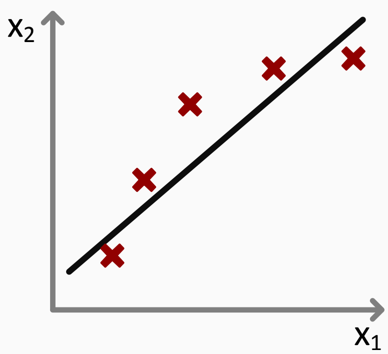
regularization $\boldsymbol{\lambda}$ optimization

solving the problem of overfitting

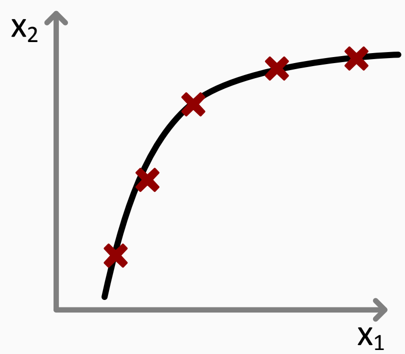
the problem of overfitting

linear regression:



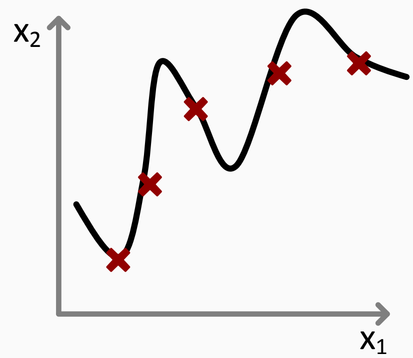
$\theta_0 + \theta_1 x$

"underfitted" or "highly biased"



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

"ideal"



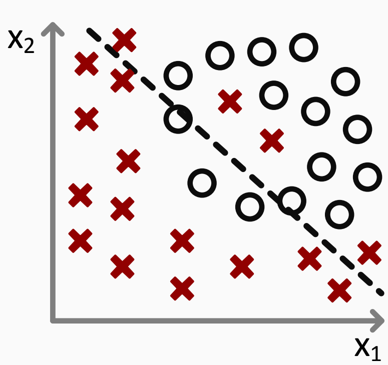
$\theta_0 + \theta_1 x + \theta_2 x^2 + \cdots$
$\theta_3 x^3 + \theta_4 x^4$

"overfitted" or "high variance"

overfitting:

too many features might cause the learned hypothesis to fit the training set very well ($J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 \approx 0$), but fail to generalize new examples (prediction)

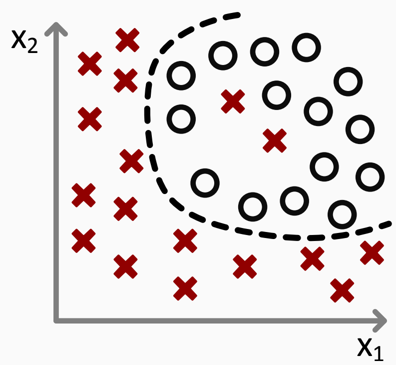
logistic regression:



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

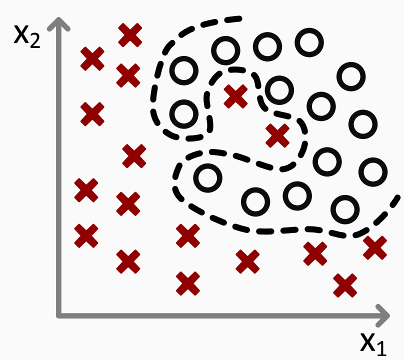
(g = sigmoid function)

"underfitted" or "highly biased"



$h_\theta(x) = g\begin{pmatrix} \theta_0 + \theta_1 x_1 + \cdots \\ \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 \end{pmatrix}$

"ideal"



$h_\theta(x) = g\begin{pmatrix} \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots \\ \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_4 x_2^2 + \cdots \end{pmatrix}$

"overfitted" or "high variance"

measures to reduce overfitting:

- reduce the number of features
  - manually select features to retain
  - model selection algorithm (principal component analysis)
- regularization
  - retain all features, but reduce the magnitude/ values of parameters θ_j
  - works well when there are many features with predictive value of y

