# Regressão Logistica

October 6, 2022

[1]:
```
from IPython.display import Image
%matplotlib inline
```

[2]:
```
from sklearn import datasets
import numpy as np
iris=datasets.load_iris()
X=iris.data[:,[2,3]]
y=iris.target
```

[3]:
```
print('Rótulos das classes:', np.unique(y))
```

```
Rótulos das classes: [0 1 2]
```

[4]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.3,
                                                    random_state=1,
                                                    stratify=y)
```

[5]:
```
print("Contador de rótulos em y:", np.bincount(y))
print("Contador de rótulos em y_train:", np.bincount(y_train))
print("Contador de rótulos em y_test:", np.bincount(y_test))
```

```
Contador de rótulos em y: [50 50 50]
Contador de rótulos em y_train: [35 35 35]
Contador de rótulos em y_test: [15 15 15]
```

[6]:
```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
sc.fit(X_train)
X_train_std=sc.transform(X_train)
X_test_std=sc.transform(X_test)
```

[8]:
```
from sklearn.linear_model import Perceptron
ppn=Perceptron(eta0=0.01, random_state=1)
ppn.fit(X_train_std, y_train)
```

[8]:
```
Perceptron(eta0=0.01, random_state=1)
```

[9]:

```
y_pred=ppn.predict(X_test_std)
```

[10]:

```
from sklearn.metrics import accuracy_score


print('Acuracia: %.3f' %accuracy_score(y_test, y_pred))
```

```
Acuracia: 0.956
```

[11]:

```
from matplotlib.colors import ListedColormap
import matplotlib.pyplot as plt

# To check recent matplotlib compatibility
import matplotlib
from distutils.version import LooseVersion


def plot_decision_regions(X, y, classifier, test_idx=None, resolution=0.02):

    # setup marker generator and color map
    markers = ('s', 'x', 'o', '^', 'v')
    colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
    cmap = ListedColormap(colors[:len(np.unique(y))])

    # plot the decision surface
    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
                           np.arange(x2_min, x2_max, resolution))
    Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
    Z = Z.reshape(xx1.shape)
    plt.contourf(xx1, xx2, Z, alpha=0.3, cmap=cmap)
    plt.xlim(xx1.min(), xx1.max())
    plt.ylim(xx2.min(), xx2.max())

    for idx, cl in enumerate(np.unique(y)):
        plt.scatter(x=X[y == cl, 0],
                    y=X[y == cl, 1],
                    alpha=0.8,
                    color=colors[idx],
                    marker=markers[idx],
                    label=cl,
                    edgecolor='black')

    # highlight test examples
    if test_idx:
        # plot all examples
        X_test, y_test = X[test_idx, :], y[test_idx]
        if LooseVersion(matplotlib.__version__) < LooseVersion('0.3.4'):
            plt.scatter(X_test[:, 0],
                        X_test[:, 1],
                        c='',
                        edgecolor='black',
                        alpha=1.0,
                        linewidth=1,
                        marker='o',
                        s=100,
                        label='test set')
        else:
            plt.scatter(X_test[:, 0],
                        X_test[:, 1],
                        c='none',
                        edgecolor='black',
                        alpha=1.0,
                        linewidth=1,
                        marker='o',
                        s=100,
                        label='test set')
```

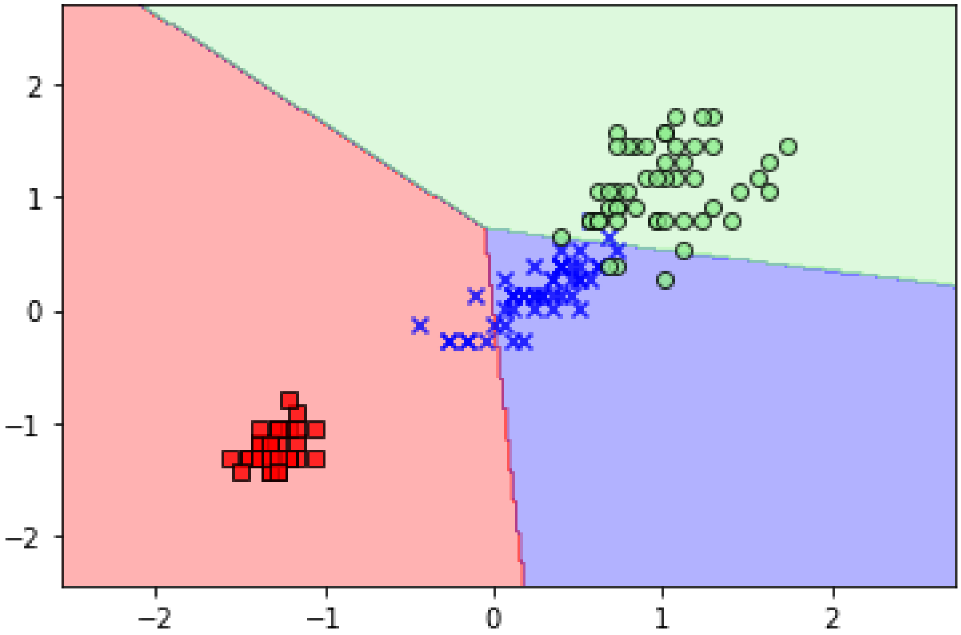
[13]:

