Assign-3

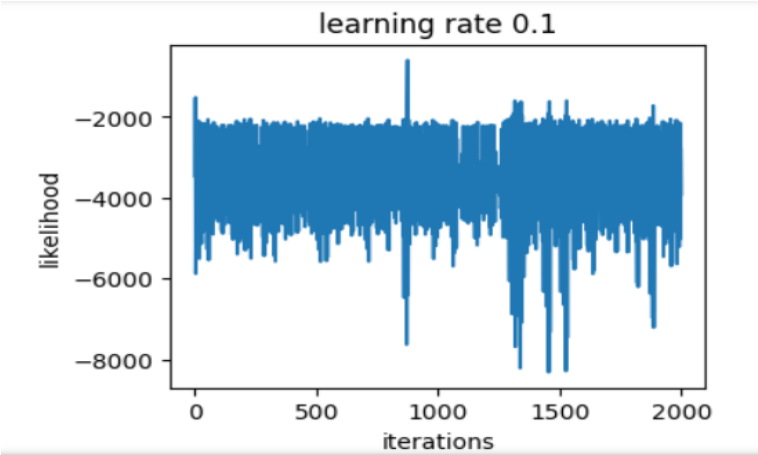
Surendra Parla

February 2024

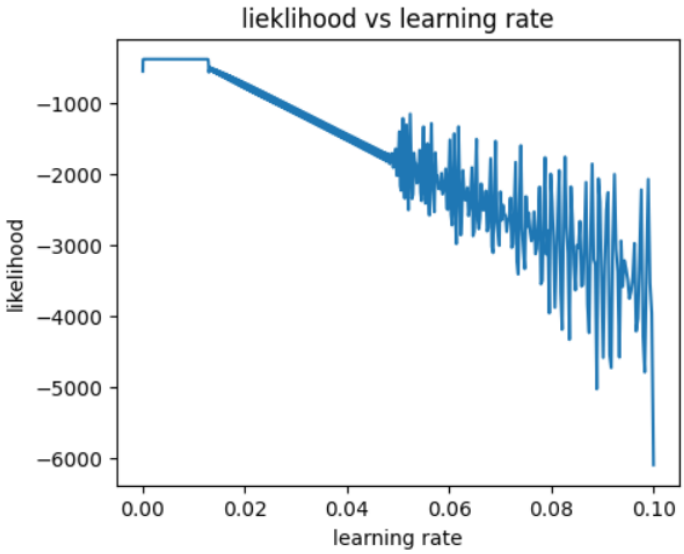
1 1

1.1 a

Start with a large learning rate like 1 and 0.1. We could see that log-likelihood is quite unstable (highly fluctuating) which points out that we are just alternating on either side and not converging.



Now, start 0.1 and for each iteration increase alpha increases linearly in the exponential space (0.1 to 0.0001) and plot the likelihood function.



From the plot we can see there are 3 phases, when alpha is between 0 and 0.02 the training is so slow and if alpha is between 0.05 and 0.1 training is very unstable. So we can choose alpha as 0.001 for first few iterations and in final stages of training we can decrease it to 0.0001

As, the dataset is small learning rate of 0.001 worked significantly well

1.2 b

With proper normalization, the model converged in 30 iterations

1.3 c

Final value of $\hat{\theta}$ is [-0.88150697, 1.35276923, -0.56007237, -0.37063885, -0.13861887,0.23049696]

1.4 d

Maximum likelihood l(θ) is -374.6

1.5 e

The asymptotic distribution of θ is a normal distribution with mean $\hat{\theta}$
[-0.88150697, 1.35276923, -0.56007237, -0.37063885, -0.13861887,0.23049696]
variance:

$$\begin{pmatrix} 78.82548935 & 4.88108223 & 15.71643104 & 18.15891713 & -29.45507336 & 144.69437922 \\ 4.88108223 & 3.35126473 & 7.077103 & 6.40572294 & 3.94964209 & -2.42939071 \\ 15.71643104 & 7.077103 & 46.21584323 & 29.31906738 & 16.17231909 & -23.0911871 \\ 18.15891713 & 6.40572294 & 29.31906738 & 31.3669466 & 9.52833911 & -7.46410436 \\ -29.45507336 & 3.94964209 & 16.17231909 & 9.52833911 & 38.07379323 & -93.99912166 \\ 144.69437922 & -2.42939071 & -23.0911871 & -7.46410436 & -93.99912166 & 350.71045394 \end{pmatrix}$$