Consider the medical diagnosis problem of classifying tumors as malignant or benign. If a hypothesis $h_\theta(x)$ has overfit the training set, it means that:

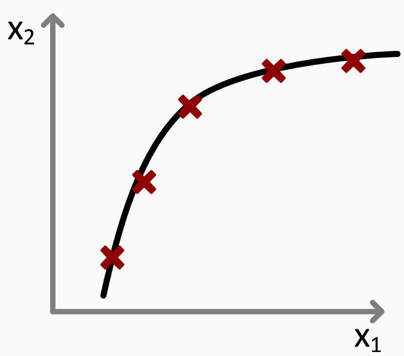
- It makes accurate predictions for examples in the training set and generalizes well to make accurate predictions on new, previously unseen examples.
- It does not make accurate predictions for examples in the training set, but it does generalize well to make accurate predictions on new, previously unseen examples.
- It makes accurate predictions for examples in the training set, but it does not generalize well to make accurate predictions on new, previously unseen examples.

Correct Response

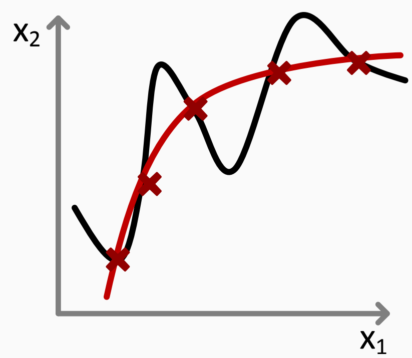
- It does not make accurate predictions for examples in the training set and does not generalize well to make accurate predictions on new, previously unseen examples.

cost function

intuition of the cost function:



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



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

previous examples illustrated how an over parameterized function is prone to overfitting the data as seen in the illustrations above. the quadratic function fits the data ideally; while the high order polynomial fits the training set well but will fail to generalize to new examples.

consider penalizing the parameters θ_3, θ_4 by making their weights insignificant:

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

assigning large values to θ_3, θ_4 in the cost function will force the values of $\theta_3, \theta_4 \approx 0$

this essentially removes the effect of higher order polynomials from above.

regularization

the general idea behind regularization is if there are small values for the parameters $\theta_1, \theta_2, \dots, \theta_n$ will lead to a model with:

a more simple hypothesis with smoother functions
a model that is less prone to overfitting

a example with housing price predictions:

features: $x_1, x_2, \dots, x_{100}$

parameters: $\theta_0, \theta_1, \theta_2, \dots, \theta_{100}$

it is not directly known which parameters will have an overbearing effect on the cost functions in a negative facet. Therefore, the cost function will be extended with a regularization parameter to compensate:

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

regularization parameter
regularization term

the additional regularization term will penalize all parameters equally; by convention, the indexing begins at $i = 1$ because θ_0 will not be penalized for being large.

lambda λ controls a cost function trade off of two goals as follows:

1. fit the training data well → captured by the least squares term
2. maintain small parameters → captured by the regularization term

In regularized linear regression, we choose θ to minimize:

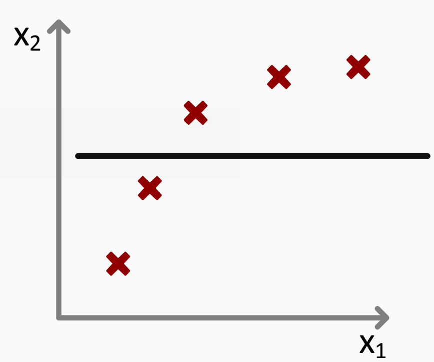
$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2\right]$$

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

- Algorithm works fine; setting λ to be very large can't hurt it.
- Algorithm fails to eliminate overfitting.
- Algorithm results in underfitting (fails to fit even the training set).

Correct Response

- Gradient descent will fail to converge.



regularized linear regression

the optimization objective for regularized linear regression:

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

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

gradient descent

repeat until convergence {

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

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

(update θ_j for $j = \cancel{0}, 1, 2, 3 \ldots, n$ simultaneously)

}

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

Suppose you are doing gradient descent on a training set of $m > 0$ examples, using a fairly small learning rate $\alpha > 0$ and some regularization parameter $\lambda > 0$. Consider the update rule:

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

Which of the following statements about the term $(1 - \alpha\frac{\lambda}{m})$ must be true?