```
X_combined_std=np.vstack((X_train_std, X_test_std))
y_combined=np.hstack((y_train, y_test))
plot_decision_regions(X=X_combined_std, y=y_combined,
                      classifier=ppn, test_idx=range(105, 159))

plt.xlabel('comprimento da pétala')
plt.ylabel('largura da pétala')
plt.legend(loc='upper left')
plt.show()
```

```
/tmp/ipykernel_145292/3603768161.py:28: UserWarning: You passed a
edgecolor/edgecolors ('black') for an unfilled marker ('x').  Matplotlib is
ignoring the edgecolor in favor of the facecolor.  This behavior may change in
the future.
  plt.scatter(x=X[y == cl, 0],
```

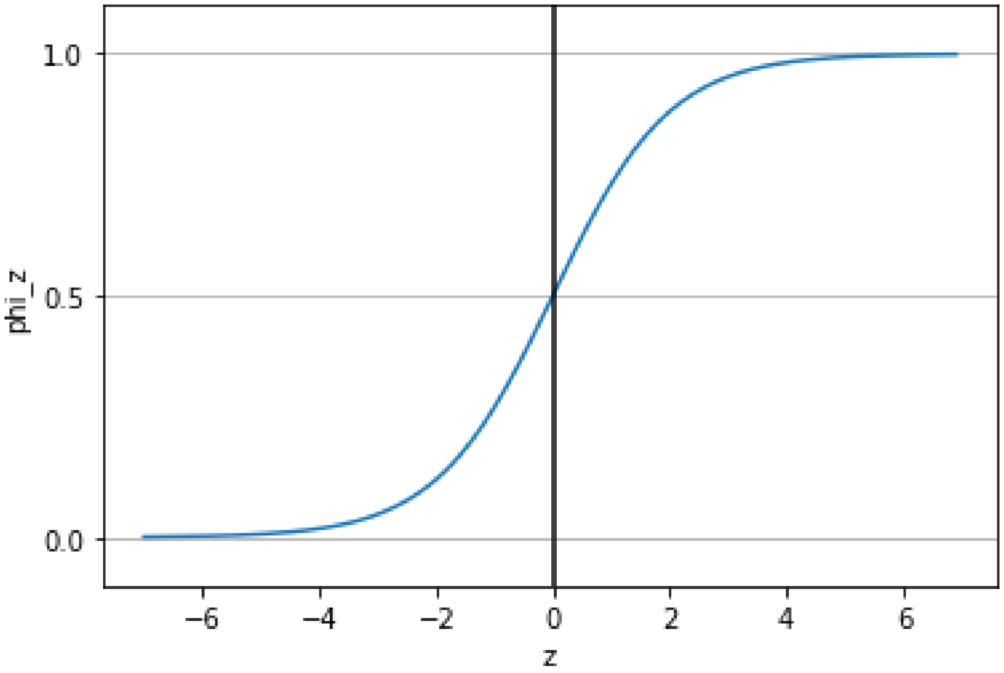
```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
/tmp/ipykernel_145292/3810697090.py in <module>
      1 X_combined_std=np.vstack((X_train_std, X_test_std))
      2 y_combined=np.hstack((y_train, y_test))
----> 3 plot_decision_regions(X=X_combined_std, y=y_combined,

