Binary Logistic Regression

Dataset preparation:

1. Use dataset [diabetes](). Code for loading dataset into 2D python list: [here]()
2. Randomly Split the dataset into Training (70%), Validation (15%) and Test (15%) set

Train (update **Θ**):

```
1.  for each sample, X = [x1, x2, … , xn]  in TRAINING set:
2.          concatenate 1 and turn it into X’ = [x1, x2, …, xn, 1]
3.  randomly initialize Θ = [Θ1, Θ2, …, Θ(n+1)] within 0 to 1
                                                    // Θ1, Θ2, …: weights, Θ(n+1): bias
4.  max_iter = 100, lr = 0.01
5.  history = list()
6.  for itr in [1, max_iter]:
7.          TJ = 0                                  // total cost
8.          for each sample, X’, in TRAINING set:
9.                  z = X’ . Θ                      // use np.dot function
10.                 h = sigmoid(z)                  // sigmoid available in python
11.                 J = - y log (h) - (1-y) log (1-h)   // h = pred label, y = true label
12.                 TJ = TJ + J
13.                 dv = X’ . (h-y)                 // dim(dv)=n+1
14.                 Θ = Θ - dv * lr                 // dim(Θ)=n+1, lr = learning rate
15.         TJ = TJ / N_train                       // N_train = #training samples
16.         append TJ into history                  // average loss
```

Validation:

```
1.  correct = 0
2.  for each sample V in the VALIDATION set:
3.          z = V’.Θ
4.          h = sigmoid(z)
5.          if h >= 0.5:    h = 1
6.          else:           h = 0
7.          if h == y:      correct = correct + 1
8.   val_acc = correct * 100 / N_val                // N_val = #validation samples
```

- [ ] Calculate validation accuracy (val_acc) for lr = 0.1, 0.01, 0.001 and 0.0001 (max_iter = 500)
- [ ] Make a table with 2 columns: learning rate lr and val_acc
- [ ] Now, take the lr with maximum val_acc
- [ ] Calculate $test\ accuracy$ for max_iter = 500 and the **chosen lr** in the previous step

Instruction

- Submit a .ipynb file and a report ([report template]) .pdf file.
- **DO NOT USE LIBRARIES SUCH AS: "Sklearn", "Scikit learning" or "pandas" for this assignment**
- **Copying will result in -100% penalty**

Marks Distribution

(1) Dataset loading, train-val-test split: 2
(2) Training code: 8
(3) Validation/ test code: 5
(4) l.r. and val_acc table: 2.5
(5) train_loss vs epoch graph plot for the best l.r.: 2.5
Task (2)-(5) have to be done without using sklearn like libraries.
Your marks will fully depend on your viva and understanding.

Resources

Logistic_Regression_CRR.pdf

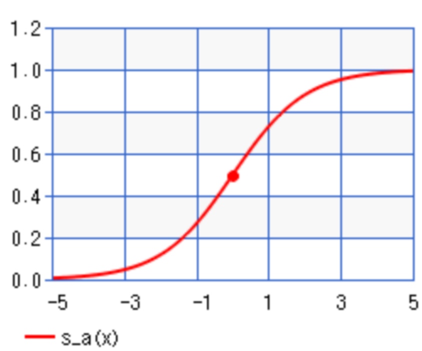
Labels = 0 or 1 ⇒ binary classification

How to predict?

Let, sample 1 of dataset, X_1 = [x1, x2, x3, 1]

Weights, **θ** = [**θ**1, **θ**2, **θ**3, **θ4**]   **θ4 is called bias**

Model/Prediction equation: ***z = X.θ = x1.θ1 + x2.θ3 + x3.θ3 + θ4***. We update weights **θ** so that z can correctly predict the label of X_1, but its value can be very big (>1) or very small (<0).



Solution: use activation function sigmoid

sigmoid(z) = $1/(1+e^{-z})$

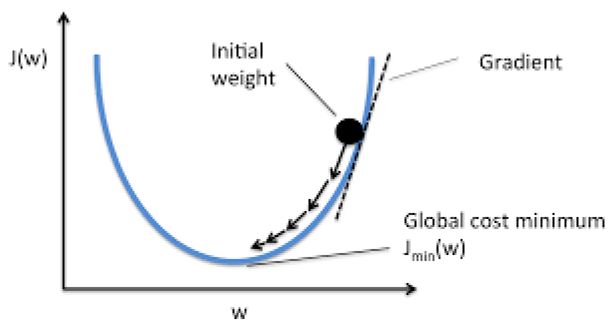
So, *h = sigmoid(z) is the predicted label of X1*

How to update weights?

Gradient descent optimization

Log loss function: J(**θ**) = - y log(h) - (1-y) log(1-h) , y is the true label and h is the predicted label

The closer h is to y, the lesser the loss.



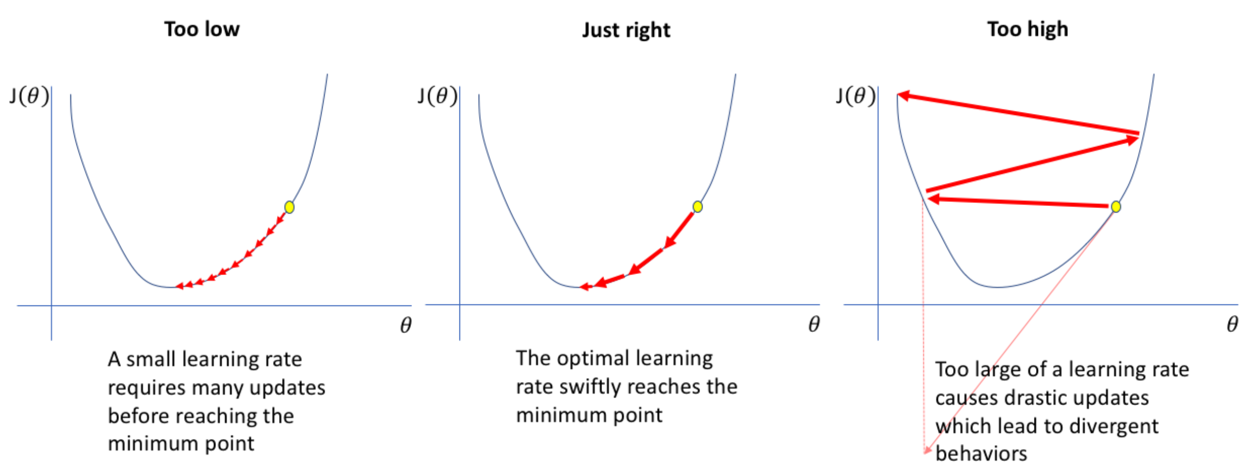
dv = Derivative of J(**θ**) = Gradient = X(h-y)

If gradient +ve, we should decrease weights, else if gradient -ve, we should increase weights.
So, update **θ** = **θ** - dv

However, weights may oscillate without reaching our desired value.

Solution: introduce learning rate lr **(0<lr<1) e.g. 0.01, 0.001, 0.0001**

θ = **θ** - dv * lr



Weights are updated using the training set.

How to choose the value of lr?

Hyperparameter tuning using validation set.