Logistic Regression

Prof. Dr. Josef F. Bürgler

Studiengang Informatik
Hochschule Luzern, Informatik

I.BA_ML

Introduction

Topic: Logistic Regression

Goals: You know why we use logistic regression and how it can be used to solve a binary classification problem.

Results: You can use logistic regression for classification (even in the case of more than two classes)

Further steps: After a short review of the classical (linear) regression we will show uses of logistic regression. Then we will derive the cost function from the Maximum Likelihood Estimation (MLE) and deduce the corresponding gradient descent algorithm.

Content

Content

Review of (Linear) Regression

- In regression analysis we want to find a (linear) relation between features (predictors, independent variables) $\boldsymbol{x} = [x_0 = 1, x_1, x_2, \ldots, x_m]^T$ like the size of house, the number of bath rooms, etc. and the dependent variable y like the price of the house.
- We are given a number of training examples $(\boldsymbol{x}^{(i)}, y^{(i)})$, $i = 1, 2, \ldots n$.
- Using either the normal equation or gradient descent we compute a parametervector $\boldsymbol{\theta} = (\theta_0, \theta_1, \theta_2, \ldots, \theta_m)^T$ by solving the least squares problem, i.e. we minimize the error (cost function)

$$J(\boldsymbol{\theta}) = \frac{1}{2} \sum_{i=1}^{n} \left(h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)}) - y^{(i)} \right)^2 \qquad \text{where} \quad h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)}) = \boldsymbol{x}^{(i)} \boldsymbol{\theta}^T$$

- This can be accomblished by
  - either advocating the normal equation

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

    where $\mathbf{X}$ is the data matrix whos rows are $\boldsymbol{x}^T = \left[x_0^{(i)} = 1, x_1^{(i)}, x_2^{(i)}, \ldots, x_m^{(i)}\right]$, $i = 1, 2, \ldots, n$ and $\boldsymbol{y} = \left[y^{(1)}, y^{(2)}, \ldots, y^{(n)}\right]^T$.
  - or by advocating the gradient descent method.

Review of (Linear) Regression (cont.)

In the case of (linear) regression the batch gradient descent method was

Start with (some initial guess) $\boldsymbol{\theta}_0$

Repeat(until convergence)$\{$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \alpha \frac{1}{n} \sum_{i=1}^{n} \left(h(\boldsymbol{\theta}_k, \boldsymbol{x}^{(i)}) - y^{(i)} \right) \boldsymbol{x}^{(i)}, \quad k = 0, 1, 2, 3, \ldots$$

$\}$

where $h(\boldsymbol{\theta}_k, \boldsymbol{x}^{(i)}) = (\boldsymbol{x}^{(i)})^T \boldsymbol{\theta} = \theta_0 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + \cdots + \theta_m x_m^{(i)}$.

We will show, that this procedure can also be applied to logistic regression. The only difference being the function h.

Content

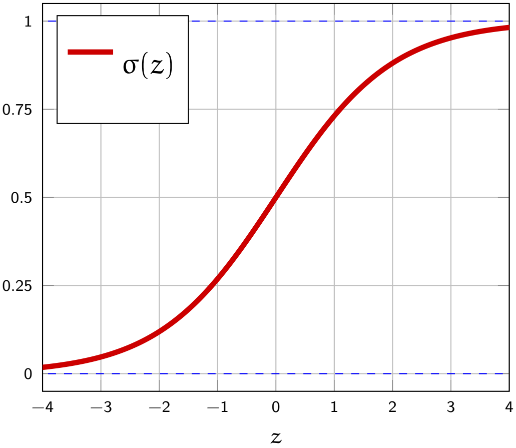
Why Logistic Regression

- The (binary) logistic Regression analysis is used to check whether or how a dependent binary variable $y = \{0, 1\}$ depends on one or more independent variables (features) $\boldsymbol{x}$.
- The dependent variable can be
  - E-Mail: Spam ($y = 1$) or not Spam ($y = 0$)
  - Person: Criminal ($y = 1$) or not Criminal ($y = 0$)
  - Student: passes the exam ($y = 1$) or does not pass the exam ($y = 0$)
- The independent variables are metric or encoded as dummy-variables in the case of categorical variables. They can be
  - E-Mail: Presence of words, number of typos, etc.
  - Person: Activity, Colleges, Work, life style, etc.
  - Student: Hours learn, parties, enough sleep, etc.
- The independent variables shouldn't be highly correlated
- Logistic regression is named for the function used at the core of the method, the logistic function, also called the sigmoid function, is defined by

