Logistic Regression

by Prof. Seungchul Lee
iSystems Design Lab
http://isystems.unist.ac.kr/
UNIST

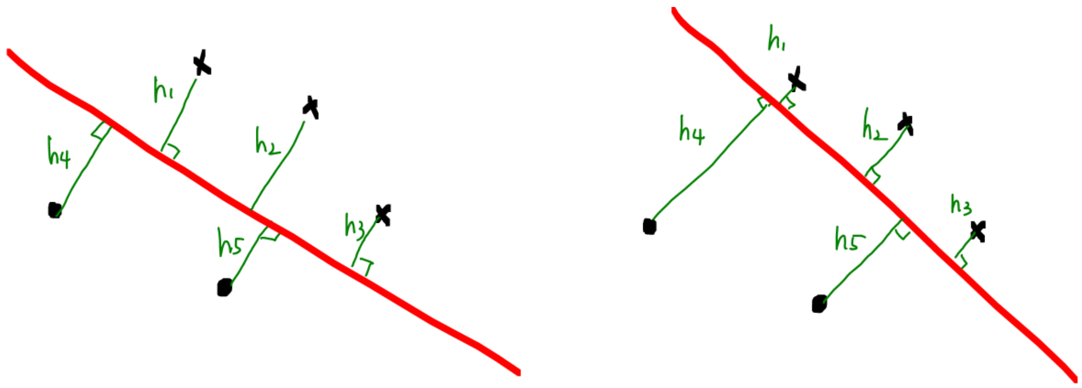
Table of Contents

1. Linear Classification: Logistic Regression

- Logistic regression is a classification algorithm - don't be confused

1.1. Using all Distances

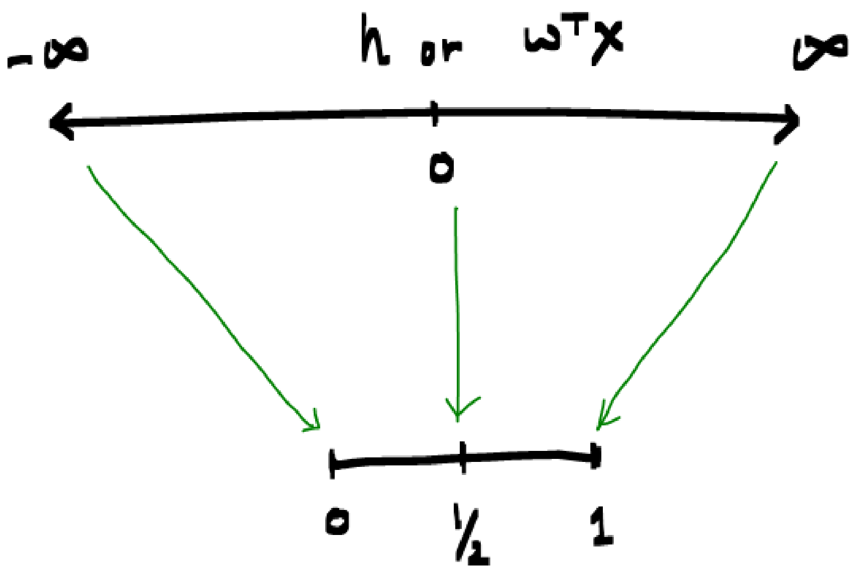
- Perceptron: make use of sign of data
- SVM: make use of margin (minimum distance)
- We want to use *distance information of all data points* $\rightarrow$ logistic regression



- basic idea: to find the decision boundary (hyperplane) of $g(x) = \omega^T x = 0$ such that maximizes $\prod_i |h_i|$
  - Inequality of arithmetic and geometric means
$$\frac{h_1 + h_2}{2} \geq \sqrt{h_1 h_2}$$
    and that equality holds if and only if $h_1 = h_2$

- Roughly speaking, this optimization of $\max \prod_i |h_i|$ tends to position a hyperplane in the middle of two classes
$$h = \frac{g(x)}{\|\omega\|} = \frac{\omega^T x}{\|\omega\|} \sim \omega^T x$$

