Logistic Regression

Multiclass Classification with Logistic Regression

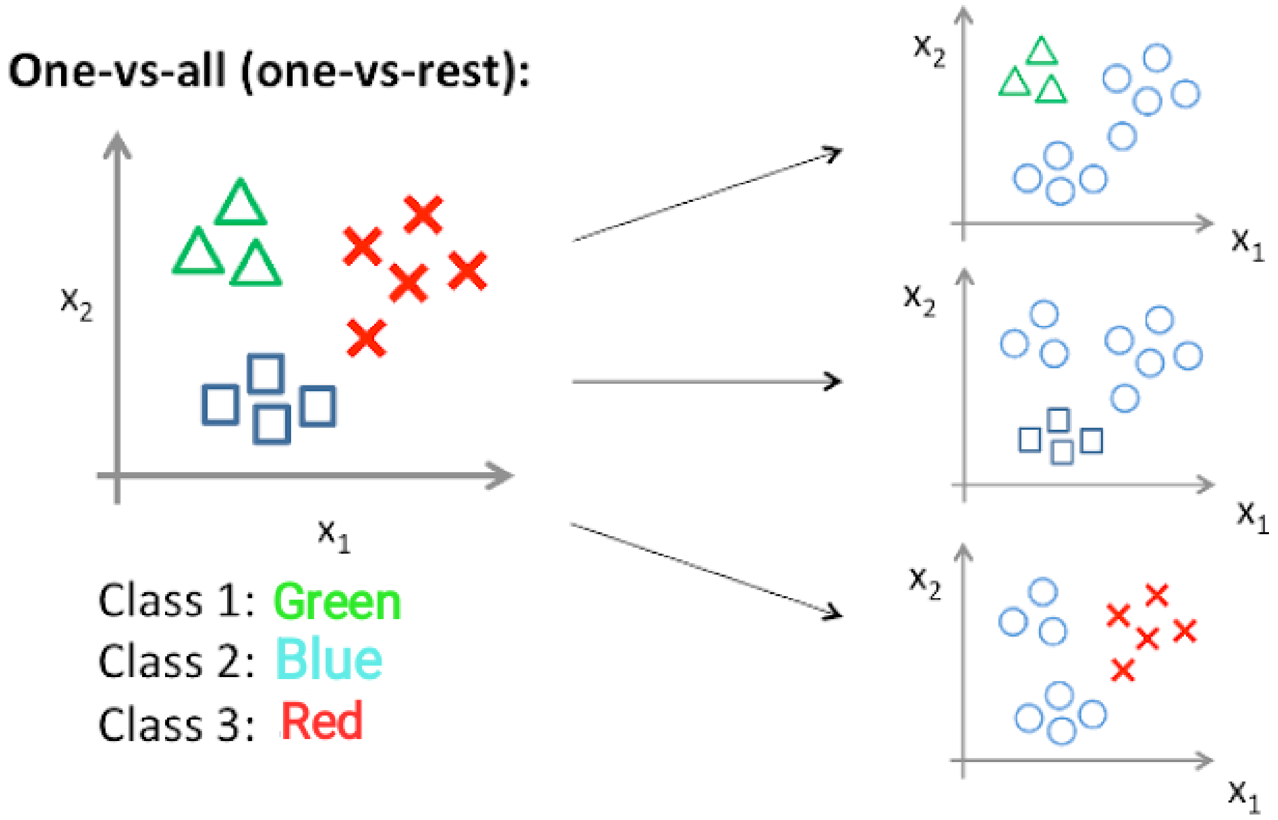
- We have understood that logistic regression, by nature, works for binary classification.

- What if we have more than 2 classes (i.e., multiclass/multinomial), can logistic regression algorithm still be applied?

- Answer is Yes!

- Question is how?!

Multiclass Classification with Logistic Regression

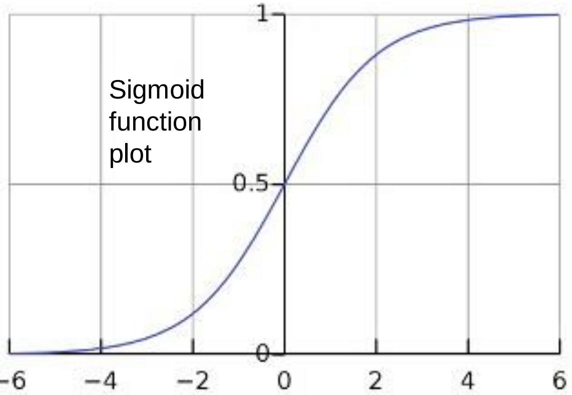


One-vs-Rest (OvR) approach

- Build k independent logistic regression models for k classes
- For example,
  1st model for Class 1 Green vs Rest
  2nd model for Class 2 Blue vs Rest
  3rd model for Class 3 Red vs Rest
- When a previously unseen instance comes in, to make a prediction, we need to run the 3 models and pick the class with the highest probability.
- Say if P(Green) > P(Blue) and P(Green) > P(Red), then predicted class is **Green**.