$$\sigma(z) = \frac{1}{1 + e^{-z}}, \; z \in \mathbb{R}.$$

Content

The logistic (or sigmoid) function



Later we will use the derivative of the logistic (or sigmoid) function

$$\sigma(z) = \frac{1}{1+e^{-z}}, \ z \in \mathbb{R}.$$

Using the appropriate rules we get

$$\begin{aligned}
\sigma'(z) &= -\left(1+e^{-z}\right)^{-2}\left(-e^{-z}\right) \\
&= \frac{1+e^{-z}-1}{(1+e^{-z})^2} \\
&= \frac{1}{1+e^{-z}}\left(1-\frac{1}{1+e^{-z}}\right) \\
&= \sigma(z)\left(1-\sigma(z)\right).
\end{aligned}$$

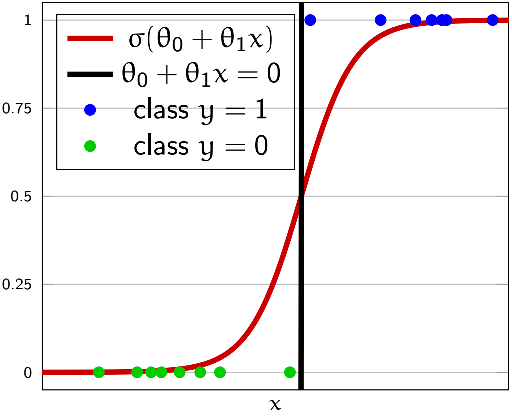
Analog findet man nach einer kleinen Rechnung

$$\sigma''(z) = \sigma(z)\left(1-\sigma(z)\right)\left(1-2\sigma(z)\right)$$

Logistic (sigmoid) function delivers probabilities

An example logistic regression equation with one input variable x is given by

$$y = \sigma(\theta_0 + \theta_1 x) = \frac{1}{1 + e^{-(\theta_0+\theta_1 x)}},\ x \in \mathbb{R}. \text{ Here we have used } z = \theta_0 + \theta_1 x$$



The blue bullets belong the class $y = 1$ (passes the exam) and the green ones to class $y = 0$ (does not pass the exam). The logistic (or sigmoid) function delivers the probability, that a data point is in class $y = 1$ or in $y = 0$.

If the probability is higher than 0.5 ($p(x) > 0.5$) then we say x belongs to class $y = 1$ else if $p(x) < 0.5$ then we say x belongs to class $y = 0$.

Logistic (sigmoid) function delivers probabilities (Example)

Example

Let's say our model predicts whether a student is male or female based on their height. Given a height $h = 155\,\mathrm{cm}$ is the student male $(y = 1)$ or female $(y = 0)$?

Assume that our algorithm has learned $\theta_0 = -100$ and $\theta_1 = 0.6\,\mathrm{cm}^{-1}$. Based on this we can compute the probability $P(\text{male}|h = 155\,\mathrm{cm})$:

$$P(\text{male}|h = 155\,\mathrm{cm}) = \hat{y} = \frac{1}{1 + \exp(-(\theta_0 + \theta_1 x))} = \frac{1}{1 + \exp(100 - 0.6 \cdot 155)}$$
$$= \frac{1}{1 + \exp(7)} = 0.9 \times 10^{-3}$$

So the probability is almost zero, that the student ist male.