- We link or squeeze $(-\infty, +\infty)$ to $(0, 1)$ for several reasons:



- If $\sigma(z)$ is the sigmoid function, or the logistic function

$$\sigma(z) = \frac{1}{1+e^{-z}} \implies \sigma(\omega^T x) = \frac{1}{1+e^{-\omega^T x}}$$

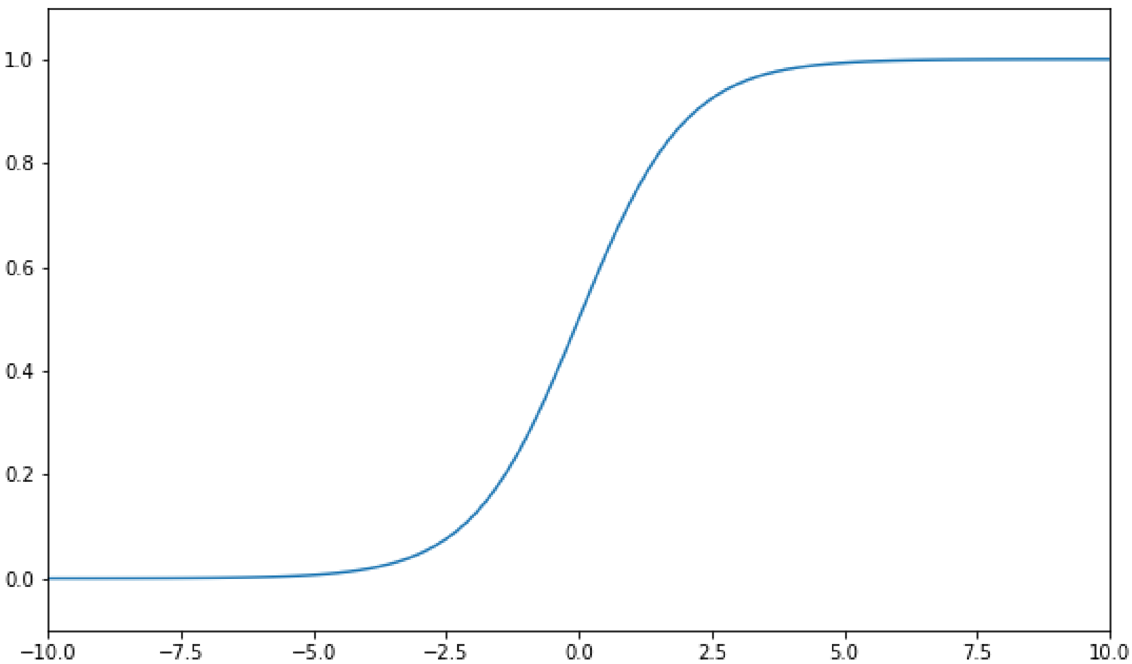
  - logistic function always generates a value between 0 and 1
  - Crosses 0.5 at the origin, then flattens out

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
```

In [2]:

```
z = np.linspace(-10,10,100)
s = 1/(1+np.exp(-z))