- $1 - \alpha\frac{\lambda}{m} > 1$
- $1 - \alpha\frac{\lambda}{m} = 1$
- $1 - \alpha\frac{\lambda}{m} < 1$

  Correct Response

- None of the above.

normal equation

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

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

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

where the identity matrix $\in \mathbb{R}^{(n+1)\times(n+1)}$

noninvertibility

suppose $m \le n$ (examples less than or equal to the features)

$$\theta = \underbrace{(X^T X)}^{-1} X^T y$$

the matrix will not be invertible or degenerate

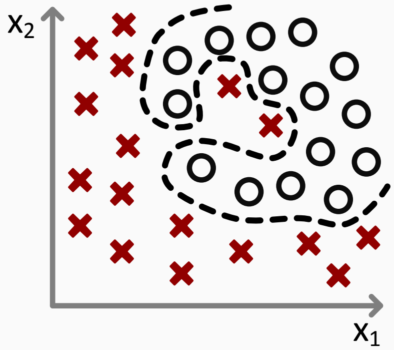
regularization does not have the same issue

if $\lambda > 0$,

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

The matrix is invertible

regularized logistic regression

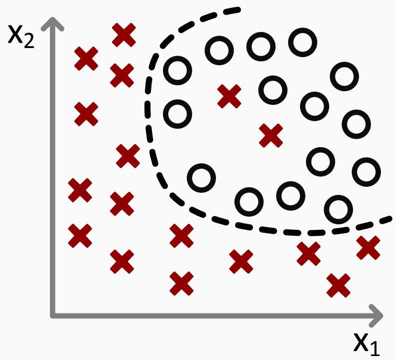


over parameterizing the sigmoid function can lead to equally overfitted models that do not generalize well:

$$h_\theta(x) = g\begin{pmatrix}\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots \\ \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_4 x_2^2 + \cdots\end{pmatrix}$$

logistic regression cost function:

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



by adding a regularization term, the model can penalize all parameters $\theta_1, \theta_2, \ldots, \theta_n$

$$+\frac{\lambda}{2m}\sum_{i=1}^{u} \theta_j^2$$

the regularized implementation is as follows:

gradient descent

repeat until convergence {

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

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

(update θ_j for $j = \cancel{0}, 1, 2, 3 \ldots, n$ simultaneously)

}

although the implementation for logistic regression is identical in appearance to linear regression; note that the hypothesis $h_\theta(x)$ is defined differently as follows:

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

When using regularized logistic regression, which of these is the best way to monitor whether gradient descent is working correctly?

- Plot $-\left[\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))\right]$ as a function of the number of iterations and make sure it's decreasing.
- Plot $-\left[\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))\right] - \frac{\lambda}{2m}\sum_{j=1}^{n} \theta_j^2$ as a function of the number of iterations and make sure it's decreasing.
- Plot $-\left[\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n} \theta_j^2$ as a function of the number of iterations and make sure it's decreasing.

Correct Response

advanced optimization

$$\texttt{theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \begin{matrix} \longleftarrow \texttt{theta(1)} \\ \longleftarrow \texttt{theta(2)} \\ \vdots \\ \longleftarrow \texttt{theta(n)} \end{matrix}$$

note the definition of theta being indexed at 1 in octave as opposed to 0 in the

```
function [jVal, gradient] = costFunction(theta)
```

`jVal =` [code to compute $J(\theta)$];

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

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

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

`gradient(2) =` [code to compute $\frac{\partial}{\partial\theta_1} J(\theta)$];

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

`gradient(2) =` [code to compute $\frac{\partial}{\partial\theta_1} J(\theta)$];

$$\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_1^{(i)} + \frac{\lambda}{m}\theta_2$$

⋮

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

note the lack of a regularization term in the code required to compute `gradient(1)`; the parameter θ_0 is not regularized as a form of convention. The parameter θ_0 is in reference to variable `gradient(1)` in octave because indexing in octave begins at 0.