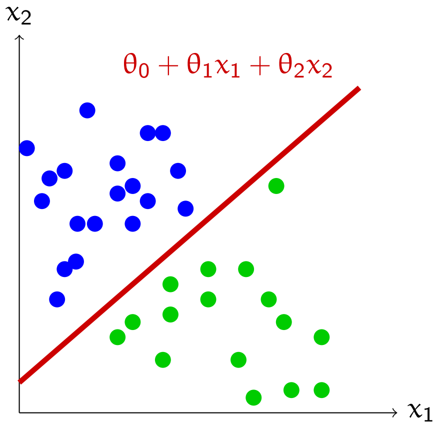
In practice we use the probabilities as follows

$$y = 1 \text{ if } p(\text{male}|h) \geqslant 0.5,$$
$$y = 0 \text{ if } p(\text{male}|h) < 0.5.$$

Content

Linear Decision Boundaries

Logistic Regression actually predicts probabilites. Based on these probabilities we are able to classify. If the probability is close to zero, we say it's class $y = 0$ and if the probability is close to one, we say it's class $y = 1$.



The decision boundary is given by the equation

$$h(\boldsymbol{\theta}, \boldsymbol{x}) = \sigma(\boldsymbol{x}^\top \boldsymbol{\theta}) \; = \; \sigma(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

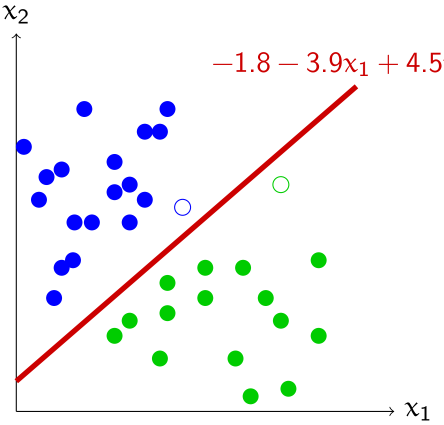
where

$$\boldsymbol{x}^\top = [x_0 = 1, x_1, x_2] \text{ and}$$
$$\boldsymbol{\theta} = [\theta_0, \theta_1, \theta_2]^\top .$$

We predict $y = 1$ if $\boldsymbol{x}^\top \boldsymbol{\theta} \geqslant 0$ and $y = 0$ if $\boldsymbol{x}^\top \boldsymbol{\theta} < 0$.

Linear Decision Boundaries (cont.)

Example



In our case the decision boundary is specified by $\boldsymbol{\theta} = [-1.8, -3.9, 4.5]^T$. This corresponds to the equation of the black line $-1.8 - 3.9x_1 + 4.5x_2 = 0$.

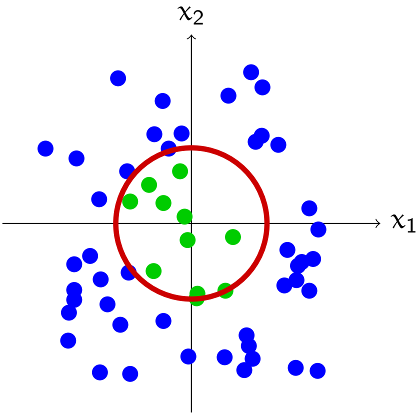
The probability for the feature vector $(x_1, x_2) = (2.2, 2.7)$ (blue circle) is given by $g(-1.8 - 3.9 \cdot 2.2 + 4.5 \cdot 2.7) = g(3.57) = 0.973 \geqslant 0.5$. Therefore, this feature vector belongs to class $y = 1$.

The probability for the feature vector $(x_1, x_2) = (3.5, 3)$ (green circle) is given by $g(-1.8 - 3.9 \cdot 3.5 + 4.5 \cdot 3) = g(-1.95) = 0.125 < 0.5$. Therefore, this feature vector belongs to class $y = 0$.

Nonlinear Decision Boundaries

Example

Nonlinear decision boundaries can can be described by nonlinear equations like $\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2$.



Here the decision boundary is a circle which is descried by the eqn. $-1 + x_1^2 + x_2^2 = 0$.

We predict $y = 1$ for the feature vector (x_1, x_2) if $x_1^2 + x_2^2 \geqslant 1$, i.e. if it lies outside the circle.

We predict $y = 0$ for the feature vector (x_1, x_2) if $x_1^2 + x_2^2 < 1$, i.e. if it lies inside the circle.

We can describe very complicated desision boundaries using, i.e.

