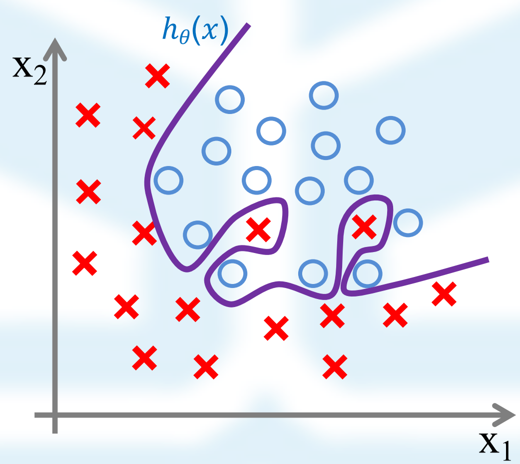
به نام او، به یاد او، برای او

Machine Learning

Regularization

The problem of overfitting



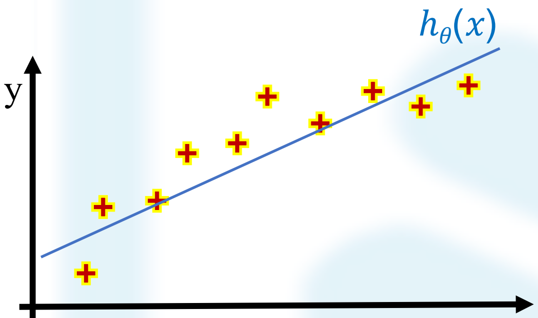
Dr. *Ali Valinejad*

valinejad.ir
valinejad@umz.ac.ir

Outlines:

- Which model is a good model?
- Definition of a good model
- Underfitting?
- Overfitting
- Regularization
- Regularized linear regression
  - Normal equation
  - Gradient Decent
- Regularized logistic regression

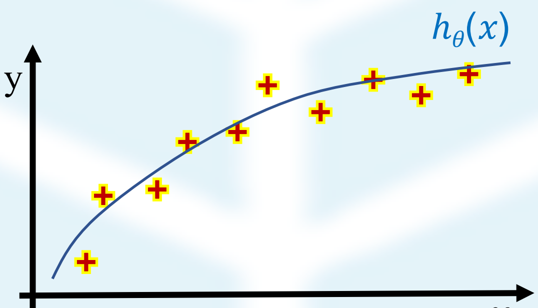
Which model is a good model?



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

Fit a linear function

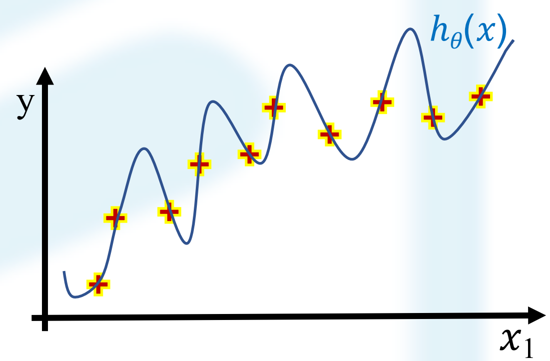
Underfitting
(High Bias)



$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2$$

Fit a quadratic function

Work Well



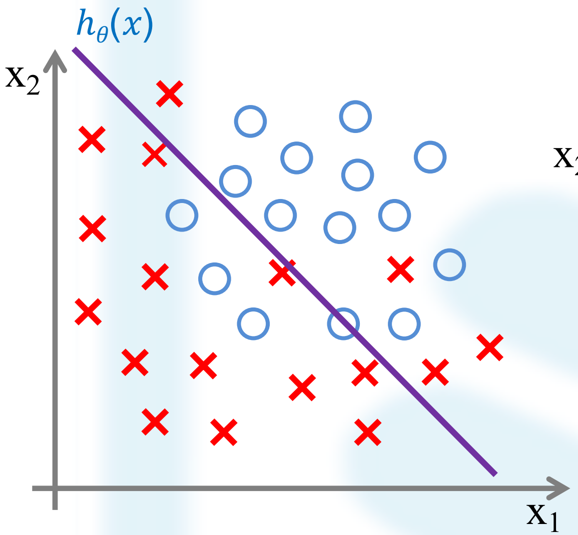
$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \cdots + \theta_{10} x_1^{10}$$

