Logistic Regression

16BCE1259
Shushil Kumar Ravishankar

What is Logistic Regression?

In regression analysis, **logistic regression** (or **logit regression**) is estimating the parameters of a logistic model; it is a form of binomial regression. Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail, win/lose, alive/dead or healthy/sick; these are represented by an indicator variable, where the two values are labelled "0" and "1".

Observations on Big Mart Sales Data

- When all the attributes are considered for the model, I got an accuracy of 98%. But this may have an over fitted model
- I tried removing attributes that contribute least to the accuracy of the model.
- It was found that `'Item_Identifier', 'Item_Weight', 'Item_Fat_Content', 'Item_Visibility', 'Outlet_Identifier', 'Outlet_Establishment_Year', 'Outlet_Size', 'Outlet_Type', 'Item_Outlet_Sales'` contribute more to the accuracy than others.
- It was also found that removing `'Item_Identifier', 'Outlet_Establishment_Year', 'Item_Outlet_Sales'` the accuracy of the model increases. This means that these attributes may be cause of overfitting of the model.
- `Outlet_Identifier' and 'Outlet_Type'` contribute the most to the accuracy.
- Using these two attributes, decision regions are plotted for different parameters.
- All the decision boundaries give an accuracy of 100% when these two parameters are used.

Comparison between the methods

A. **Newton's Method**

Newton's method uses in a sense a ***better*** quadratic function minimisation. A better because it uses the quadratic approximation (i.e. first AND *second* partial derivatives).

Moreover, the geometric interpretation of Newton's method is that at each iteration one approximates `f(x)` by a quadratic function around x_n, and then takes a step towards the maximum/minimum of that quadratic function (in higher dimensions, this may also be a saddle point). Note that if `f(x)` happens to be a quadratic function, then the exact extremum is found in one step.

Drawbacks:

1. It's computationally ***expensive*** because of The Hessian Matrix (i.e. second partial derivatives calculations).
2. It attracts to ***Saddle Points*** which are common in multivariable optimization (i.e. a point its partial derivatives disagree over whether this input should be a maximum or a minimum point!).

B. **Limited-memory Broyden–Fletcher–Goldfarb–Shanno Algorithm:**

In a nutshell, it is analogue of the Newton's Method but here the Hessian matrix is ***approximated*** using updates specified by gradient evaluations (or approximate gradient evaluations). In other words, using an estimation to the inverse Hessian matrix.
The term Limited-memory simply means it stores only a few vectors that represent the approximation implicitly.

When dataset is ***small***, L-BFGS relatively performs the best compared to other methods especially it saves a lot of memory, however there are some "*serious*" drawbacks such that if it is unsafeguarded, it may not converge to anything.

C. **A Library for Large Linear Classification:**

It's a linear classification that supports logistic regression and linear support vector machines (*A linear classifier achieves this by making a classification decision based on the value of a linear combination of the characteristics i.e. feature value*).
The solver uses a coordinate descent (CD) algorithm that solves optimization problems by successively performing approximate minimization along coordinate directions or coordinate hyperplanes.

It applies *Automatic parameter selection* (a.k.a L1 Regularization) and it's recommended when you have high dimension dataset (*recommended for solving large-scale classification problems*)

Drawbacks:

1. It may get stuck at a *non-stationary point* (i.e. non-optima) if the level curves of a function are not smooth.
2. Also cannot run in parallel.
3. It cannot learn a true multinomial (multiclass) model; instead, the optimization problem is decomposed in a "one-vs-rest" fashion so separate binary classifiers are trained for all classes.

D. **Stochastic Average Gradient:**

SAG method optimizes the sum of a finite number of smooth convex functions. Like stochastic gradient (SG) methods, the SAG method's iteration cost is independent of the number of terms in the sum. However, by ***incorporating a memory of previous gradient values the SAG method achieves a faster convergence rate*** than black-box SG methods.
It is **faster** than other solvers for *large* datasets, when both the number of samples and the number of features are large.

Drawbacks:

1. It only supports L2 penalization.
2. Its memory cost of `O(N)`, which can make it impractical for large N (*because it remembers the most recently computed values for approx. all gradients*).

E. **SAGA:**

The SAGA solver is a *variant* of SAG that also supports the non-smooth *penalty=l1* option (i.e. L1 Regularization). This is therefore the solver of choice for ***sparse*** multinomial logistic regression and it's also suitable ***very Large*** dataset.