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1 x_2 + \theta_5 x_2^2 + \theta_6 x_1^3 + \theta_7 x_1^2 x_2 + \theta_8 x_1 x_2^2 + \theta_9 x_2^3 + \cdots$$

Content

Maximum Likelihood Estimation

Having trained the logistic regression model, i.e. having found a suitable parameter vector $\boldsymbol{\theta}$, we want the prediction

$$\hat{y} = h(\boldsymbol{\theta}, \boldsymbol{x}) \;=\; \sigma(\boldsymbol{x}^\top \boldsymbol{\theta}) \text{ where } \quad \sigma(z) = \frac{1}{1 + e^{-z}} \text{ logistic function,}$$

of the probability that $y = 1$, given $\boldsymbol{x}$, i.e. $\hat{y} = P(y = 1|\boldsymbol{x})$ to be as "good as possible".

For a given feature vector $\boldsymbol{x}$

- if $y = 1$, then $p(y|\boldsymbol{x}) = \hat{y}$ is the chance of passing the exam, and
- if $y = 0$, then $p(y|\boldsymbol{x}) = 1 - \hat{y}$ is the chance of not passing the exam

We can combine the two equations into one equation

$$p(y|\boldsymbol{x}) = \hat{y}^y \left(1 - \hat{y}\right)^{1-y} \;=\; \begin{cases} \hat{y}^1 \left(1 - \hat{y}\right)^0 = \hat{y} & \text{if } y = 1 \\ \hat{y}^0 \left(1 - \hat{y}\right)^1 = 1 - \hat{y} & \text{if } y = 0 \end{cases}$$

In order to maximize the probability $p(y|\boldsymbol{x})$ we can just as well maximize $\log p(y|\boldsymbol{x})$, i.e. the log of the probability (because log is a strictly monotonic function).

Maximum Likelihood Estimation (cont.)

We can rewrite the log of the probability as follows:

$$\begin{aligned}\log p(y|\boldsymbol{x}) &= \log\left(\hat{y}^{y}\,(1-\hat{y})^{1-y}\right) \;=\; \log\left(\hat{y}^{y}\right) + \log\left((1-\hat{y})^{1-y}\right)\\ &= y\log\left(\hat{y}\right) + (1-y)\log\left(1-\hat{y}\right)\end{aligned}$$

We want to maximize the probability of all the labels $p(\text{"labels in the training set"})$. Assuming the training data are drawn independently and identically distributed (iid) the probability for the labels is

$$p(\text{"labels in the training set"}) = \prod_{i=1}^{n} p\left(y^{(i)}|\boldsymbol{x}^{(i)}\right)$$

In the Maximum Likelihood Estimation the parameter vector $\boldsymbol{\theta}$ is choosen, such that this probability is maximal. Because log is strictly monotonic, we can just as well maximize the log of the probability, i.e.

$$\log p(\text{"labels \ldots"}) = \log\left(\prod_{i=1}^{n} p\left(y^{(i)}|\boldsymbol{x}^{(i)}\right)\right) = \sum_{i=1}^{n}\log\left(p\left(y^{(i)}|\boldsymbol{x}^{(i)}\right)\right)$$

Maximum Likelihood Estimation (cont.)

Minimizing the cost function $J(\boldsymbol{\theta})$ (which is the negative of the log-probability) means maximizing the probabilities. If we want to carry out MLE, then we have to find the parameter vector $\boldsymbol{\theta}$, such that the following cost function is minimal:

$$\begin{aligned} J(\boldsymbol{\theta}) &= -\frac{1}{n}\sum_{i=1}^{n} \log\left(p\left(y^{(i)}|\boldsymbol{x}^{(i)}\right)\right) \\ &= -\frac{1}{n}\sum_{i=1}^{n}\left[y^{(i)}\log\left(h(\boldsymbol{\theta},\boldsymbol{x}^{(i)})\right) + \left(1-y^{(i)}\right)\log\left(1-h(\boldsymbol{\theta},\boldsymbol{x}^{(i)})\right)\right] \end{aligned}$$

where

$$h(\boldsymbol{\theta},\boldsymbol{x}^{(i)}) \;=\; \sigma\left((\boldsymbol{x}^{(i)})^T\boldsymbol{\theta}\right) \quad \text{and} \quad \sigma(z) = \frac{1}{1+e^{-z}}.$$