Fit a 10th order polynomial

Overfitting
(High variance)

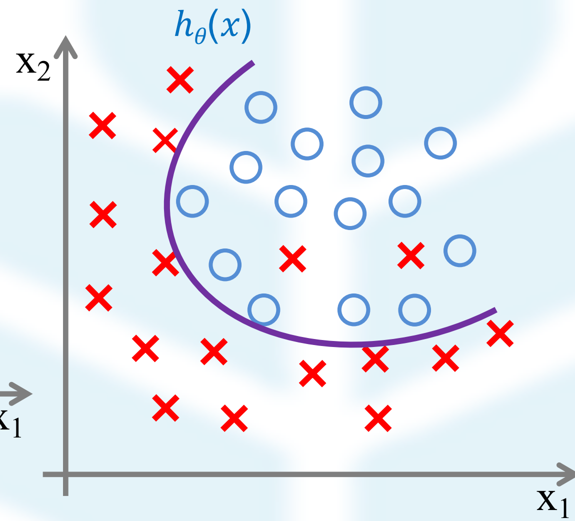
$$h_\theta(x) = \theta_0 + \theta_1 x_1$$

Which model is a good model?



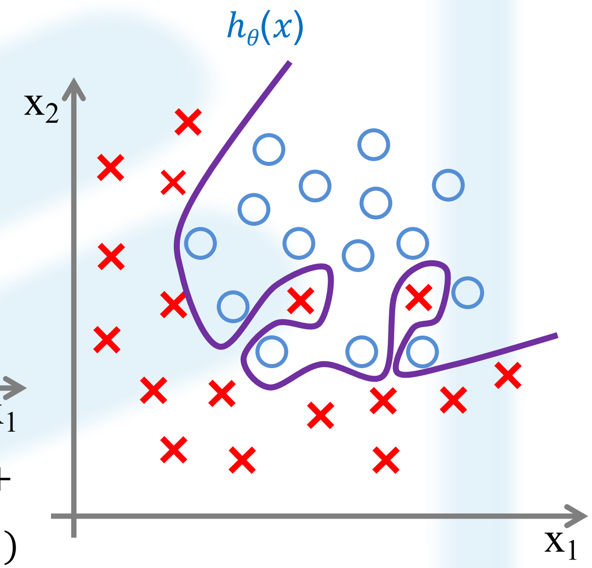
$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

Underfitting
(High Bias)



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

Work Well



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

Overfitting
(High variance)

$$g(z) = \frac{1}{1 + e^{-z}}$$

Definition of a good model

The definition of a 'good' model with *least training error*:

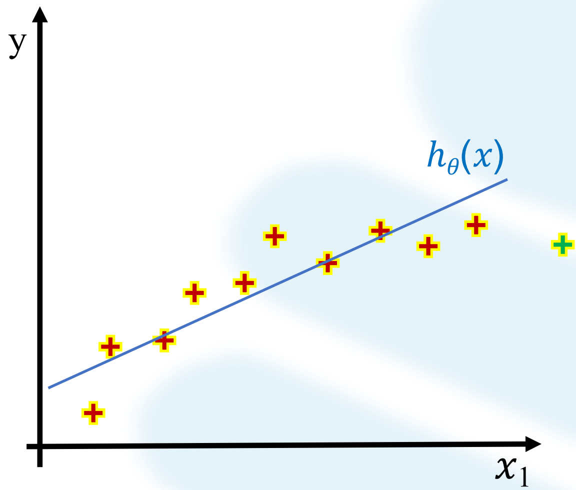
Step 1. define a hypothesis function $h_\theta(x)$ (i.e. model).