      4                       classifier=ppn, test_idx=range(105, 159))
      5
```

```
/tmp/ipykernel_145292/3603768161.py in plot_decision_regions(X, y, classifier,␣
 ↪test_idx, resolution)
     37     if test_idx:
     38         # plot all examples
---> 39         X_test, y_test = X[test_idx, :], y[test_idx]
     40 
     41 

IndexError: index 150 is out of bounds for axis 0 with size 150
```



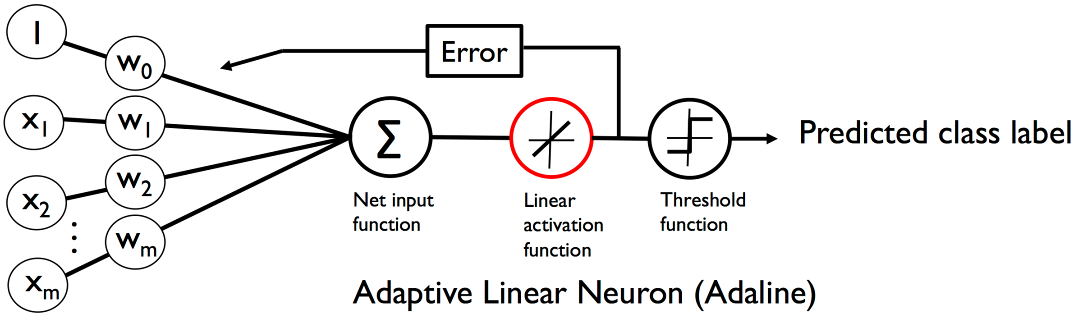
[14]:

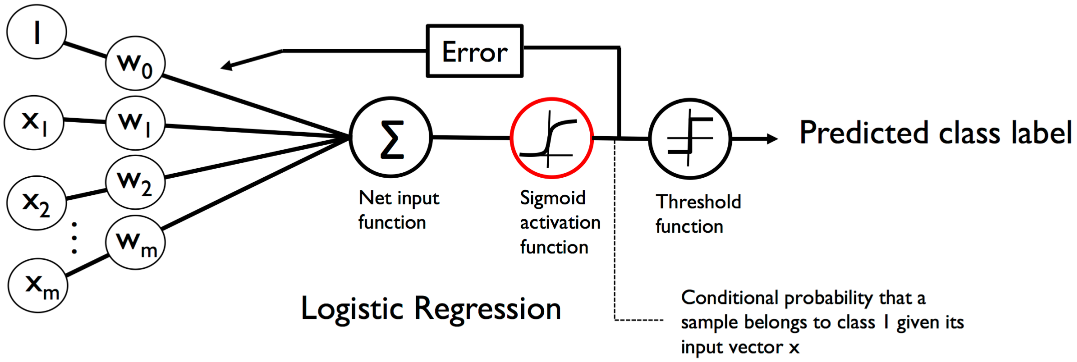
```
import matplotlib.pyplot as plt
import numpy as np

def sigmoid(z):
    return 1.0/(1.0+np.exp(-z))

z=np.arange(-7,7, 0.1)

phi_z=sigmoid(z)

plt.plot(z, phi_z)
plt.axvline(0.0, color='k')
plt.ylim(-0.1, 1.1)
plt.xlabel('z')
plt.ylabel('phi_z')

plt.yticks([0.0, 0.5, 1.0])
ax = plt.gca()
ax.yaxis.grid(True)