2 2

2.1 a

log odds

$$\begin{aligned} \omega^\star = \log \frac{\mathbb{P}(y=1|x)}{\mathbb{P}(y=0|x)} &= \frac{\frac{1}{(1+e^{-\theta^T x_i})}}{\frac{1}{(1+e^{\theta^T x)}}} \\ &= \frac{(1+e^{\theta^T x)}}{(1+e^{-\theta^T x)}} \\ &= \theta^{\star T} X \end{aligned}$$

where $\theta^{\star T}$ is [-0.88150697, 1.35276923, -0.56007237, -0.37063885, -0.13861887,0.23049696] and X is the new sample
$variance = Var(\theta^{\star T} X) = X^T Var(\theta^{\star T}) X = X^T I_{\theta^*}^{-1} X$

3 3

3.1 a

Lets assume I am on the titanic my feature vector would be X = $\begin{pmatrix} 2 \\ 0 \\ 25 \\ 0 \\ 0 \\ 20 \end{pmatrix}$

Y_{new} (probability of survival is 0.42179714) is 0.
So, I would have been deceased in the titanic.

3.2 b

Variance is $X_{new}^T I_{\theta^*}^{-1} X = 85.73272155$
$\sigma = 9.25919659$
$\tau = 1.96 * \sigma / \sqrt{800}$
$\tau = 0.72949679$

3.3 c

$\hat{\omega} = \theta^T X = -0.31540031$
Interval for log-odds is (-0.31540031 ± 0.72949679)
So, odds are in the interval [0.36787944, 1.4918247] which shows that we may or may not have survived.

4 4

4.1 a

With p-value $= \Phi_{\chi^2}\left(\frac{\widehat{\theta}_j^2}{\nu_j^2}\right)$ we conclude that the j^{th} feature is significant if $\widehat{\theta}_j^2 > \nu_j^2 \Phi_{\chi^2}^{-1}(\alpha)$ (1)

4.2 b

$\nu_j^2 = \begin{pmatrix} 78.82548 & 3.35126 & 46.21584 & 31.36694 & 38.07379 & 350.71045 \end{pmatrix}$
$\widehat{\theta}_j^2 = \begin{pmatrix} 0.77705454 & 1.82998458 & 0.31368106 & 0.13737316 & 0.01921519 & 0.05312885 \end{pmatrix}$
$\left(\frac{\widehat{\theta}_j^2}{\nu_j^2}\right) = \begin{pmatrix} 9.85790955e-03 \\ 5.46057899e-01 \\ 6.78730574e-03 \\ 4.37955151e-03 \\ 5.04682851e-04 \\ 1.51489203e-04 \end{pmatrix}$ $\Phi_{\chi^2}^{-1}(\alpha) = 0.00393$ for 1 dof and 95% significance

Therefore features Pclass, Sex, Age, Siblings/Spouses Aboard are significant and features Parents/Children Aboard and Fare are insignificant

4.3 c

Sex is the most significant feature.
So, changing the feature and calculating the Y_{new} ($wegetprobability 0.5262661$)
$= 1$.
I would have survived if we change the significant feature.

5 Appendix

Accuracy by 90:10 split is around 80%.

```
split_ratio = 0.9
num_samples = df.shape[0]
num_train = int(num_samples * split_ratio)
num_test = num_samples - num_train
y = df.iloc[:, 0].astype(np.int32).to_numpy()
Y_train, Y_test = y[:num_train], y[num_train:]
x = df.iloc[:, 1:7].astype(np.float64).to_numpy().T
```

```
X_train, X_test = x[:,:num_train], x[:,num_train:]
mean = X_train.mean(axis=1).reshape([6,1])
std = X_train.std(axis=1).reshape([6,1])
X_train, X_test = (X_train-mean)/std, (X_test-mean)/std

def sigmoid(x):
  return 1 / (1 + np.exp(-x))
```