plt.figure(figsize=(10,6))
plt.plot(z, s)
plt.xlim([-10, 10])
plt.ylim([-0.1, 1.1])
plt.show()
```



- Benefit of mapping via the logistic function
  - monotonic: same or similar optimziation solution
  - continuous and differentiable: good for gradient descent optimization
  - probability or confidence: can be considered as probability

$$P\left(y=+1 \mid x, \omega\right)=\frac{1}{1+e^{-\omega^{T} x}} \quad \in \ [0,1]$$

  - Often we do note care about predicting the label y
  - Rather, we want to predict the label probabilities $P\left(y \mid x, \omega\right)$
    - the probability that the label is $+1$
$$P\left(y=+1 \mid x, \omega\right)$$
    - the probability that the label is 0
$$P\left(y=0 \mid x, \omega\right)=1-P\left(y=+1 \mid x, \omega\right)$$

- Goal: we need to fit ω to our data

1.2. Probabilistic Approach (or MLE)

Consider a random variable $y \in \{0, 1\}$

$$P(y = +1) = p, \quad P(y = 0) = 1 - p$$

where $p \in [0, 1]$, and is assumed to depend on a vector of explanatory variables $x \in \mathbb{R}^n$

Then, the logistic model has the form

$$\begin{aligned} p &= \frac{1}{1 + e^{-\omega^T x}} = \frac{e^{\omega^T x}}{e^{\omega^T x} + 1} \\ 1 - p &= \frac{1}{e^{\omega^T x} + 1} \end{aligned}$$

We can re-order the training data so

- for $x_1, \cdots, x_q$, the outcome is $y = +1$, and
- for $x_{q+1}, \cdots, x_m$, the outcome is $y = 0$

