Homework 4: ML

Vishakh Padmakumar
DS-GA 1003 · Spring 2021 · NYU Center for Data Science

1 Logistic Regression

Q1. Show that the two approaches are equivalent, i.e. they will produce the same solution for w.

ERM:
Hypothesis space, $F_{score} : \{x \rightarrow w^T x \mid w \in \mathbb{R}^d\}$
Predictions, $\hat{y} = sign(x^T w)$
Outcome space, $Y : +1/-1$
Loss function, $l(y, w) = \log(1 + exp(-yw^T x))$

Chosen $\hat{f} = \underset{f \in F_{score}}{\arg\min}\, \hat{R}_n(f) = \underset{f \in F_{score}}{\arg\min}\, \frac{1}{n} \sum_{i=1}^{n} l(f(x_i), y_i)$

Therefore chosen weights $\hat{w} = \underset{f \in F_{score}}{\arg\min}\, \frac{1}{n} \sum_{i=1}^{n} l(f(x_i), y_i) = \underset{w \in \mathbb{R}^d}{\arg\min}\, \frac{1}{n} \sum_{i=1}^{n} \log(1 + exp(-y_i w^T x_i))$

$= \underset{w \in \mathbb{R}^d}{\arg\min} \sum_{i=1}^{n} \log(1 + exp(-y_i w^T x_i))$ **... (1)**

MLE with Bernoulli response distribution and logistic link function:
$p(y = 1|x; w) = \frac{1}{1+exp(-w^T x)}$

$p(y = -1|x; w) = 1 - p(y = 1|x; w) = \frac{exp(-w^T x)}{1+exp(-w^T x)}$

Consider function $f, f(y, x, w) = \frac{1}{1+e^{-yw^T x}}$.

We can see that $f(1, x, w) = \frac{1}{1+exp(-w^T x)}$ and that $f(-1, x, w) = \frac{1}{1+exp(w^T x)} = \frac{exp(-w^T x)}{1+exp(-w^T x)}$
We use this to just observe that f corresponds to the pmf at $y = 1$ and $y = -1$.

Therefore $p(y = y_i|x_i, w) = \frac{1}{1+exp(-y_i w^T x_i)}$

Likelihood, $L(D, w) = \prod_{i=1}^{n} p(y_i|x_i, w) = \prod_{i=1}^{n} \frac{1}{1+exp(-y_i w^T x_i)}$

Log likelihood, $LL(D, w) = \sum_{i=1}^{n} \log(\frac{1}{1+exp(-y_i w^T x_i)}) = -\sum_{i=1}^{n} \log(1 + exp(-y_i w^T x_i)))$

MLE, $\hat{(}w) = \underset{w \in \mathbb{R}^d}{\arg\max}\, LL(D, w)$

$$= \arg\max_{w \in \mathbb{R}^d} - \sum_{i=1}^{n} \log(1 + exp(-y_i w^T x_i)))$$

$$= \arg\min_{w \in \mathbb{R}^d} \sum_{i=1}^{n} \log(1 + exp(-y_i w^T x_i))) \textbf{ ... (2)}$$

We can see that the obtained results **(1)** and **(2)** that the two approaches yield the same result

Q2. Show that the decision boundary of logistic regression is given by $\{x : x^T w = 0\}$. Note that the set will not change if we multiply the weights by some constant c.

At the decision boundary, the likelihood of the point x being put into either category of y is equal i.e. $p(y = 1|x, w) = p(y = -1|x, w)$.
From the previous question, we can substitute in the values: $\frac{1}{1+exp(-w^T x)} = \frac{exp(-w^T x)}{1+exp(-w^T x)}$ or $exp(-w^T x) = 1$ implying $w^T x = 0$
Therefore the decision boundary is given by all points x such that $\{x : x^T w = 0\}$ and is unchanged when we multiply w by some c

Q3. Suppose the data is linearly separable and by gradient descent/ascent we have reached a decision boundary defined by $\hat{w}$ where all examples are classified correctly. Show that we can always increase the likelihood of the data by multiplying a scalar c on $\hat{w}$, which means that MLE is not well-defined in this case.

Consider log likelihood $LL(D, \hat{w}) = -\sum_{i=1}^{n} \log(1 + exp(-y_i \hat{w}^T x_i))$

To find the MLE, we maximize this likelihood. If we scale the weights by a scalar c to $c\hat{w}$, $LL(D, c\hat{w}) = -\sum_{i=1}^{n} \log(1 + exp(-y_i c\hat{w}^T x_i))$

To observe the relationship with c, we take derivative:
$\frac{dLL(D,c\hat{w})}{dc} = -\sum_{i=1}^{n} \frac{1}{1+exp(-y_i c\hat{w}^T x_i)} \times exp(-y_i c\hat{w}^T x_i) \times -y_i \hat{w}^T x_i$
$= \sum_{i=1}^{n} \frac{1}{1+exp(-y_i c\hat{w}^T x_i)} \times exp(-y_i c\hat{w}^T x_i) \times y_i \hat{w}^T x_i$

We are given that at this $\hat{w}$, all points are classified correctly, so the margin term $y_i \hat{w}^T x_i$ is always positive. So the derivative of the log likelihood w.r.t. c is always positive and so the log likelihood as a function of c is increasing and hence increasing c will always increase our MLE. Hence the MLE is not well defined.