plt.show()
```



[15]:
```
Image(filename='images/03_03.png')
```

[15]:





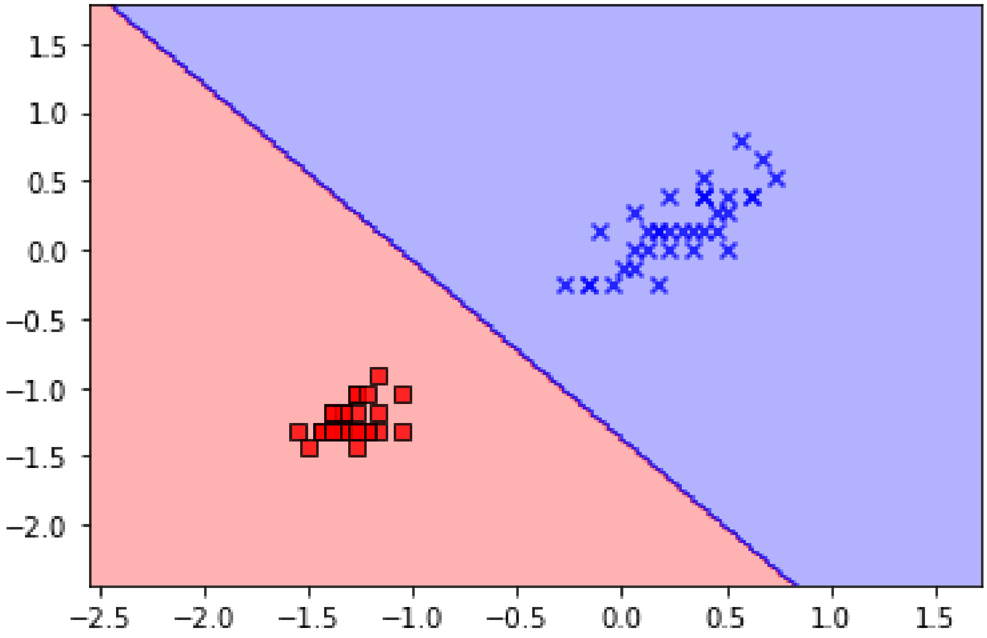
[18]:
```
class LogisticRegressionGD(object):

    def __init__(self, eta=0.05, n_iter=100, random_state=1):
        self.eta = eta
        self.n_iter = n_iter
        self.random_state=random_state

    def fit(self, X, y):
        rgen=np.random.RandomState(self.random_state)
        self.w_ = rgen.normal(loc=0.0, scale=0.01, size=1+X.shape[1])
        self.cost_ = []
        for i in range(self.n_iter):
            net_input=self.net_input(X)
            output = self.activation(net_input)
            errors = (y - output)
            self.w_[1:] += self.eta * X.T.dot(errors)
            self.w_[0] += self.eta * errors.sum()
            cost = -y.dot(np.log(output)) - ((1 - y).dot(np.log(1-output)))
            self.cost_.append(cost)
        return self

    def net_input(self, X):
        return np.dot(X, self.w_[1:]) + self.w_[0]

    def activation(self, z):
        return 1. / (1. +np.exp(-np.clip(z, -250, 250)))

    def predict(self, X):
        return np.where(self.net_input(X) >= 0.0, 1, 0)
```

[20]:
```
X_train_01_suset = X_train_std[(y_train == 0) | (y_train == 1)]
y_train_01_subset = y_train[(y_train == 0) | (y_train == 1)]

logreg = LogisticRegressionGD(eta=0.05, n_iter=1000, random_state=1)
logreg.fit(X_train_01_suset, y_train_01_subset)
```

[20]:
```
<__main__.LogisticRegressionGD at 0x7eff65107b80>
```

[21]:
```
plot_decision_regions(X=X_train_01_suset, y=y_train_01_subset,
                      classifier=logreg)
```

