Logistic Regression Part 2

Dr Mehrdad Ghaziasgar

Senior Lecturer in Computer Science
Department of Computer Science
University of the Western Cape
mghaziasgar@uwc.ac.za

UNIVERSITY *of the* WESTERN CAPE

Content

Content – Part 2

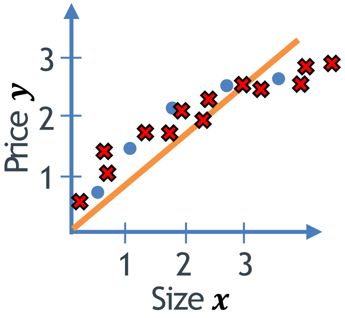
Overfitting and Underfitting

Introduction

Introduction

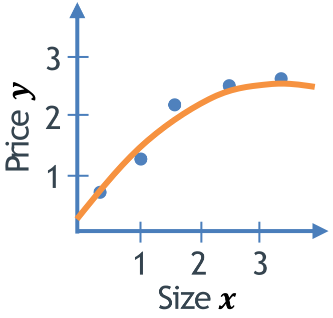
- Underfitting and overfitting are problems that can arise that cause a model (either regression or classification) to perform very poorly on test (new) examples
- We'll describe these problems and then describe how we can mitigate them using a technique called "Regularization"

Introduction - Overfitting and Underfitting in Linear Regression



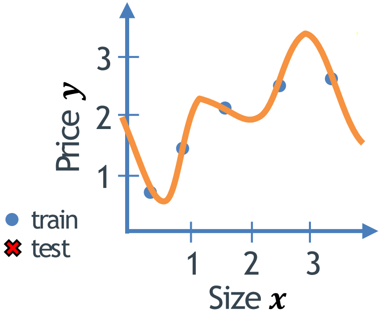
$h_\theta(x) = \theta_0 + \theta_1 x$

Underfitting
“High Bias”



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

“Just right”

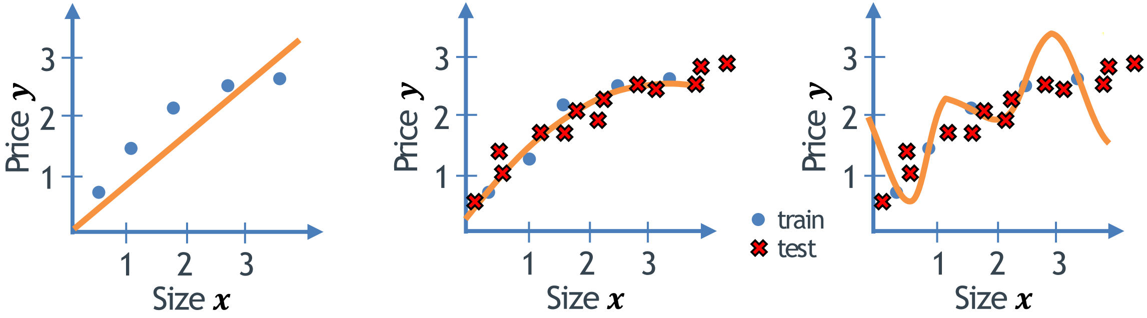


$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

Overfitting
“High Variance”

- Underfitting: The model $h_\theta(x)$ is too simple (too few features) and doesn’t fit the training examples very well or even at all (so $J(\theta)$ is very large) and so the model will also not reflect reality and doesn’t fit new (test) examples well either
- The model has high “bias” i.e. it has a biased (rigid, inflexible) nature

Introduction - Overfitting and Underfitting in Linear Regression



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

Underfitting
"High Bias"

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

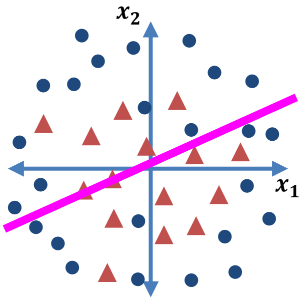
"Just right"

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

Overfitting
"High Variance"

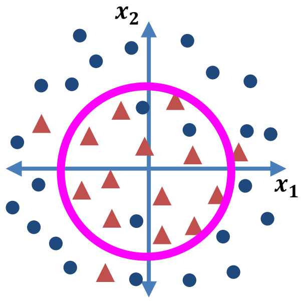
- Overfitting: The model $h_\theta(x)$ is too complex (too many features) and fits the training examples very well or even perfectly (so $J(\theta) \approx 0$) but the model doesn't reflect reality and doesn't fit the test examples very well (or at all)
- The model has "high variance" i.e. it has a "too flexible" / "too variant" nature