In addition we scaled the equation by the number of samples n (which does not influence the minimization). Note that we write $(\boldsymbol{x}^{(i)})^T\boldsymbol{\theta}$ and not $\boldsymbol{\theta}^T\boldsymbol{x}^{(i)}$ which is the same, because the transpose of a scalar is the scalar. The reason is, that $(\boldsymbol{x}^{(i)})^T$ represents the i-th row in the data matrix $\mathbf{X}$.

Maximum Likelihood Estimation (cont.)

We can therefore compute $h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)}) = \sigma\left((\boldsymbol{x}^{(i)})^T \boldsymbol{\theta}\right)$ very efficiently (i.e. using parallelization) by first computing the matrix vector product

$$\mathbf{X}\boldsymbol{\theta}$$

which results in a nx1-vector, then apply the sigmoid function component wise

$$\sigma\left(\mathbf{X}\boldsymbol{\theta}\right)$$

and finally substract the target vector $\boldsymbol{y}$

$$\sigma\left(\mathbf{X}\boldsymbol{\theta}\right) - \boldsymbol{y}$$

We can show, that gradient descent can be accomblished using the following (matrix-) equation

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \alpha \frac{1}{n} \mathbf{X}^T \left(\sigma\left(\mathbf{X}\boldsymbol{\theta}\right) - \boldsymbol{y}\right)$$

Content

Gradient Descent

In order to compute the gradient of the logistic regression cost function, we need the following partial derivatives

$$\begin{aligned}\frac{\partial}{\partial\theta_k}h(\boldsymbol{\theta},\boldsymbol{x}^{(i)}) &= \frac{\partial}{\partial\theta_k}\sigma\left((\boldsymbol{x}^{(i)})^\top\boldsymbol{\theta}\right)\\ &= \sigma\left((\boldsymbol{x}^{(i)})^\top\boldsymbol{\theta}\right)\left(1-\sigma\left((\boldsymbol{x}^{(i)})^\top\boldsymbol{\theta}\right)\right)\frac{\partial}{\partial\theta_k}\left((\boldsymbol{x}^{(i)})^\top\boldsymbol{\theta}\right)\\ &= \sigma\left((\boldsymbol{x}^{(i)})^\top\boldsymbol{\theta}\right)\left(1-\sigma\left((\boldsymbol{x}^{(i)})^\top\boldsymbol{\theta}\right)\right)x_k^{(i)}\\ &= h(\boldsymbol{\theta},\boldsymbol{x}^{(i)})\left(1-h(\boldsymbol{\theta},\boldsymbol{x}^{(i)})\right)x_k^{(i)}\end{aligned}$$

Here we applied the chain rule, the derivative of the sigmoid function $\sigma'(z) = \sigma(z)\,(1-\sigma(z))$, and

$$\frac{\partial}{\partial\theta_k}\left((\boldsymbol{x}^{(i)})^\top\boldsymbol{\theta}\right) = \frac{\partial}{\partial\theta_k}\left(\theta_0+\theta_1x_1^{(i)}+\theta_2x_2^{(i)}+\cdots+\theta_mx_m^{(i)}\right) \;=\; x_k^{(i)}$$

Gradient Descent (cont.)

Therefore we have

$$\begin{aligned}
\frac{\partial}{\partial \theta_k} \log\left(h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)})\right) &= \frac{1}{h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)})} \frac{\partial}{\partial \theta_k} h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)}) \\
&= \frac{1}{h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)})} h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)}) \left(1 - h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)})\right) x_k^{(i)} \\
&= \left(1 - h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)})\right) x_k^{(i)}
\end{aligned}$$

and

$$\begin{aligned}
\frac{\partial}{\partial \theta_k} \log\left(1 - h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)})\right) &= \frac{1}{1 - h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)})} \frac{\partial}{\partial \theta_k} \left(1 - h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)})\right) \\
&= -\frac{1}{1 - h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)})} \frac{\partial}{\partial \theta_k} h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)}) \\
&= -\frac{1}{1 - h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)})} h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)}) \left(1 - h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)})\right) x_k^{(i)} \\
&= -h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)}) x_k^{(i)}
\end{aligned}$$

Gradient Descent (cont.)