Multiclass Classification with Logistic Regression

- In the One-vs-Rest (OvR) approach, k models need to be built for k classes, which does not sound very efficient. Can we make some improvement, at least a bit?
- Recall that for 2 classes (e.g., classes 0 and 1), we only need to build 1 logistic regression model. Now for k classes, let's try to build k – 1 classes.



0.5 is the default threshold.

If predicted A_i (i.e., probability) ≥ 0.5, rounded to 1.

If predicted A_i (i.e., probability) < 0.5, rounded to 0.

For 2 classes (0 and 1)

$$Z = b + w_1x_1 + w_2x_2 + \cdots$$

$$A = \frac{1}{1+e^{-z}}$$

$$\frac{1}{A} = 1 + e^{-z}$$

$$\frac{1}{A} - 1 = e^{-z}$$

$$\frac{1-A}{A} = \frac{1}{e^z}$$

$$\frac{A}{1-A} = e^z$$

$$\ln\left(\frac{A}{1-A}\right) = Z$$

$$\ln\left(\frac{A(1)}{A(0)}\right) = Z$$

$A(1)$ is the probability of a sample belongs to class 1
$A(0)$ is the probability of a sample belongs to class 0

Now let's say 3 classes (0, 1 and 2)

1st model: *class* 0 *vs* 1

$$\ln\left(\frac{A(1)}{A(0)}\right) = Z^{(01)}$$
$$= b^{(01)} + w_1{}^{(01)}x_1 + w_2{}^{(01)}x_2 + \cdots$$

$$A(1) = A(0)e^{Z^{(01)}}$$

2nd model: *class* 0 *vs* 2

$$\ln\left(\frac{A(2)}{A(0)}\right) = Z^{(02)}$$
$$= b^{(02)} + w_1{}^{(02)}x_1 + w_2{}^{(02)}x_2 + \cdots$$

$$A(2) = A(0)e^{Z^{(02)}}$$

Important constraint:

$$A(0) + A(1) + A(2) = 1$$

$$A(0) + A(0)e^{Z^{(01)}} + A(0)e^{Z^{(02)}} = 1$$

So,

$$A(0) = \frac{1}{1 + e^{Z^{(01)}} + e^{Z^{(02)}}}$$

$$A(1) = \frac{e^{Z^{(01)}}}{1 + e^{Z^{(01)}} + e^{Z^{(02)}}}$$

$$A(2) = \frac{e^{Z^{(02)}}}{1 + e^{Z^{(01)}} + e^{Z^{(02)}}}$$

Regularization

- $\min_{w,b} J(w,b)$, where $J = L$, another common symbol to denote loss/error in ML
- Parameters: w ∈ $\mathbb{R}^{n_x}$, b ∈ $\mathbb{R}$, $\boldsymbol{\lambda}$ = regularization parameter, n_x = no. of features
- J(w, b) = $\frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}\|w\|_2^2$

  w is a vector of all weights

$\hat{y}^{(i)}$ is the predicted value of the i^{th} sample
$y^{(i)}$ is the true value of the i^{th} sample
L is the Cross Entropy Loss function we used previously

- L_2 regularization: $\|w\|_2^2 = \sum_{j=1}^{n_x} w_j^2 = w^T w$
- L_1 regularization: $\|w\|_1 = \sum_{j=1}^{n_x} |w_j|$

$\sum_{j=1}^{n_x} w_j^2 = w^T w = [w_1\ w_2\ w_3]\begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}$

$= w_1^2 + w_2^2 + w_3^2$

$\sum_{j=1}^{n_x} |w_j| = |w_1| + |w_2| + |w_3|$

DigiPen INSTITUTE OF TECHNOLOGY SINGAPORE

Regularization

Loss function with L2 regularization

$$J(w, b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}\|w\|_2^2$$

Find derivatives of loss w.r.t weights, similar to what we did previously in the selling house example.
Only differences lies in the introduction of L2 regularization term here

$$\frac{\partial J}{\partial w} = dw = (from\ backprop) + \frac{\lambda}{m} w_{old}$$

Gradient descent

$$w_{new} = w_{old} - \alpha dw = w_{old} - \alpha[(from\ backprop) + \frac{\lambda}{m} w_{old}]$$

$$w_{new} = w_{old} - \alpha \frac{\lambda}{m} w_{old} - \alpha(from\ backprop)$$

$$w_{new} = (1 - \alpha \frac{\lambda}{m}) w_{old} - \alpha(from\ backprop)$$

<1, so called "Weight decay"

Regularization

- **Essentially what regularization does is adding extra penalty to complicated model with higher values of weights.**

- That's why regularization is an efficient technique to prevent overfitting and also an important hyperparameter in ML.

- Additional Reading
  - The *penalty* hyperparameter in sklearn logistic regression
  - For linear regression, check Lasso for L1 regularization and Ridge for L2 regularization

DigiPen
INSTITUTE OF TECHNOLOGY
SINGAPORE