Q4. Prove that the L2 regularized logistic regression is convex.

We can write the first term in the regularized loss $\log(1 + exp(-y_i w^T x_i))$ as a composition of two functions.

Let $g(w) = y_i w^T x_i$ and $f(a) = \log(1 + exp(-a))$. So the first term of the loss is $f(g(w))$.

Consider the second derivative of f: $\frac{d^2}{da^2}f(a) = \frac{exp(a)}{(1+exp(a))^2}$. This is non-negative at all points, so f is convex.

We also know that g is an affine function i.e. it is convex and concave. Since f is convex and g is affine, $f(g(w))$ is also convex. We also know that $\lambda||w||^2$ is convex so their sum, i.e. the L2 regularized logistic regression objective, is a convex function.

Q5. Complete the $f_{objective}$ function in the skeleton code, which computes the objective function for $J_{logistic}(w)$.

```
def f_objective(theta, X, y, l2_param=1):
    objective = 0
    for i in range(len(X)):
        temp = np.logaddexp(0, -y[i] *(theta @ X[i]))
        objective+=temp

    objective/=len(X)
    objective+=l2_param*(np.linalg.norm(theta)**2)
    return objective
```

Q6. Complete the fit logistic regression function in the skeleton code using the minimize function from scipy.optimize. Use this function to train a model on the provided data. Make sure to take the appropriate preprocessing steps, such as standardizing the data and adding a column for the bias term

```
def fit_logistic_reg(X, y, objective_function, l2_param=1):
    fn = functools.partial(objective_function, X=X, y=y, l2_param=l2_param)
    init = np.zeros(X.shape[1])
    res = minimize(fn, init)
    return res.x

if __name__=="__main__":
    X_train = read_file("X_train.txt")
    y_train = read_file("y_train.txt")
    X_val = read_file("X_val.txt")
    y_val = read_file("y_val.txt")

    #Standardizing input to have zero mean and unit variance

    standardize = StandardScaler()
    standardize.fit(X_train)
```

```
    #print(X_train[0])
    print("Before standardize ", np.mean(X_train[:, 0]), np.std(X_train[:, 0]))

    X_train = standardize.transform(X_train)
    X_val = standardize.transform(X_val)

    #print(X_train[0])
    print("After Standardize ", np.mean(X_train[:, 0]), np.std(X_train[:, 0]))

    #Add bias term of ones to both train and validation data

    X_train = np.hstack((X_train, np.ones((X_train.shape[0], 1))))
    X_val = np.hstack((X_val, np.ones((X_val.shape[0], 1))))

    #print(X_train.shape)
    #print(X_val.shape)

    #Changing {0, 1} true labels to label space of {-1, 1}

    print(y_train[:5])
    y_train = np.where(y_train == 0, float(-1), y_train)
    y_val = np.where(y_val == 0, float(-1), y_val)
    print(y_train[:5])

    # Calling fit function

    w = fit_logistic_reg(X_train, y_train, f_objective, 1)
```

Q7. Find the l2 regularization parameter that minimizes the log-likelihood on the validation set. Plot the log-likelihood for different values of the regularization parameter.

The best l2 parameter we obtain is at 0.01. The relevant code that obtains the same is provided below along with the corresponding output.

Output for Q7:

```
[(1e-05, -234.8091130502293), (0.001, -234.5733020043379),
    (0.01, -233.3258178272187), (0.05, -234.40865491072563),
    (0.1, -238.28200924440827), (0.5, -256.3444974899046),
    (1, -264.0504423888256), (10, -275.5212361913735),
    (100, -277.0794024381505)]

Best l2 param at 2 - 0.01
```

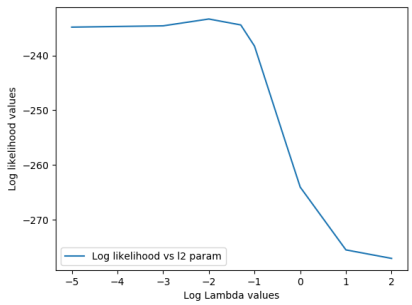

Figure 1: Plot of log likelihood for different values of regularization paramter. Note that the x axis is also on a log scale for λ

Q8. To see how well-calibrated our predicted probabilities are, break the predictions on the validation set into groups based on the predicted probability (you can play with the size of the groups to get a result you think is informative). For each group, examine the percentage of positive labels. You can make a table or graph. Summarize the results. You may get some ideas and references from scikit-learn's discussion.