Finally the partial derivative of the i-th summand $J_i(\boldsymbol{\theta})$ in the cost function is

$$
\begin{aligned}
\frac{\partial}{\partial \theta_k} J_i(\boldsymbol{\theta}) &= -\frac{1}{n}\frac{\partial}{\partial \theta_k}\left[y^{(i)} \log\left(h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)})\right) + \left(1 - y^{(i)}\right) \log\left(1 - h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)})\right)\right] \\
&= -\frac{1}{n}\left[y^{(i)}\left(1 - h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)})\right) - (1 - y^{(i)}) h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)})\right] x_k^{(i)} \\
&= -\frac{1}{n}\left[y^{(i)} - y^{(i)} h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)}) - h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)}) + y^{(i)} h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)})\right] x_k^{(i)} \\
&= \frac{1}{n}\left(h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)}) - y^{(i)}\right) x_k^{(i)}
\end{aligned}
$$

And finally we obtain the same gradient as in the case of standard (linear) regression

$$
\frac{\partial}{\partial \theta_k} J(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}\left(h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)}) - y^{(i)}\right) x_k^{(i)}
$$

where $h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)}) = \sigma\left((\boldsymbol{x}^{(i)})^{\mathsf{T}} \boldsymbol{\theta}\right)$ and $\sigma(z) = \dfrac{1}{1 + e^{-z}}$.

Gradient Descent — the Algorithm

Given n samples in the training set, $(\boldsymbol{x}^{(i)}, y^{(i)})$, $i = 1, 2, \ldots, n$ of a binary classification problem ($y \in \{0, 1\}$) the cost function

$$J(\boldsymbol{\theta}) = \sum_{i=1}^{n} J_i(\boldsymbol{\theta})$$
$$= -\frac{1}{n} \sum_{i=1}^{n} \left[y^{(i)} \log\left(h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)}) \right) + \left(1 - y^{(i)} \right) \log\left(1 - h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)}) \right) \right]$$

is minimized using for example gradient descent as follows: Start with some (usually) random paramter vector $\boldsymbol{\theta}_0$ and iterate (until convergence)

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \alpha \frac{1}{n} \sum_{i=1}^{n} \left(h(\boldsymbol{\theta}_k, \boldsymbol{x}^{(i)}) - y^{(i)} \right) \boldsymbol{x}^{(i)}, \quad k = 0,\ 1,\ 2,\ 3,\ \ldots$$

where $h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)}) = \sigma\left((\boldsymbol{x}^{(i)})^T \boldsymbol{\theta} \right)$ and $\sigma(z) = \dfrac{1}{1 + e^{-z}}$. Note in the case of standard (linear) regression we had $h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)}) = (\boldsymbol{x}^{(i)})^T \boldsymbol{\theta}$.

Content

(Binary) Classification

Once we have trained the system and learned a suitable parameter vector $\boldsymbol{\theta}$ using gradient descent, we can now estimate the class for a certain set of features, represented by the vector $\boldsymbol{x}$:

$$\hat{y} = \begin{cases} 1 & \text{if } \sigma(\boldsymbol{x}^\top \boldsymbol{\theta}) \geqslant 0.5 \\ 0 & \text{if } \sigma(\boldsymbol{x}^\top \boldsymbol{\theta}) < 0.5 \end{cases}$$

If we have more than two classes, we use the method "one versus the rest". If we have three classes, we have to solve three classification problems: first class 1 versus the other two classes 2 and 3, than class 2 versus the other two classes 1 and 3 and finally class 3 versus the other tow classes 1 and 2. For each of these three cases we find a parameter vector $\boldsymbol{\theta}_i$ $(i = 1, 2, 3)$.