The likelihood function

$$\mathscr{L} = \prod_{i=1}^{q} p_i \prod_{i=q+1}^{m} (1 - p_i) \qquad \left(\sim \prod_i |h_i| \right)$$

the log likelihood function

$$\begin{aligned} \ell(\omega) = \log \mathscr{L} &= \sum_{i=1}^{q} \log p_i + \sum_{i=q+1}^{m} \log(1 - p_i) \\ &= \sum_{i=1}^{q} \log \frac{\exp\left(\omega^T x_i\right)}{1 + \exp(\omega^T x_i)} + \sum_{i=q+1}^{m} \log \frac{1}{1 + \exp(\omega^T x_i)} \\ &= \sum_{i=1}^{q} \left(\omega^T x_i\right) - \sum_{i=1}^{m} \log\left(1 + \exp\left(\omega^T x_i\right)\right) \end{aligned}$$

Since ℓ is a concave function of ω, the logistic regression problem can be solved as a convex optimization problem

$$\hat{\omega} = \arg\max_{\omega} \mathrm{l}(\omega)$$

1.3. CVXPY

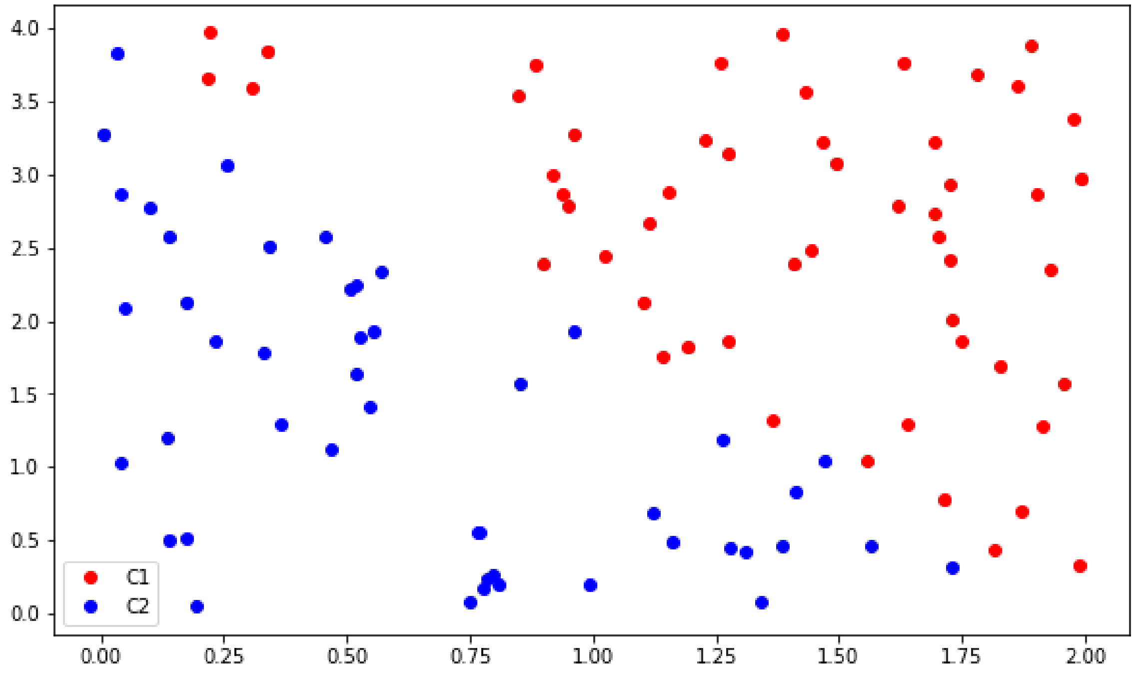
$$\omega = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}, \qquad x = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$$

$$X = \begin{bmatrix} \left(x^{(1)}\right)^T \\ \left(x^{(2)}\right)^T \\ \left(x^{(3)}\right)^T \\ \vdots \end{bmatrix} = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} \\ 1 & x_1^{(3)} & x_2^{(3)} \\ \vdots & \vdots & \vdots \end{bmatrix}$$

Source: Section 7.1.1 from http://cvxr.com/cvx/examples/cvxbook/Ch07_statistical_estim/html/logistics.html (http://cvxr.com/cvx/examples/cvxbook/Ch07_statistical_estim/html/logistics.html)

In [3]:

```
m = 100
w = np.array([[-4], [2], [1]])
X = np.hstack([np.ones([m,1]), 2*np.random.rand(m,1), 4*np.random.rand(m,1)])

w = np.asmatrix(w)
X = np.asmatrix(X)

y = (np.exp(X*w)/(1+np.exp(X*w))) > 0.5

C1 = np.where(y == True)[0]
C2 = np.where(y == False)[0]

y = np.empty([m,1])
y[C1] = 1
y[C2] = 0
y = np.asmatrix(y)

plt.figure(figsize = (10,6))
plt.plot(X[C1,1], X[C1,2], 'ro', label='C1')
plt.plot(X[C2,1], X[C2,2], 'bo', label='C2')
plt.legend()
plt.show()
```



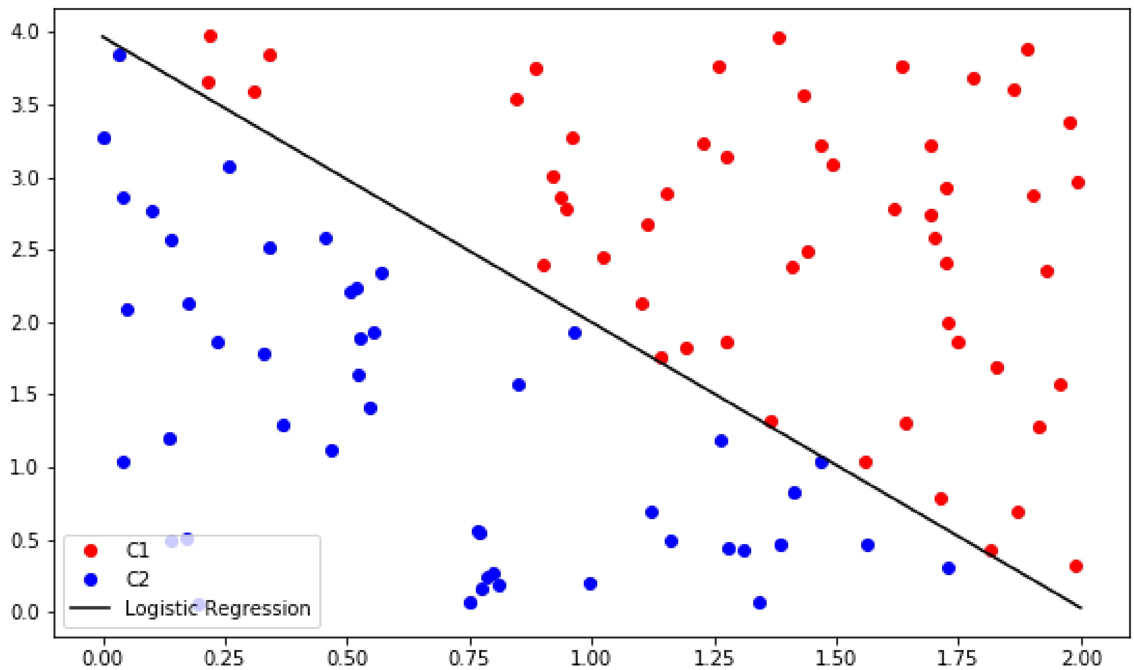
$$
\begin{aligned}
\ell(\omega) = \log \mathscr{L} &= \sum_{i=1}^{q} \log p_i + \sum_{i=q+1}^{m} \log(1 - p_i) \\
&= \sum_{i=1}^{q} \log \frac{\exp(\omega^T x_i)}{1 + \exp(\omega^T x_i)} + \sum_{i=q+1}^{m} \log \frac{1}{1 + \exp(\omega^T x_i)} \\
&= \sum_{i=1}^{q} (\omega^T x_i) - \sum_{i=1}^{m} \log(1 + \exp(\omega^T x_i))
\end{aligned}
$$

Refer to cvx functions (http://www.cvxpy.org/en/latest/tutorial/functions/)

- scalar function: `cvx.sum_entries(x)` = $\sum_{ij} x_{ij}$
- elementwise function: `cvx.logistic(x)` = $\log(1 + e^x)$

In [4]:

```
import cvxpy as cvx

w = cvx.Variable(3, 1)

obj = cvx.Maximize(y.T*X*w - cvx.sum_entries(cvx.logistic(X*w)))
prob = cvx.Problem(obj).solve()

w = w.value

xp = np.linspace(0,2,100).reshape(-1,1)
yp = - w[1,0]/w[2,0]*xp - w[0,0]/w[2,0]

plt.figure(figsize = (10,6))
plt.plot(X[C1,1], X[C1,2], 'ro', label='C1')
plt.plot(X[C2,1], X[C2,2], 'bo', label='C2')
plt.plot(xp, yp, 'k', label='Logistic Regression')
plt.legend()
plt.show()
```



In a more compact form

Change $y \in \{0, +1\} \to y \in \{-1, +1\}$ for compuational convenience

- Consider the following function

$$P(y=+1) = p = \sigma(\omega^T x), \quad P(y=-1) = 1-p = 1-\sigma(\omega^T x) = \sigma(-\omega^T x)$$

$$P(y \mid x, \omega) = \sigma\left(y\omega^T x\right) = \frac{1}{1+\exp(-y\omega^T x)} \in [0,1]$$

- Log-likelihood

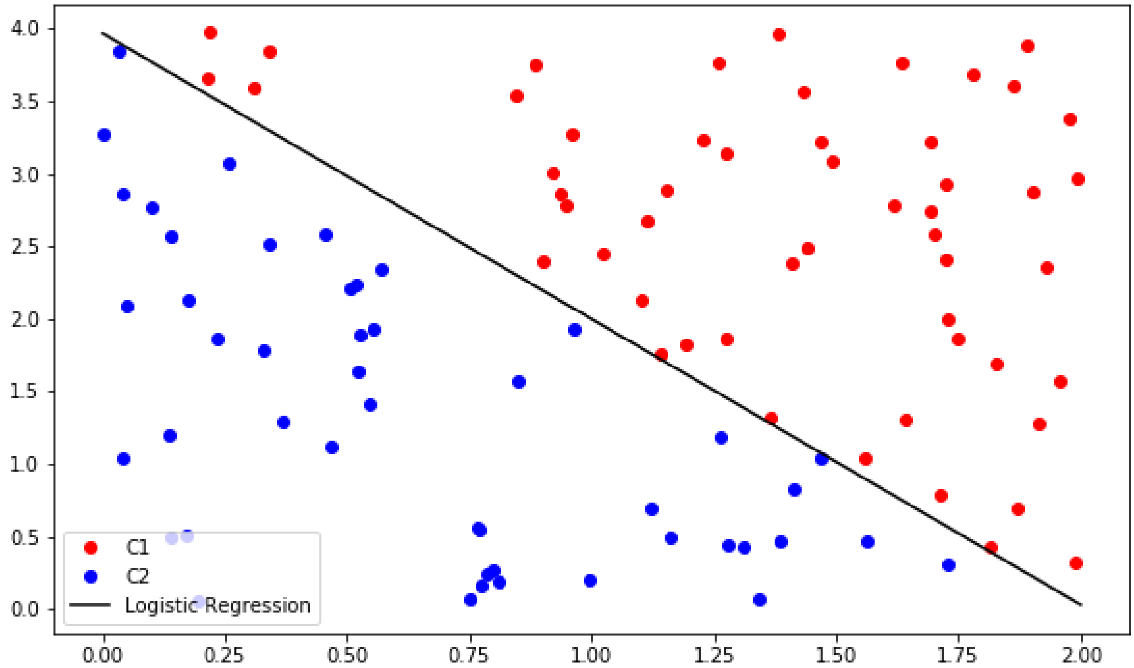
$$\begin{aligned}
\ell(\omega) = \log \mathscr{L} = \log P(y \mid x, \omega) &= \log \prod_{n=1}^{m} P(y_n \mid x_n, \omega) \\
&= \sum_{n=1}^{m} \log P(y_n \mid x_n, \omega) \\
&= \sum_{n=1}^{m} \log \frac{1}{1+\exp(-y_n \omega^T x_n)} \\
&= \sum_{n=1}^{m} -\log\bigl(1+\exp\bigl(-y_n \omega^T x_n\bigr)\bigr)
\end{aligned}$$

- MLE solution

$$\begin{aligned}
\hat{\omega} &= \arg\max_{\omega} \sum_{n=1}^{m} -\log\bigl(1+\exp\bigl(-y_n \omega^T x_n\bigr)\bigr) \\
&= \arg\min_{\omega} \sum_{n=1}^{m} \log\bigl(1+\exp\bigl(-y_n \omega^T x_n\bigr)\bigr)
\end{aligned}$$

In [5]:

```
y = np.empty([m,1])
y[C1] = 1
y[C2] = -1
y = np.asmatrix(y)

w = cvx.Variable(3, 1)

obj = cvx.Minimize(cvx.sum_entries(cvx.logistic(-cvx.mul_elemwise(y,X*w))))
prob = cvx.Problem(obj).solve()

w = w.value

xp = np.linspace(0,2,100).reshape(-1,1)
yp = - w[1,0]/w[2,0]*xp - w[0,0]/w[2,0]

