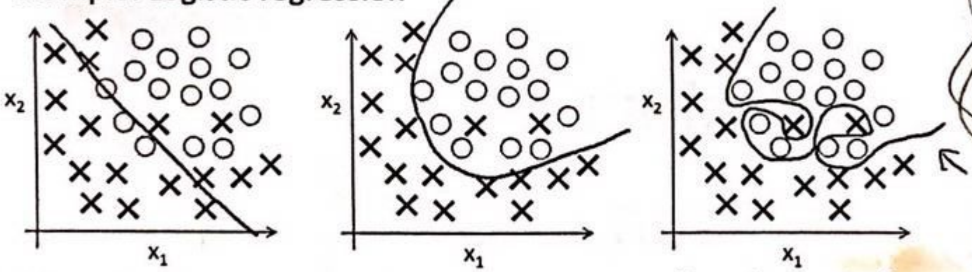
→ $J(\theta) \approx 0$ olsa dahi, yeni previously unseen data'yı tahmin edemez!!! Hypothesis

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).

Andrew Ng

(?) Problem feature sayısından ziyade $h_\theta(x)$'in derecesinin yani x^3, x^4 lerin işin içine girmesi değil mi?

Example: Logistic regression



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

(g = sigmoid function)

"Underfit"

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

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

"Over-fit"

A lot of high order polynomials !

Andrew Ng

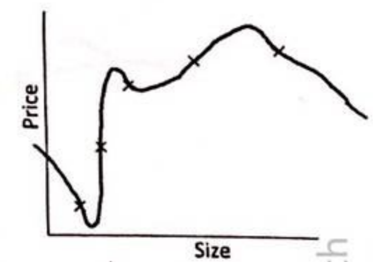
(?) Logistic Regression'da tek bir hata bile sonsuz cost getirmiyor mu? O zaman gradient descent $J(\theta)$'yı nasıl minimize edecek, ilk ve 2. algorithmler için sonsuz konuşursak ∞ getirmiyor. Sonsuz olması için $h(x)$'in 1 vermesi lazım. Yenemezb

Addressing 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 in neighborhood
$x_6 =$ kitchen size
$\vdots$
x_{100}

→ Çok fazla feature varsa ve çok az training data var ise OVERFITTING bir problem olabilir.

Böylece mesela yazı tanımada tek bir yazıyı tanıyor cunku overfitted.



Tek bir feature varsa, plotting ile kaçıncı derece polinom kullanacağıma karar verebilirim, ama genelde feature sayısı çok daha fazla olur.

Andrew Ng