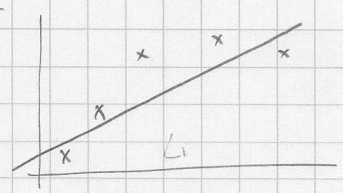
Regularization (of LinReg and LogReg)

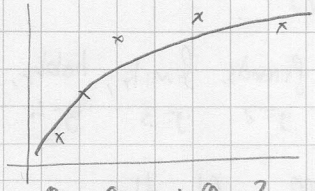
The problem of overfitting

Linear Regression



$\theta_0 + \theta_1 x$

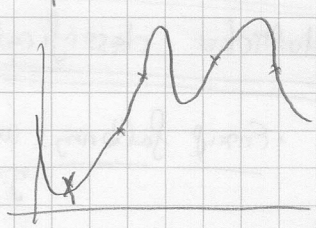
Underfitted or High Bias



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

"Just Right"

passes through all dataset points, but fails to predict outside these

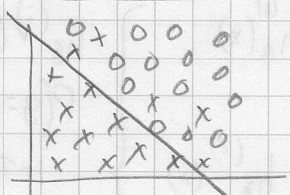


$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \dots$

Overfitted or High variance

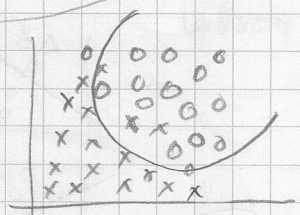
If we have too many features, the learned hypothesis may fit the training set very well ($J(\theta) \approx 0$) but fail to generalize to new examples

Same with Logistic Regression

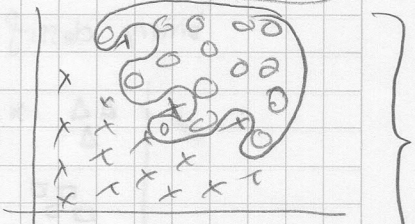


$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

underfitted



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



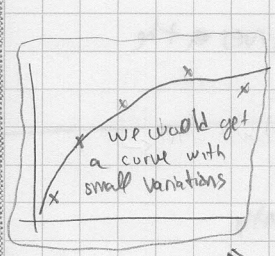
$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \dots)$

overfitted

How to address overfitting

We can:

- Reduce number of features
  - manually (error-prone)
  - model selection algorithms (they automatically select which features to use)
- Regularization
  - keep all features but reduce magnitude of θ_j
  - works well when we have a lot of features, each one contributing a little bit to y.



→ the idea is to force small values to θ_j:
- simpler hypothesis (simpler functions, smoother...) (difficult to intuitively understand it...)
- less prone to overfitting

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

→ regularization term

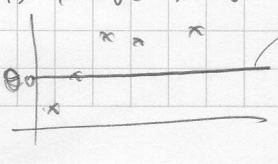
→ we don't penalize θ_0 (just a convention, in practice it would not change much)

∇ We want small θ_j,
so I guess that, since this is an optimization problem, we want to consider this in the cost function so the high θ_j solutions get penalized. This is what the regularization term does... ∇

→ adding a term with λ to all θ_j, we penalize those solutions that may have some θ_j large.

large value so when we minimize $J(\theta)$, the values of θ_j will be small (regularization parameter)

- if λ is very large $\Rightarrow \theta_j \approx 0$



Regularized Gradient Descent for Lin Reg

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

Repeat {

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

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

}

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

grouping all θ_j in 2nd line

$h_\theta(x) = \theta^T x$

Repeat {

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

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

}

→ same term as in not regularized gradient descent

number a little bit less than 1 (so we get small θ_j)

→ the small θ_j are also obtained because of the regularization term in the $J(\theta)$ to minimize

Regularized Normal Equation ... skipped

Regularized Gradient Descent for Log Reg

(same as Lin Reg)

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

Repeat {

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

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

}

Regularization for Advanced Optimization

How to compute the cost and the derivatives to fit advanced optimization algorithms.