plt.figure(figsize = (10,6))
plt.plot(X[C1,1], X[C1,2], 'ro', label='C1')
plt.plot(X[C2,1], X[C2,2], 'bo', label='C2')
plt.plot(xp, yp, 'k', label='Logistic Regression')
plt.legend()
plt.show()
```



2. Multiclass Classification

- Generalization to more than 2 classes is straightforward
  - one vs. all (one vs. rest)
  - one vs. one

- Using the soft-max function instead of the logistic function (refer to UFLDL Tutorial (http://ufldl.stanford.edu/tutorial/supervised/SoftmaxRegression/))
  - see them as probability

$$P\left(y=k \mid x, \omega\right)=\frac{\exp\left(\omega_k^T x\right)}{\sum_k \exp\left(\omega_k^T x\right)} \in [0,1]$$

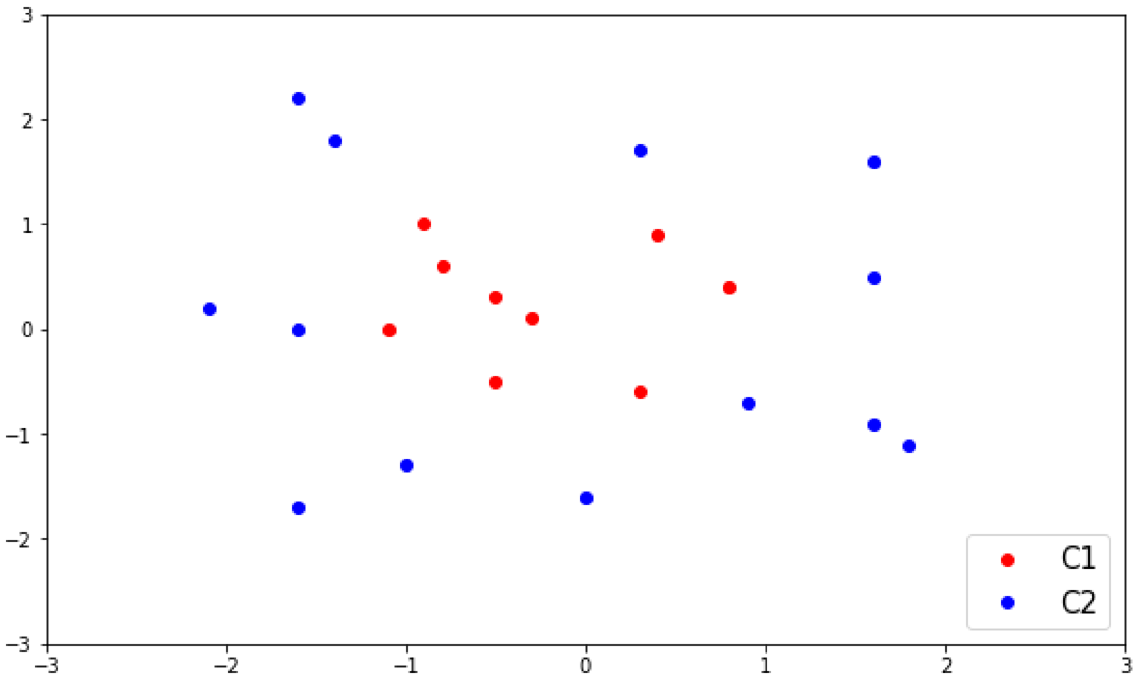
- We maintain a separator weight vector ω_k for each class k

3. Non-linear Classification

- Same idea as for linear regression: non-linear features, either explicit or implicit Kernels

In [6]:

```
X1 = np.array([[-1.1,  0], [-0.3,  0.1], [-0.9,  1],[0.8,  0.4],[0.4,  0.9],[0.3,-0.6],
[-0.5, 0.3],
        [-0.8,  0.6],[-0.5, -0.5]])

X2 = np.array([[-1,  -1.3], [-1.6 , 2.2],  [0.9, -0.7],[1.6,  0.5],[1.8, -1.1],[1.6,
1.6],[-1.6, -1.7],
        [-1.4,  1.8],[1.6, -0.9],[0, -1.6],[0.3, 1.7],[-1.6 , 0],[-2.1,0.2]])

X1 = np.asmatrix(X1)
X2 = np.asmatrix(X2)