Introduction - Overfitting and Underfitting in Logistic Regression



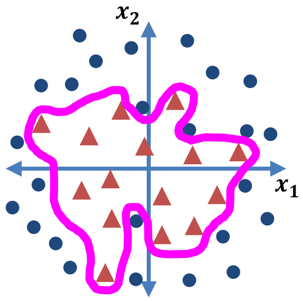
$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

Underfitting
"High Bias"



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

"Just right"



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

Overfitting
"High Variance"

Introduction – Addressing High Bias and Variance

- To address high bias:
    - Increase the number of features i.e. new types of features, high-order features, combination of features
- To address high variance, there are two main methods:
    1. Reduce the Features
        - Manually (pain-stakingly) select and remove features
        - Use train-cv-test sets to compare various models and select the best one
        - Works well when you have relatively fewer features and/or you have specific ideas about models to use/compare
    2. Apply Regularization
        - Use as complex of a model as you like (number of features, high-order terms, combination of features)
        - Apply regularization to reduce the impact of these features towards predicting $\boldsymbol{y}$ by adjusting/reducing all of the $\boldsymbol{\theta}$ parameters
        - Works well when you have a LOT of features and manual selection will be hard

Overfitting and Underfitting

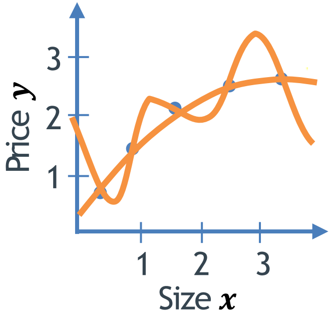
The Concept of Regularization

Regularization - Concept

- If we minimize $\boldsymbol{J(\theta)}$ with this hypothesis, we'll get a $\boldsymbol{h_\theta(x)}$ that overfits
- Suppose that we penalize $\boldsymbol{\theta_3}$ and $\boldsymbol{\theta_4}$ to make them really small by modifying the cost function as follows:

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \frac{1}{2m} \sum_{\boldsymbol{i}=1}^{\boldsymbol{m}} \left(\boldsymbol{h_\theta}\left(\boldsymbol{x}^{(i)}\right) - \boldsymbol{y}^{(i)}\right)^2 + 100\ \boldsymbol{\theta_3^2} + 100\ \boldsymbol{\theta_4^2}$$

- With everything else the same, the only way to reduce (and minimize) the cost in the same way as before is to ensure that $\boldsymbol{\theta_3}$ and $\boldsymbol{\theta_4}$ become really really small
  - Has the effect of setting $\boldsymbol{\theta_3}$ and $\boldsymbol{\theta_4}$ such that $+ 100\ \boldsymbol{\theta_3^2} + 100\ \boldsymbol{\theta_4^2}$ is set to 0 to get back the original cost i.e. setting $\boldsymbol{\theta_3}$ and $\boldsymbol{\theta_4}$ close to 0
  - Also the effect of almost excluding the third and fourth order terms in $\boldsymbol{h_\theta(x)}$



$$\boldsymbol{h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2} \\ + \underset{\approx 0}{\boldsymbol{\theta_3 x^3}} + \underset{\approx 0}{\boldsymbol{\theta_4 x^4}}$$

Regularization - Intuition

- Intuition 2: Hamburger shop:
  - As the manager of a hamburger joint, you're selling 10 200g hamburgers per day @ R10 per hamburger
  - The "cost" of a burger is:

$$\text{Cost} = P_{\text{Meat}} + P_{\text{Veggies}} + P_{\text{Roll}}$$
$$= 3.50 + 1.20 + 1.50 = \text{R}6.20$$

  - At this cost, the boss is making a handsome R2000 per day in profits
  - Life is good

Regularization - Intuition

- Intuition 2: Hamburger shop:
  - Suddenly, the centre announces that they are supporting “animal rights” or some weird thing like that
  - They are imposing an extra “cost” on every kg of meat you sell
  - They will charge you 1cent for every 1g of meat you sell (basically the weight of the meat)
  - The “cost” of a burger now changes:

$$\begin{aligned}\text{Cost} &= P_{\text{Meat}} + P_{\text{Veggies}} + P_{\text{Roll}} \\ &= 3.50 + 1.20 + 1.50\end{aligned}$$