Step 2. define a Cost Function $J(\boldsymbol{\theta})$ for error measuring.

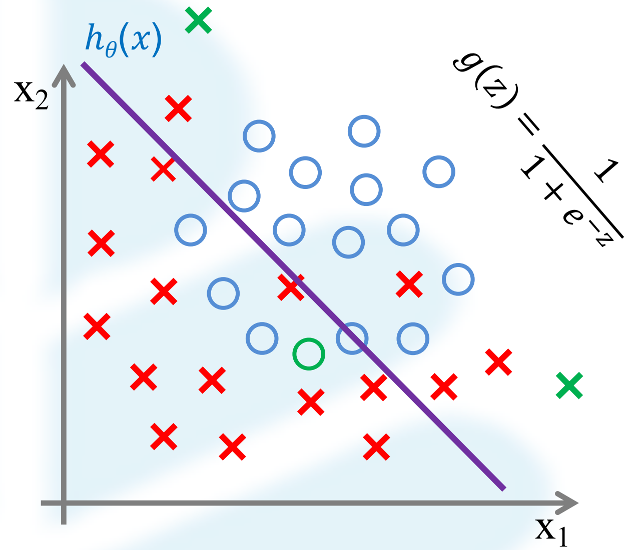
Step 3. learn parameters of $h_\theta(x)$ by minimizing $J(\boldsymbol{\theta})$.

Underfitting

Underfitting



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

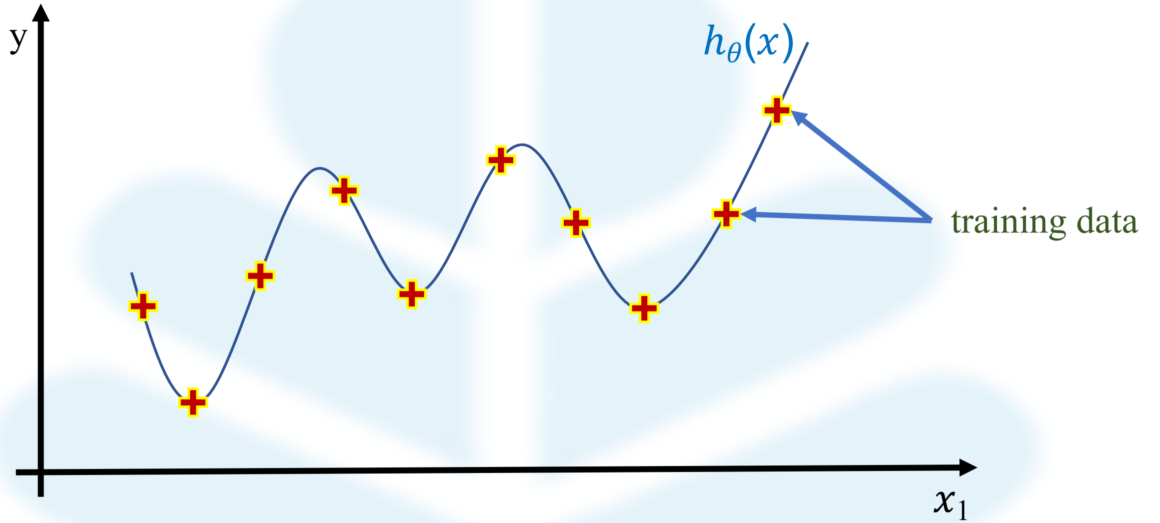


$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

- ❖ This model has huge training error,
- ❖ When new data (green point) comes in, this model could has huge *prediction* error.