```
/tmp/ipykernel_145292/3603768161.py:28: UserWarning: You passed a
edgecolor/edgecolors ('black') for an unfilled marker ('x').  Matplotlib is
ignoring the edgecolor in favor of the facecolor.  This behavior may change in
the future.
  plt.scatter(x=X[y == cl, 0],
```

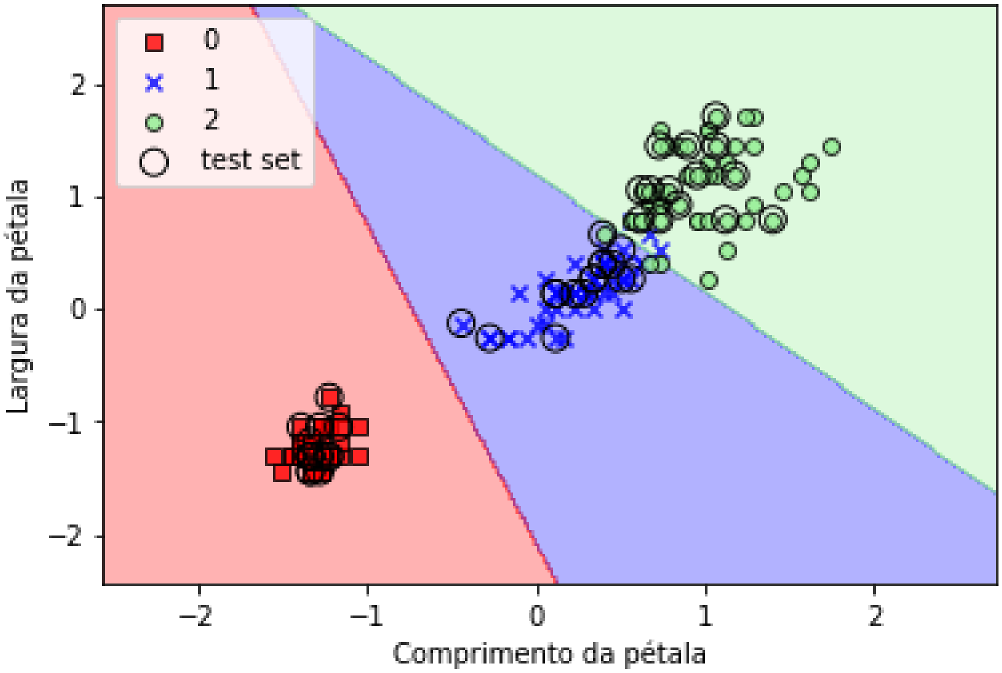


[22]:
```
from sklearn.linear_model import LogisticRegression

lr =LogisticRegression(C=100.0, random_state=1, solver='lbfgs',
                       multi_class='ovr')
lr.fit(X_train_std, y_train)
```

[22]:
```
LogisticRegression(C=100.0, multi_class='ovr', random_state=1)
```

[23]:
```
plot_decision_regions(X_combined_std, y_combined,
                      classifier=lr, test_idx=range(105, 150))

plt.xlabel('Comprimento da pétala')

plt.ylabel('Largura da pétala')

plt.legend(loc='upper left')