  - Unfortunately, the boss says he’s not willing to accept that much less profit and orders you to reduce the new cost down to R6.50 **somehow.**
  - You’ve got to reduce your cost somehow to keep profits constant

  - The prices at which you’re buying meat, veggies and rolls are also still the same
  - What can you change to keep the cost the same as it was before?
    - Reducing the “weight” of the burger

Regularization - Intuition

- Intuition 2: Hamburger shop:
  - Here’s what needs to happen to get the new cost down from R8.20 to R6.50:

$$\text{Cost} = \text{P}_{\text{Meat}} + \text{P}_{\text{Veggies}} + \text{P}_{\text{Roll}} + \boldsymbol{\theta}_{\text{Meat}} \times 0.01$$

$$6.50 = 3.50 + 1.20 + 1.50 + \boldsymbol{\theta}_{\text{Meat}} \times 0.01$$

$$\boldsymbol{\theta}_{\text{Meat}} = \frac{6.50 - (3.50 + 1.20 + 1.50)}{0.01}$$

$$\boldsymbol{\theta}_{\text{Meat}} = \frac{6.50 - (3.50 + 1.20 + 1.50)}{0.01}$$

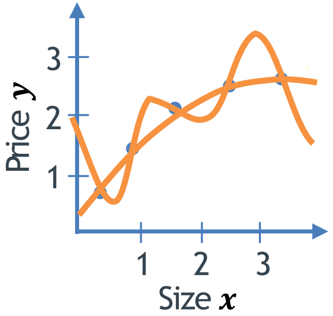
$$\boldsymbol{\theta}_{\text{Meat}} = 30\text{g of meat}$$

Regularization - Concept

- If we minimize $J(\theta)$ with this hypothesis, we'll get a $h_\theta(x)$ that overfits
- Suppose that we penalize θ_3 and θ_4 to make them really small by modifying the cost function as follows:

$$\min_{\theta} \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right)-y^{(i)}\right)^2 + 100\,\theta_3^2 + 100\,\theta_4^2$$

- With everything else the same, the only way to reduce (and minimize) the cost in the same way as before is to ensure that θ_3 and θ_4 become really really small
  - Has the effect of setting θ_3 and θ_4 such that $+\,100\,\theta_3^2 + 100\,\theta_4^2$ is set to 0 to get back the original cost i.e. setting θ_3 and θ_4 close to 0
  - Also the effect of almost excluding the third and fourth order terms in $h_\theta(x)$



$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \underset{\approx 0}{\theta_3} x^3 + \underset{\approx 0}{\theta_4} x^4$$

Regularization - Concept

- In the example we saw: we knew which terms were higher-order i.e. $\boldsymbol{\theta_3}$ and $\boldsymbol{\theta_4}$
- Adding those specific terms to the cost function reduced their weights
- In practice: we aren't always sure which features we need to regularize
    - We regularize ALL of the weights (except $\boldsymbol{\theta_0}$) equally
    - We add all of them to the cost function

$$J_{\mathbf{reg}}(\boldsymbol{\theta}) = J(\boldsymbol{\theta}) + \frac{\lambda}{2m}\sum_{j=1}^{n} \theta_j^2$$

- $J(\boldsymbol{\theta})$ is the normal cost function that we had before (either for linear or logistic regression)
- We don't regularize $\boldsymbol{\theta_0}$ so j starts from 1, not 0.

- The parameter λ is the regularization parameter which helps control how much to regularize:
    - Setting $\lambda = 0$ means that the whole regularization term falls away i.e. no regularization at all – potential **overfitting** – **high variance**
    - Setting $\lambda > 0$ means increasing amount of regularization by penalizing all $\boldsymbol{\theta_j}$ (except $\boldsymbol{\theta_0}$) at the same time – moving towards **underfitting** – **high bias**

Overfitting and Underfitting

Regularized Linear Regression

Regularized Linear Regression

- What we arrived at previously:

$$J_{\text{reg}}(\boldsymbol{\theta}) = \boldsymbol{J}(\boldsymbol{\theta}) + \frac{\lambda}{2m}\sum_{j=1}^{\boldsymbol{n}} \boldsymbol{\theta}_j^2$$