plt.figure(figsize=(10, 6))
plt.plot(X1[:, 0], X1[:,1], 'ro', label='C1')
plt.plot(X2[:, 0], X2[:,1], 'bo', label='C2')
plt.axis([-3,3,-3,3])
plt.legend(loc = 4, fontsize = 15)
plt.show()
```



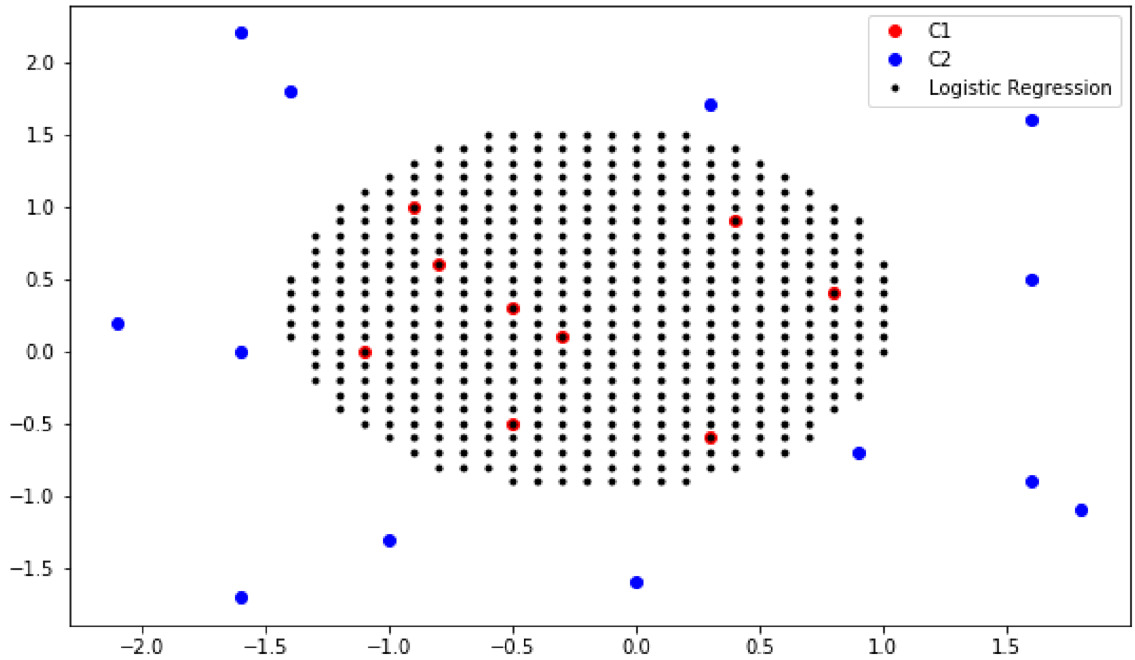
$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \Longrightarrow \quad z = \phi(x) = \begin{bmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}$$

In [7]:

```
N = X1.shape[0]
M = X2.shape[0]

X = np.vstack([X1, X2])
y = np.vstack([np.ones([N,1]), -np.ones([M,1])])

X = np.asmatrix(X)
y = np.asmatrix(y)

m = N + M
Z = np.hstack([np.ones([m,1]), np.sqrt(2)*X[:,0], np.sqrt(2)*X[:,1], np.square(X[:,0]),
 \
              np.sqrt(2)*np.multiply(X[:,0],X[:,1]), np.square(X[:,1])])

w = cvx.Variable(6, 1)
obj = cvx.Minimize(cvx.sum_entries(cvx.logistic(-cvx.mul_elemwise(y,Z*w))))
prob = cvx.Problem(obj).solve()

w = w.value
```

In [8]:

```
# to plot
[X1gr, X2gr] = np.meshgrid(np.arange(-3,3,0.1), np.arange(-3,3,0.1))

test_X = np.hstack([X1gr.reshape(-1,1), X2gr.reshape(-1,1)])
test_X = np.asmatrix(test_X)

m = test_X.shape[0]
test_Z = np.hstack([np.ones([m,1]), np.sqrt(2)*test_X[:,0], np.sqrt(2)*test_X[:,1],
np.square(test_X[:,0]), \
               np.sqrt(2)*np.multiply(test_X[:,0],test_X[:,1]),
np.square(test_X[:,1])])
q = test_Z*w

B = []

for i in range(m):
    if q[i,0] > 0:
        B.append(test_X[i,:])

B = np.vstack(B)

plt.figure(figsize=(10, 6))
plt.plot(X1[:,0], X1[:,1], 'ro', label='C1')
plt.plot(X2[:,0], X2[:,1], 'bo', label='C2')
plt.plot(B[:,0], B[:,1], 'k.', label='Logistic Regression')
plt.legend()
plt.show()
```



In [9]:

```
%%javascript
$.getScript('https://kmahelona.github.io/ipython_notebook_goodies/ipython_notebook_toc.
js')
```