plt.show()
```

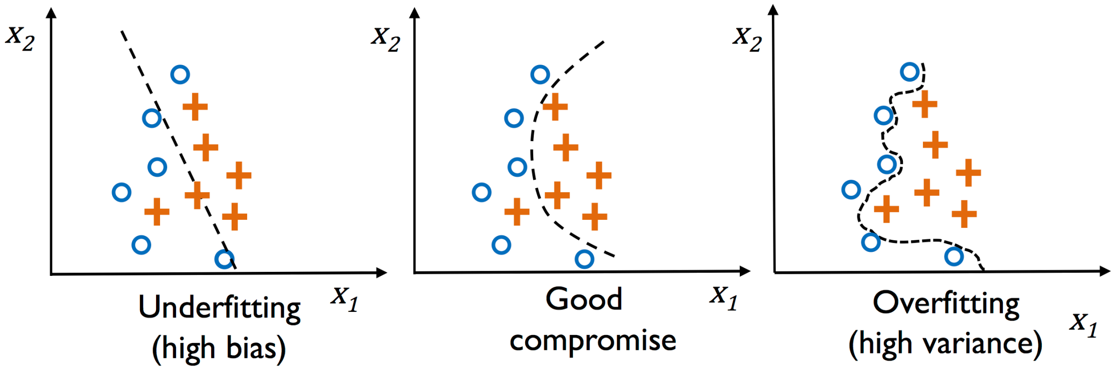
```
/tmp/ipykernel_145292/3603768161.py:28: UserWarning: You passed a
edgecolor/edgecolors ('black') for an unfilled marker ('x').  Matplotlib is
ignoring the edgecolor in favor of the facecolor.  This behavior may change in
the future.
  plt.scatter(x=X[y == cl, 0],
```



[24]:

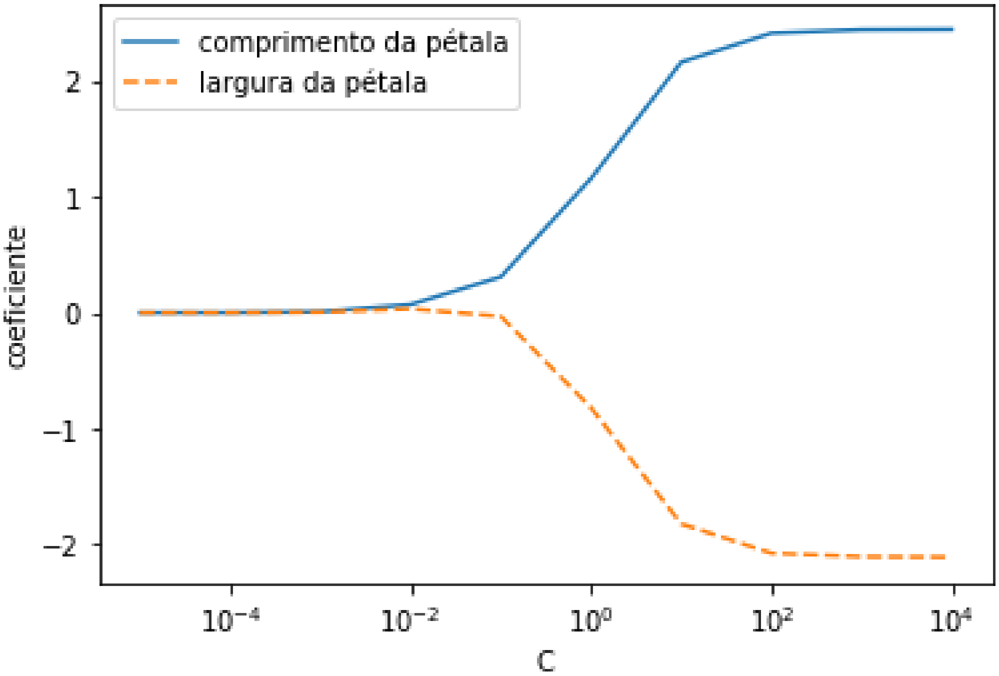
```
Image(filename='images/03_07.png')
```

[24]:



[25]:

```
weights, params = [], []
for c in np.arange(-5, 5):
    lr=LogisticRegression(C=10.**c, random_state=1, solver='lbfgs',
                          multi_class='ovr')
    lr.fit(X_train_std, y_train)
    weights.append(lr.coef_[1])
    params.append(10.**c)

weights = np.array(weights)
plt.plot(params, weights[:, 0],
         label='comprimento da pétala')
plt.plot(params, weights[:,1], linestyle='--',
         label='largura da pétala')
plt.xlabel('C')
plt.ylabel('coeficiente')
plt.legend(loc='upper left')
plt.xscale('log')
plt.show()
```


comprimento da pétala
largura da pétala
2
1
0
−1
−2
coeficiente
10⁻⁴
10⁻²
10⁰
10²
10⁴
C

```
[ ]:
```