- So for linear regression:

$$J_{\text{reg}}(\boldsymbol{\theta}) = \frac{1}{2m}\sum_{\boldsymbol{i}=1}^{\boldsymbol{m}}\left(\boldsymbol{h}_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - \boldsymbol{y}^{(i)}\right)^2 + \frac{\lambda}{2m}\sum_{j=1}^{\boldsymbol{n}} \boldsymbol{\theta}_j^2$$

$$J_{\text{reg}}(\boldsymbol{\theta}) = \frac{1}{2m}\left[\sum_{\boldsymbol{i}=1}^{\boldsymbol{m}}\left(\boldsymbol{h}_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - \boldsymbol{y}^{(i)}\right)^2 + \lambda\sum_{j=1}^{\boldsymbol{n}} \boldsymbol{\theta}_j^2\right]$$

Regularized Linear Regression

- With the new cost function, we can simply apply gradient descent (or any other optimization/minimization technique) as before
- This involves making continuous updates to all $\boldsymbol{\theta_j}$ to minimize the cost

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

?

- We'll be updating $\boldsymbol{\theta_j}$ using the same update rules as before
- One thing has changed here: $\frac{\partial}{\partial\theta_j}J(\boldsymbol{\theta})$ because the $J(\boldsymbol{\theta})$ has changed

Repeat until convergence:

Update all $\boldsymbol{\theta_j}$ simultaneously:

$$\theta_j \leftarrow \theta_j - \alpha\boxed{\frac{\partial}{\partial\theta_j}J(\boldsymbol{\theta})}$$

Regularized Linear Regression

- For any parameter $\boldsymbol{\theta_j}$:

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

Repeat until convergence:

Update all $\boldsymbol{\theta}$ simultaneously:

$$\boldsymbol{\theta_0} \leftarrow \boldsymbol{\theta_0} - \boldsymbol{\alpha} \frac{1}{m} \sum_{i=1}^{m} \left(\boldsymbol{\theta}^{T} \boldsymbol{x}^{(i)} - \boldsymbol{y}^{(i)} \right) \cdot \boldsymbol{x}_{0}^{(i)}$$

$$\boldsymbol{\theta_j} \leftarrow \boldsymbol{\theta_j} - \boldsymbol{\alpha} \left[\frac{1}{m} \sum_{i=1}^{m} \left(\boldsymbol{\theta}^{T} \boldsymbol{x}^{(i)} - \boldsymbol{y}^{(i)} \right) \cdot \boldsymbol{x}_{1}^{(i)} + \frac{\lambda}{m} \boldsymbol{\theta_j} \right]$$

Regularized Linear Regression

- The update rule for $\boldsymbol{\theta_0}$ is exactly the same as before: we're not regularizing it
- The update rules for all other $\boldsymbol{\theta_j}$ are different; they now have an added term

$$\boldsymbol{\theta_j} \leftarrow \boldsymbol{\theta_j} - \boldsymbol{\alpha}\left[\frac{1}{m}\sum_{\boldsymbol{i}=1}^{\boldsymbol{m}}\left(\boldsymbol{\theta}^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{i})} - \boldsymbol{y}^{(\boldsymbol{i})}\right) \cdot \boldsymbol{x}_{\mathbf{1}}^{(\boldsymbol{i})} + \frac{\lambda}{m}\boldsymbol{\theta_j}\right]$$

- If we slightly restructure this update rule, something interesting emerges:

$$\boldsymbol{\theta_j} \leftarrow \boldsymbol{\theta_j} - \boldsymbol{\alpha}\frac{1}{m}\sum_{\boldsymbol{i}=1}^{\boldsymbol{m}}\left(\boldsymbol{\theta}^{\boldsymbol{T}}\boldsymbol{x}^{(i)} - \boldsymbol{y}^{(i)}\right) \cdot \boldsymbol{x}_{\mathbf{1}}^{(i)} - \boldsymbol{\alpha}\frac{\lambda}{m}\boldsymbol{\theta_j}$$

$$\boldsymbol{\theta_j} \leftarrow \boldsymbol{\theta_j} - \boldsymbol{\alpha}\frac{\lambda}{m}\boldsymbol{\theta_j} - \boldsymbol{\alpha}\frac{1}{m}\sum_{\boldsymbol{i}=1}^{\boldsymbol{m}}\left(\boldsymbol{\theta}^{\boldsymbol{T}}\boldsymbol{x}^{(i)} - \boldsymbol{y}^{(i)}\right) \cdot \boldsymbol{x}_{\mathbf{1}}^{(i)}$$

