MSc - Data Mining

Topic 06 : Classification

Part 02 : Logistic Regression

Dr Bernard Butler and Dr Kieran Murphy

Department of Computing and Mathematics, WIT.
(bbutler@tssg.org; kmurphy@wit.ie)

Spring Semester, 2021

Outline

- Relationship with linear regression
- (Sigmoid) Logistic function

Outline

Linear Regression I

Dataset

Take the IRIS dataset (4 features, target discrete with 3 levels $[0, 1, 2]$) and simplify it by* taking only the first feature, and merging class 1 ('versicolor') and class 2 ('virginica').

```
X, y, target_names = iris.data[:,:1], iris.target, iris.target_names
y[y>1] = 1
target_names = np.array([target_names[0], 'other'])
```

So we have target

```
y
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```

and target names of

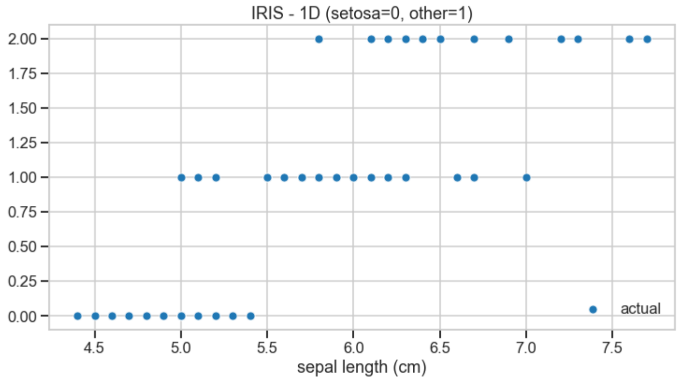
```
target_names
```

```
array(['setosa', 'other'], dtype='<U6')
```

*Python for Data Science — Cheat Sheet Numpy Basics

Linear Regression II

```python
plt.figure(figsize=(12,6))
sns.scatterplot(x=X_test.T[0],y=y_test, label="actual")
plt.xlabel(iris.feature_names[0])
plt.title("IRIS - 1D (setosa=0, other=1)")
plt.legend(loc="lower right")
plt.show()
```



From graph, it looks like we could predict class using sepal length, in particular if sepal length is <5.5 then predict `setosa` else predict `other`.
How many observations will be miss classified then?

Linear Regression III

Following standard procedure we split data ...

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.6, random_state=666)
```

... import and create instance of model (`LinearRegression`) ...

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
```

... fit model (using training data — feature(s) and target) ...

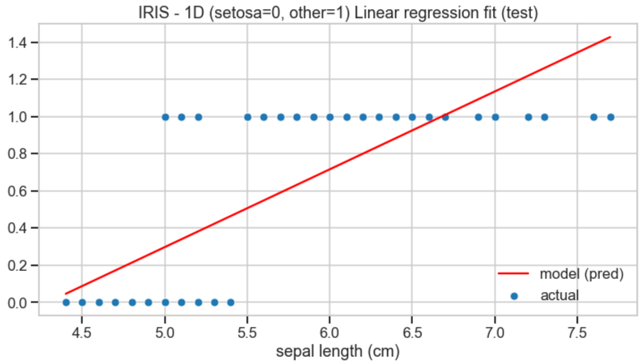
```
model.fit(X_train, y_train)
```

... predict (using feature(s) in test data) ...

```
y_pred = model.predict(X_test)
```

Linear Regression III

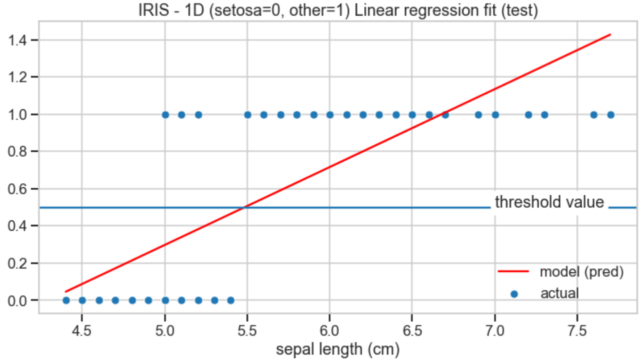
```
plt.figure(figsize=(12,6))
sns.scatterplot(x=X_test.T[0],y=y_test, label="actual")
sns.lineplot(x=X_test.T[0], y=y_pred, color='red', label="model (pred)")
plt.xlabel(iris.feature_names[0])
plt.title("IRIS - 1D (setosa=0, other=1) Linear regression fit (test)")
plt.legend(loc="lower right")
plt.show()
```



This is called a linear probability model. It has the advantages that much of the regression statistical theory can be applied to it.
But output is in range $(-\infty, \infty)$, we want 0 or 1

Linear Regression III