Fig. 2 is the required figure when we break predictions into 10 buckets, and Fig. 3 is the same graph but for 20 buckets. We break up the predictions on the validation sets into buckets, each of width 0.1 in the first case and 0.05 in second case. And we use scikit-learn's calibration curve to give us the proportion of samples that are positive label and the mean predicted probability in each bin. The orange line indicates the mean predicted probability in each bin so it is understandable that this would be essentially a straight line graph. By in large the fraction of positives in each bin also increases as the bins approach 1 in both cases. A well calibrated classifier would resemble the straight line even for the fraction of positives in each bucket. We see that the calibration falls off the expected pattern at buckets close to 1 and close to 0. But these often also have a smaller fraction of examples in them as seen in the output. It's hard to say in isolation if the predictions are calibrated but it is encouraging that the overall trend does resemble an increasing one. The fact that predictions close to 0.5 are where the model is often miscalibrated is understandable, we can observe this in Fig. 3 where buckets just below 0.5 have a much lower fraction of positives than the ones slightly above 0.5.

2 Coin Flipping with Partial Observability

Q9. Show that $p(x = H|\theta_1, \theta_2) = \theta_1\theta_2$

We need to find the probability that a head was reported at x given parameters θ_1 and θ_2.

z is unaffected by θ_2 i.e. the true value is not affected by θ_2. And the reporting i.e. x given z is a head is only conditional on θ_2.

So the required probability is the product of probability that the true value z is actually a head given paramter θ_1 times the probability that it is also reported as a head given parameter θ_2

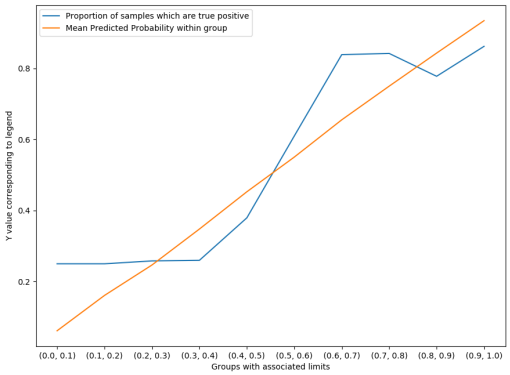

Figure 2: Mean predicted probability in group (orange) and percentage of positive labels in group (blue) for all groups of size 0.1 from 0 to 1

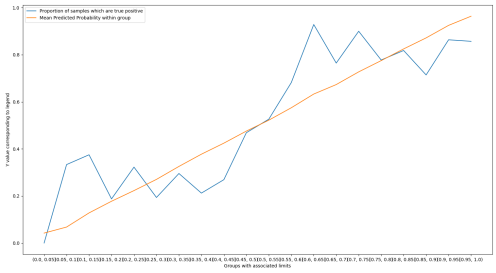

Figure 3: Mean predicted probability in group (orange) and percentage of positive labels in group (blue) for all groups of size 0.1 from 0 to 1

$p(x = H|\theta_1, \theta_2) = p(z = H|\theta_1)p(x = H|z = H, \theta_2) = \theta_1\theta_2$

Q10. Given a set of reported results D_r of size N_r , where the number of heads is n_h and the number of tails is n_t , what is the likelihood of D_r as a function of θ_1 and θ_2.

$p(x = H|\theta_1, \theta_2) = \theta_1\theta_2$
$p(x = T|\theta_1, \theta_2) = 1 - p(x = H|\theta_1, \theta_2) = 1 - \theta_1\theta_2$

Likelihood, $L(D_r; \theta_1, \theta_2) = \prod_{i=1}^{N_r} p(x = x_i|\theta_1, \theta_2) = (\theta_1\theta_2)^{n_h}(1 - \theta_1\theta_2)^{n_t}$

Q11. Can we estimate θ_1 and θ_2 using an MLE?

Log likelihood, $LL(D_r|\theta_1, \theta_2) = n_h \log(\theta_1\theta_2) + n_t \log(1 - \theta_1\theta_2)$

If we try to maximize the log likelihood by taking the derivative and setting it to zero:

$\frac{dLL(D_r|\theta_1,\theta_2)}{d\theta_1} = \frac{n_h}{\theta_1\theta_2}\theta_2 - \frac{n_t}{1-\theta_1\theta_2}\theta_2 = 0$
$\theta_1\theta_2 = \frac{n_h}{n_h+n_t}$
$\frac{dLL(D_r|\theta_1,\theta_2)}{d\theta_2} = \frac{n_h}{\theta_1\theta_2}\theta_1 - \frac{n_t}{1-\theta_1\theta_2}\theta_1 = 0$
$\theta_1\theta_2 = \frac{n_h}{n_h+n_t}$

So we are not able to disentangle the exact values of θ_1 and θ_2, we are only able to find the value of their product. So MLE in this manner with only D_r is not appropriate.