We say feature vector $\boldsymbol{x}$ is in class i if

$$i = \underset{k \in \{1,2,3\}}{\text{argmax}}\, \sigma(\boldsymbol{x}^\top \boldsymbol{\theta}_k)$$

Content

Simple Example

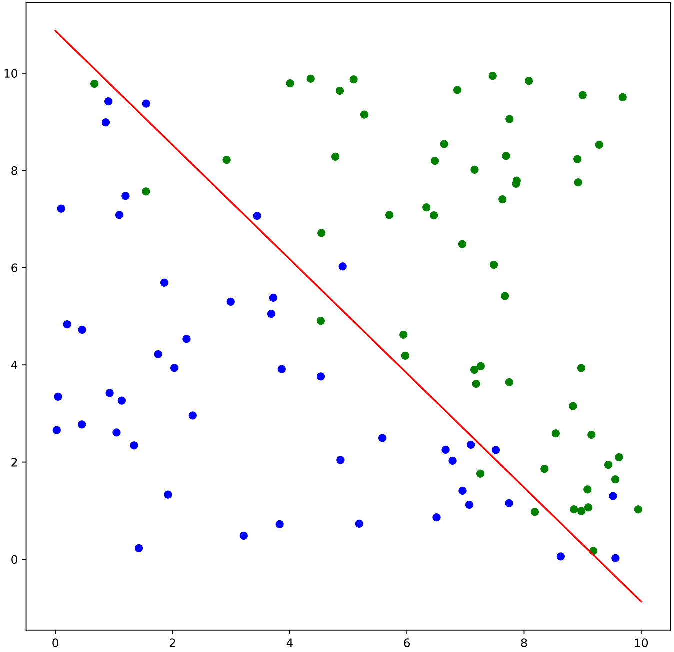
```python
import numpy as np
from numpy import genfromtxt
from matplotlib import pyplot as plt

data_01=genfromtxt('classification_data.csv',
                delimiter=',')
n = data_01.shape[0]
X = np.c_[np.ones((n,1)),data_01[:,0:2]]
y = np.array([data_01[:,2]]).transpose()

def sigmoid(z):
    return 1/(1+np.exp(-z))

def cost_function(X, y, theta):
    y_hat = sigmoid(np.dot(X,theta))
    J_i = - y*np.log(y_hat)
        - (1-y)*np.log(1-y_hat)
    J = J_i.sum()/len(y)
    return J
```

```python
def update_theta(X, y, theta, alpha):
    y_hat = sigmoid(np.dot(X, theta))
    gradient = np.dot(X.T,y_hat - y)/len(y)
    theta -= alpha*gradient
    return theta



def train(X, y, theta, alpha, kmax):
    cost_history = []
    for i in range(kmax):
        theta = update_theta(X,y,theta,alpha)
        cost = cost_function(X,y,theta)
        cost_history.append(cost)
    return theta, cost_history
```

Simple Example

```
theta = np.array([[0.1],[-0.1],[0.2]])
alpha = 0.1; kmax = 100000

theta, cost_history = train(X, y, theta, alpha, kmax)
print("decision boundary: %.3f + %.3f * x1 + %.3f * x2 = 0"
     % (theta[0],theta[1],theta[2]))

x1 = np.array(X[:,1].T); x2 = np.array(X[:,2].T)
fig, ax = plt.subplots(1,1, figsize=(10,10))
color = ['blue' if l == 0 else 'green' for l in y]
scat = ax.scatter(x1, x2, color=color)

y = lambda x: ((-1)*(theta[0] + theta[1]*x) / theta[2])

def plot_line(y, data_pts):
   x_vals = [i for i in range(int(min(data_pts)-1), int(max(data_pts))+2)]
   y_vals = [y(x) for x in x_vals]
   plt.plot(x_vals,y_vals, 'r')

plot_line(y, x1)
plt.show()
```

Simple Example



Conclusion

- Students know when and why to advocate logistic regression.
- Students know from first principles (MLE) how the cost function is derived.
- Students are able to implement logistic regression in python.
- Students can solve by hand the first step of the gradient descent method in logistic regression.
- Students can solve more complicated examples using python.
- Students are able to judge, whether their solution is meaningful.

I'm happy to answer Your

Questions