Overfitting

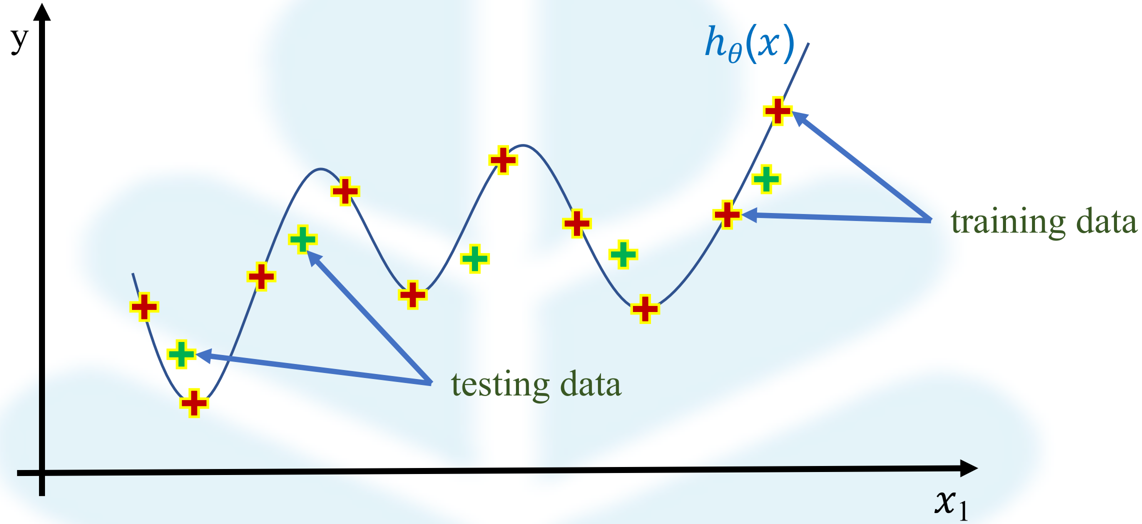
Is this is a good model?



- This model has zero training error,
- This model fits training data perfectly,
- The predict label of all training data is equal to their corresponding truth label.

Is this is a good model?

How come the performance of a good model is terrible?

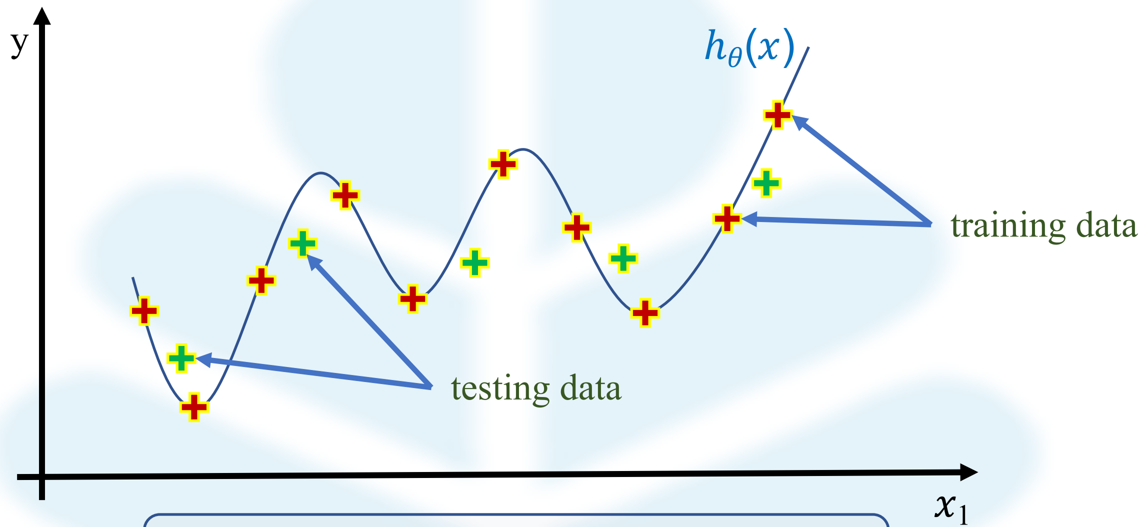


- ❖ The model has little *training error*.
- ❖ When new data (green point) comes in, this model has huge *prediction* error.

The learned model works well for training data but ***terrible*** for testing data (unknown data).

A truly good model must have both little training error and little prediction error.