$$\boldsymbol{\theta_j} \leftarrow \left(\mathbf{1} - \boldsymbol{\alpha}\frac{\lambda}{m}\right)\boldsymbol{\theta_j} - \boldsymbol{\alpha}\frac{1}{m}\sum_{\boldsymbol{i}=1}^{\boldsymbol{m}}\left(\boldsymbol{\theta}^{\boldsymbol{T}}\boldsymbol{x}^{(i)} - \boldsymbol{y}^{(i)}\right) \cdot \boldsymbol{x}_{\mathbf{1}}^{(i)}$$

- Given that m will usually be larger than $\boldsymbol{\alpha}\lambda$, the term $0 < \boldsymbol{\alpha}\frac{\lambda}{\boldsymbol{m}} < 1$ e.g 0.01
- Therefore, also $0 < \left(\mathbf{1} - \boldsymbol{\alpha}\frac{\lambda}{m}\right) < 1$ e.g. 0.99
- Multiplying this by $\boldsymbol{\theta_j}$ has the effect of shrinking by e.g. 1% it before the update

Regularized Linear Regression – Using sklearn

- Now that we know the details, we can use sklearn to fit a regularized model as follows:

```
from sklearn.linear_model import Ridge

model = Ridge(alpha= 10 )

model.fit(X,y)
```

$\lambda = 10$

- Note that X should **not** contain the extra column of 1s for feature $\boldsymbol{x_0}$.
- The Ridge class has other parameters: find out what they are/do

Normal Equation With Regularization

- Given a feature matrix $\boldsymbol{X}$, and the corresponding output matrix $\boldsymbol{y}$
- The following equation solves for $\boldsymbol{\theta}$ that best fits the data with regularization:

$$\boldsymbol{\theta} = \left(\boldsymbol{X}^T \boldsymbol{X} \qquad\qquad\qquad\qquad \right)^{-1} \boldsymbol{X}^T \boldsymbol{y}$$

- Where
  - the matrix added is an $(\boldsymbol{n}+1) \times (\boldsymbol{n}+1)$ and
  - has a diagonal of 1s (apart from the top-left entry) and 0s everywhere else
- E.g.

$$\boldsymbol{X} = \begin{array}{c} \\ \\ \end{array}\begin{array}{ccccc} & \text{Size} & \text{\#Rms} & \text{Age} & \text{\#Grgs} \\ \end{array}$$

$$\boldsymbol{X} = \begin{bmatrix} 1 & 460 & 4 & 12 & 2 \\ 1 & 70 & 1 & 5 & 0 \\ 1 & 155 & 3 & 8 & 2 \\ 1 & 429 & 6 & 10 & 3 \end{bmatrix} \qquad \boldsymbol{y} = \begin{bmatrix} 6639 \\ 1681 \\ 3969 \\ 5095 \end{bmatrix}$$

with column labels Size, #Rms, Age, #Grgs over the columns of $\boldsymbol{X}$ and Price over $\boldsymbol{y}$.

Overfitting and Underfitting

Regularized Logistic Regression

Regularized Logistic Regression

- What we arrived at previously:

$$J_{\text{reg}}(\boldsymbol{\theta}) = \boldsymbol{J}(\boldsymbol{\theta}) + \frac{\lambda}{2m}\sum_{j=1}^{\boldsymbol{n}} \boldsymbol{\theta}_j^2$$

- So for linear regression:

$$J_{\text{reg}}(\boldsymbol{\theta}) = -\frac{1}{m}\sum_{i=1}^{\boldsymbol{m}}\left[\boldsymbol{y}^{(i)}\ln\left(\boldsymbol{h}_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right)\right) + (1-\boldsymbol{y}^{(i)})\ln\left(1-\boldsymbol{h}_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right)\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{\boldsymbol{n}} \boldsymbol{\theta}_j^2$$

- Where:

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

Regularized Logistic Regression

- With the new cost function, we can simply apply gradient descent (or any other optimization/minimization technique) as before
- This involves making continuous updates to all $\boldsymbol{\theta_j}$ to minimize the cost

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

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

?

