

In a logistic regression model, we use a weighted sum of these two predictors to calculate a joint score S :

$$S = \beta_0 + \beta_1 X_1 + \beta_2 X_2. \quad (1)$$

The logistic regression model gives us appropriate values for the parameters β_0 , β_1 , β_2 using two sets of example genes:

- OP: Adjacent genes, on the same strand of DNA, known to belong to the same operon;
- NOP: Adjacent genes, on the same strand of DNA, known to belong to different operons.


```
print "Iteration: ", iteration, "Log-likelihood function: ", loglikelihood
>>>
>>> model = LogisticRegression.train(xs, ys, update_fn=show_progress)
Iteration: 0 Log-likelihood function: -11.7835020695
Iteration: 1 Log-likelihood function: -7.15886767672
Iteration: 2 Log-likelihood function: -5.76877209868
Iteration: 3 Log-likelihood function: -5.11362294338
Iteration: 4 Log-likelihood function: -4.74870642433
Iteration: 5 Log-likelihood function: -4.50026077146
Iteration: 6 Log-likelihood function: -4.31127773737
Iteration: 7 Log-likelihood function: -4.16015043396
Iteration: 8 Log-likelihood function: -4.03561719785
Iteration: 9 Log-likelihood function: -3.93073282192
Iteration: 10 Log-likelihood function: -3.84087660929
Iteration: 11 Log-likelihood function: -3.76282560605
Iteration: 12 Log-likelihood function: -3.69425027154
Iteration: 13 Log-likelihood function: -3.6334178602
Iteration: 14 Log-likelihood function: -3.57900855837
Iteration: 15 Log-likelihood function: -3.52999671386
Iteration: 16 Log-likelihood function: -3.48557145163
Iteration: 17 Log-likelihood function: -3.44508206139
Iteration: 18 Log-likelihood function: -3.40799948447
Iteration: 19 Log-likelihood function: -3.3738885624
Iteration: 20 Log-likelihood function: -3.3423876581
Iteration: 21 Log-likelihood function: -3.31319343769
Iteration: 22 Log-likelihood function: -3.2860493346
Iteration: 23 Log-likelihood function: -3.2607366863
```

3 Using the logistic regression model for classification

Classification is performed by calling the `classify`