```
plt.figure(figsize=(12,6))
sns.scatterplot(x=X_test.T[0],y=y_test, label="actual")
sns.lineplot(x=X_test.T[0], y=y_pred, color='red', label="model (pred)")
plt.xlabel(iris.feature_names[0])
plt.title("IRIS - 1D (setosa=0, other=1) Linear regression fit (test)")
plt.legend(loc="lower right")
plt.show()
```

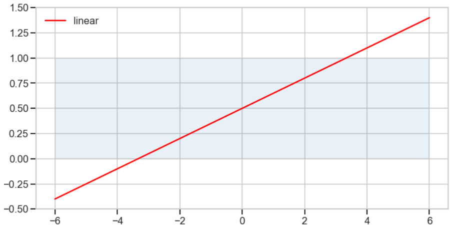


This is called a linear probability model. It has the advantages that much of the regression statistical theory can be applied to it.
But output is in range $(-\infty, \infty)$, we want 0 or 1

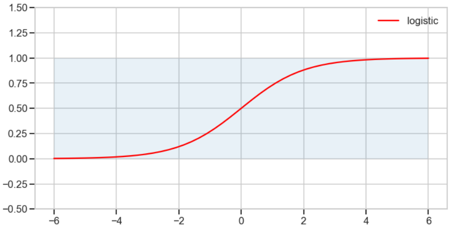
Pick a threshold value, default is 0.5.
If $y <$ `threshold`
    predict 0
else
    predict 1

...it would be nice ...

Can we ~~replace~~ map the infinite interval $(-\infty, \infty)$ to a finite interval, say to $(0, 1)$?



$\rightsquigarrow$



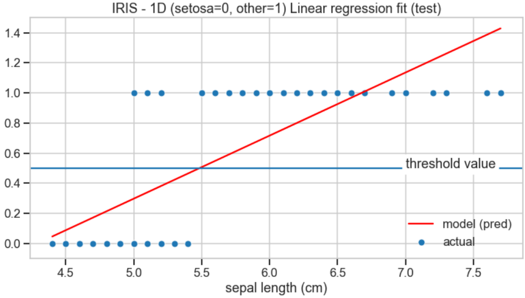
A Sigmoid function[†] is any function that has a stretched S-shaped curve. One example of a sigmoid function is the

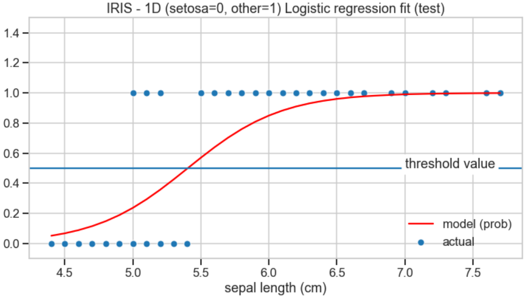
Logistic function, σ

$$\sigma(x) = \frac{1}{1+\exp(-x)} = \frac{1}{1+e^{-x}} = \frac{e^x}{1+e^x}$$

[†] Sigmoid function (Wikipedia), and Logistic function (Wikipedia),

Linear Regression vs Logistic Regression





Linear Regression Model

$$y = mx + c$$

Logistic Regression Model

$$y = \sigma(z) \quad \text{where} \quad z = mx + c$$

Aside: The function linking, z, output of linear step, to y, the model output is called a link function.

Of all the Sigmoid functions, why pick Logistic? I

First a little bit of mathematical manipulation . . .

$$
\begin{aligned}
y &= \frac{1}{1+\mathrm{e}^{-z}} && \text{definition of } \sigma(z) \\
y + y\mathrm{e}^{-z} &= 1 && \text{multiply both sides by } (1+\mathrm{e}^{-z}) \\
\mathrm{e}^{-z} &= \frac{1-y}{y} && \text{solve for } \mathrm{e}^{-z} \\
\mathrm{e}^{z} &= \frac{y}{1-y} && \text{invert both sides} \\
z &= \ln\left(\frac{y}{1-y}\right) && \text{apply logs}
\end{aligned}
$$

So we have

$$\boxed{y = \frac{1}{1+\mathrm{e}^{-z}} \quad\Longleftrightarrow\quad z = \ln\left(\frac{y}{1-y}\right)}$$

Of all the Sigmoid functions, why pick Logistic? II

- If y represents a probability, then
$$\frac{y}{1-y}$$
represent the odds — an alternative measure of the likelihood of a particular outcome[‡].
- Then
$$z = \ln\left(\frac{y}{1-y}\right)$$
is the log-odds, or the logit.
- This implies that increasing one of the independent variables multiplicatively scales the odds of the given outcome at a constant rate, with each independent variable having its own parameter. So the feature coefficients in a logistic regression have a similar interpretation (but not the same!) as in linear regression.

[‡]Odds are defined as the ratio of the number of events that produce that outcome relative to the number that do not.

Logistic Regression — Probability of being in Predicted Class

Unlike some classifiers, the `LogisticRegression` classifier can also report on the probability of an observation being in the predicted class[§].