- We'll be updating $\boldsymbol{\theta_j}$ using the same update rules as before

- One thing has changed here: $\frac{\partial}{\partial\boldsymbol{\theta_j}}J(\boldsymbol{\theta})$ because the $J(\boldsymbol{\theta})$ has changed

Repeat until convergence:

Update all $\boldsymbol{\theta_j}$ simultaneously:

$$\boldsymbol{\theta_j} \leftarrow \boldsymbol{\theta_j} - \boldsymbol{\alpha}\boxed{\frac{\boldsymbol{\partial}}{\boldsymbol{\partial\theta_j}}J(\boldsymbol{\theta})}$$

Regularized Logistic Regression

- For any parameter $\boldsymbol{\theta_j}$:

$$\frac{\partial}{\partial \boldsymbol{\theta_j}} J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{\boldsymbol{i}=1}^{\boldsymbol{m}} \left(\boldsymbol{h_\theta}\left(\boldsymbol{x}^{(\boldsymbol{i})}\right) - \boldsymbol{y}^{(\boldsymbol{i})} \right) \boldsymbol{x}_j^{(\boldsymbol{i})} + \frac{\lambda}{m} \boldsymbol{\theta_j}$$

Repeat until convergence:

Update all $\boldsymbol{\theta}$ simultaneously:

$$\boldsymbol{\theta_0} \leftarrow \boldsymbol{\theta_0} - \boldsymbol{\alpha} \frac{1}{m} \sum_{\boldsymbol{i}=1}^{\boldsymbol{m}} \left(\boldsymbol{h_\theta}\left(\boldsymbol{x}^{(\boldsymbol{i})}\right) - \boldsymbol{y}^{(\boldsymbol{i})} \right) \cdot \boldsymbol{x}_{\boldsymbol{0}}^{(\boldsymbol{i})}$$

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

- Algorithm appears to be identical to gradient descent for regularized linear regression
  - It is NOT! The hypothesis here is that of logistic regression

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

Regularized Logistic Regression Using Adv. Optim. Algorithms

- Implemented in SciPy's optimize library:

```
from scipy.optimize import minimize

theta = np.zeros(X.shape[1]) #Initialize all thetas to zeros
result = minimize(costFunc, theta, args=(X,y), method='BFGS', jac=gradientFunc,
options={'maxiter' : 400, 'disp': True})

print(result.x)
```

- costFunc is a function that returns the cost as per:

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

- gradientFunc is a function that returns a list/vector of the gradients of all the $\boldsymbol{\theta_j}$ as per:

$$\frac{\partial}{\partial\boldsymbol{\theta_j}}J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m}\left(\boldsymbol{h_\theta}\left(\boldsymbol{x}^{(i)}\right)-\boldsymbol{y}^{(i)}\right)\boldsymbol{x}_j^{(i)} + \frac{\lambda}{m}\boldsymbol{\theta_j}$$

Regularized Logistic Regression Using Adv. Optim. Algorithms

- Implemented in SciPy’s optimize library:

```
from scipy.optimize import minimize

theta = np.zeros(X.shape[1]) #Initialize all thetas to zeros
result = minimize(costFunc, theta, args=(X,y), method='BFGS', jac=gradientFunc,
options={'maxiter' : 400, 'disp': True})

print(result.x)
```

- result.x will contain the optimal $\boldsymbol{\theta}$ values
- Note that X must contain an extra column of zeros representing feature $\boldsymbol{x_0}$

- Predictions can then be made by applying $\boldsymbol{h_\theta}(\boldsymbol{x}) = g(\boldsymbol{\theta}^{\boldsymbol{T}}\boldsymbol{x})$

Regularized Logistic Regression Using sklearn

- Implemented in sklearn specifically for Logistic Regression:

```
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression(random_state=0, solver='lbfgs', multi_class='auto', C = 10 )

clf.fit(X, y)
```

$C = \frac{1}{\lambda} = 10$

So here:

$\lambda = \frac{1}{C} = 0.1$

- Regularization in the LogisticRegression class is specified differently: it is specified via a parameter C which is the inverse of λ.

$$C = \frac{1}{\lambda} \quad \text{so} \quad \lambda = \frac{1}{C}$$

- C has the same goal as, but opposite effect to, λ i.e.:
  - Setting C very small means setting λ very large i.e. **more** regularization
  - Setting C very large means setting λ very small i.e. **less** regularization

THE END

Of Logistic Regression Part 2