Overfitting



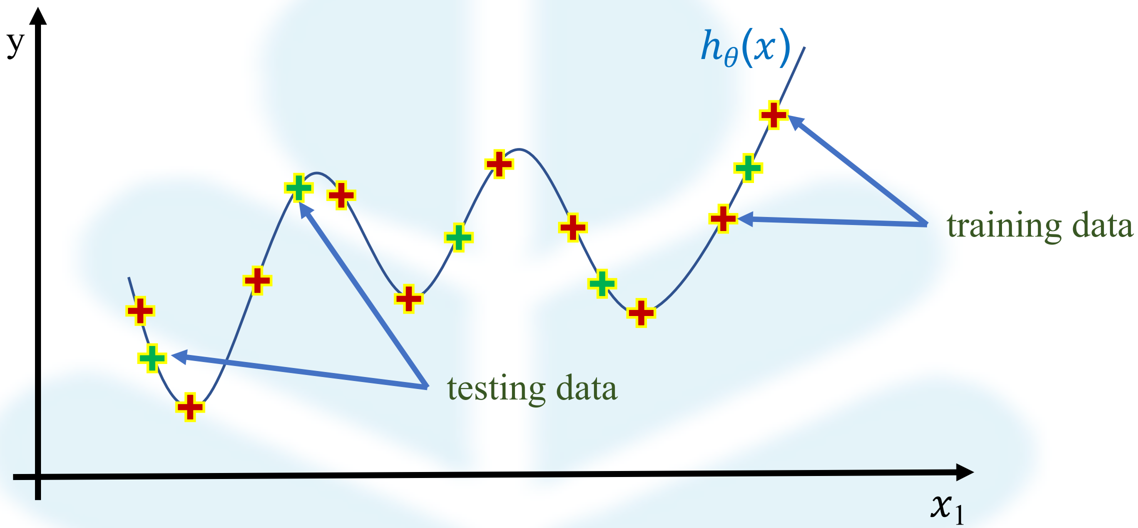
The problem here is called *Overfitting*.

Overfitting: If we have ***too many features***, the learned hypothesis may fit the training set very well, but fail to generalize to new examples.

Algorithm has high variance:

- **Variance**, in the context of Machine Learning, is a type of error that occurs due to a model's sensitivity to small fluctuations in the training set.
- Thise means that hypothesis can basically fit any training data.
- High variance would cause an algorithm to model the noise in the training set.

Overfitting



If we have a whole dataset that captured *all possibilities* of the problem, we don't need to worry about overfitting.

When new data comes in, it shall fall into one possibility, hence, the model can predict it perfectly.

A truly good model must have both little training error and little prediction error.

Overfitting

Addressing overfitting:

- ❖ Reduce number of features
  - Manually select which features to keep.
  - Model selection algorithm.
- ❖ Regularization
  - 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.

Regularization

Regularization

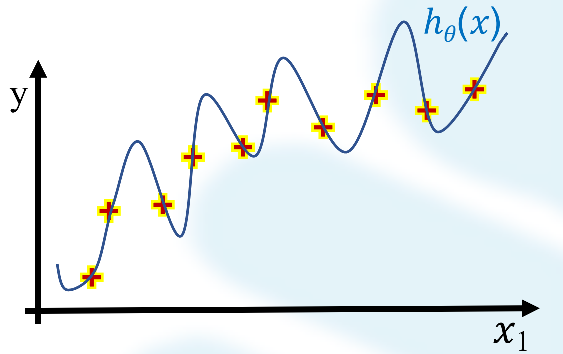
When ***overfitting*** occurs, we get an over complex model with too many features. One way to avoid it is to apply ***Regularization*** and then we can get a better model with proper features.

Regularization is a technique applied to Cost Function $J(\theta)$ in order to avoid Overfitting.

The ***core idea*** in ***Regularization*** is to keep more important features and ignore unimportant ones. The importance of feature is measured by the value of its parameter θ_j.