```
y_prob = model.predict_proba(X_test)
y_prob
```

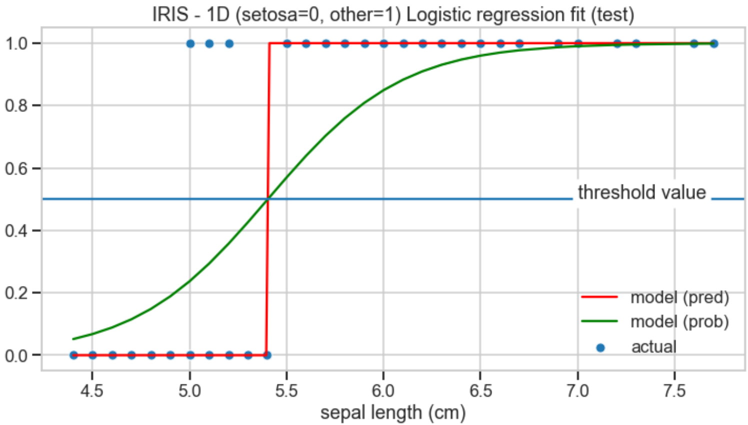
```
array([[0.36106274, 0.63893726],
       [0.24038055, 0.75961945],
       [0.76292166, 0.23707834],
       [0.24038055, 0.75961945],
       [0.91120554, 0.08879446],
       [0.43024916, 0.56975084],
       [0.29719846, 0.70280154],
       [0.03992405, 0.96007595],
       [0.06912596, 0.93087404],
       [0.43024916, 0.56975084],
       [0.02275676, 0.97724324],
       [0.93203415, 0.06796585],
```

```
y_log_prob = model.predict_log_proba(X_test)
y_log_prob
```

```
array([[-1.01870355e+00, -4.47949007e-01],
       [-1.42553199e+00, -2.74937692e-01],
       [-2.70599925e-01, -1.43936465e+00],
       [-1.42553199e+00, -2.74937692e-01],
       [-9.29867864e-02, -2.42143102e+00],
       [-8.43390801e-01, -5.62556133e-01],
       [-1.21335515e+00, -3.52680729e-01],
       [-3.22077634e+00, -4.07428849e-02],
       [-2.67182500e+00, -7.16313014e-02],
       [-8.43390801e-01, -5.62556133e-01],
       [-3.78289290e+00, -2.30196948e-02],
       [-7.03858210e-02, -2.68874994e+00],
```

[§]This is a big deal as it allows us to punish learners when they miss-classify based on how confident the classifier was — cross entropy loss functions (week 7 & 8).

Logistic Regression



Predicted	other	setosa
Actual		
other	38	3
setosa	0	19