Regularization

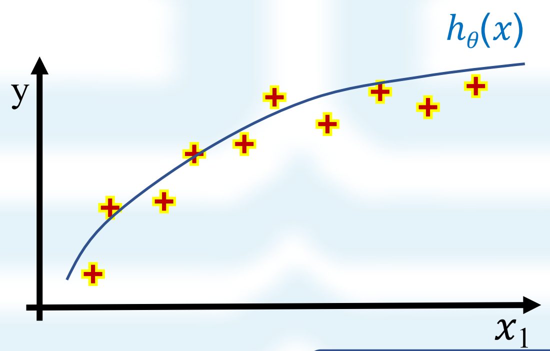
Intuition



$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \cdots + \theta_{10} x_1^{10}$

Suppose we penalize and make $\theta_3, \theta_4, \ldots, \theta_{10}$ really small.

$$\min_{\theta} \; \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + 1000\theta_3^2 + 1000\theta_4^2 + \cdots + 1000\theta_{10}^2\right]$$



$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \underbrace{\boxed{\theta_3 x_1^3 + \theta_4 x_1^4 \ldots + \theta_{10} x_1^{10}}}_{\approx\, 0}$$

Regularized linear regression

Regularized linear regression

- **Standard linear regression:**

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

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

- **Regularized linear regression:**

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

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

In linear regression, we modify its cost function by adding regularization term.
The value of θ_j is controlled by regularization parameter λ.
Note that m is the number of data and n is the number of features (parameters).

Regularized linear regression

Normal equations to minimize $J(\theta)$

Regularized linear regression

❖ **Normal equations to minimize $J(\theta)$**

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

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

Given a training set, define the ***design matrix*** *X*

$$X := \begin{bmatrix} - & (x^{(1)})^T & - \\ - & (x^{(2)})^T & - \\ & \vdots & \\ - & (x^{(m)})^T & - \end{bmatrix} = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_n^{(1)} \\ 1 & x_1^{(2)} & \dots & x_n^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(m)} & \dots & x_n^{(m)} \end{bmatrix} \in \mathbb{R}^{m\times(n+1)}$$

Regularized linear regression

❖ **Normal equations to minimize $J(\theta)$**

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

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

$if\ \lambda > 0:$

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

Regularized linear regression

Batch Gradient descent to minimize $J(\theta)$

Regularized linear regression

❖ **Batch Gradient descent to minimize $J(\theta)$**

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

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

Repeat {

$$\theta := \theta - \alpha \nabla_\theta J(\theta)$$

}

α: Learning rate

Regularized linear regression

❖ **Batch Gradient descent to minimize $J(\theta)$**

Repeat {

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

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

(simultaneously update for every $j = 0,1, \dots, n$)

}

Regularized logistic regression

Regularized linear regression

❖ Standard logistic regression:

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

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

❖ Regularized logistic regression:

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

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

In logistic regression, we modify its cost function by adding regularization term.
The value of θ_j is controlled by regularization parameter $\boldsymbol{\lambda}$.
Note that m is the number of data and n is the number of features (parameters).

Regularized logistic regression

❖ **Batch Gradient descent to minimize $J(\theta)$**

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

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

Repeat {

$$\boldsymbol{\theta} := \boldsymbol{\theta} - \alpha \nabla_\theta J(\boldsymbol{\theta})$$

}

α: Learning rate

Regularized logistic regression

❖ **Batch Gradient descent to minimize $J(\theta)$**

Repeat {

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

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

(simultaneously update for every $j = 0,1, \dots, n$)

}

Regularized

Two advantages of using regularization:

- The ***prediction error*** of the regularized model is ***lesser***, that is, it works well in testing data.
- The regularization model is ***simpler*** since it has less features (parameters).

References

1- https://www.geeksforgeeks.org

2- Andrew Ng, https://www.coursera.org/learn/machine-learning

3- https://medium.com/@qempsil0914/courseras-machine-learning-notes-week3-overfitting-and-regularization-partii-3e3f3f36a287