ML Homework 4

Liheng Cao

March 23, 2021

1

When w_0 increases, then the probability of predicting class 1 becomes higher, and vice versa.

$$h(x) = \frac{1}{1 + e^{-(w_0 + \cdots)}}$$

This is because as w_0 gets bigger, the result of exponentiation gets smaller. As the denominator gets smaller, the quotient gets larger. $h(x)$ is used as the probability we will predict a certain class. The higher $h(x)$ is, the more likely we will predict class 1 (as opposed to class 0).

2

The logistic function is

$$\frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}$$

If we were to double $\mathbf{w}$, the result wouldn't actually change, as long as the threshold is 0.5.

The geometric interpretation is because $\mathbf{w}^T\mathbf{x}$ represents a (hyper)plane. Multiplying the equation of a plane doesn't affect any points that are on the plane (threshold $= 0.5 \implies$ on hyperplane), but it makes the points that are off it seem further away, increasing the certainty with which we pick a class. For example, if $\mathbf{w}^T\mathbf{x} = 0$, then there is no change ($0 \cdot 2 = 0$). But if it's positive, it becomes a larger positive number, and vice versa.

3

(a)

$$\begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}$$

(b) :

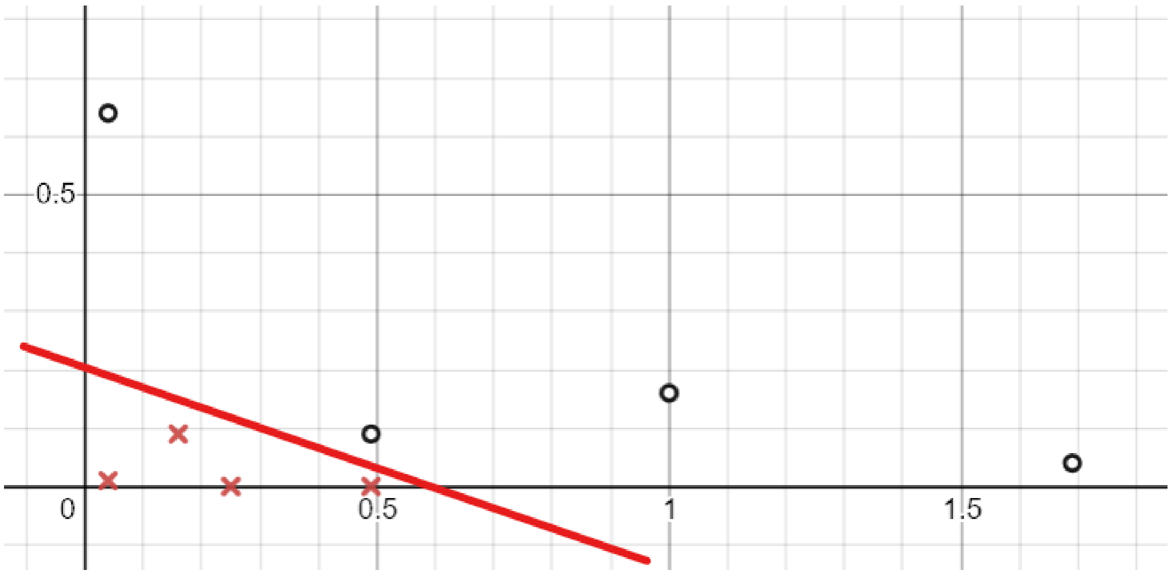

Figure 1: o: class 0, x: class 1

(c) $\text{TPR} = \dfrac{1}{3+1} = 1/4$

(d) $\text{FPR} = \dfrac{3}{3+1} = 3/4$

(e) $\text{accuracy} = \dfrac{3+3}{3+1+3+1} = 3/4$

(f) $\text{recall} = \dfrac{3}{3+1} = 3/4$

(g) $\text{precision} = \dfrac{3}{3+1} = 3/4$

(h)

$$-[0 \cdot \ln(0.389) + (1-0)\ln(1-0.389)] - \cdots - [1 \cdot \ln(0.638) + (1-1)\ln(1-0.638)] = 4.25205106\ldots$$

(i)

$$-\left[0 \cdot \ln\left(\frac{1}{1+\exp(-\mathbf{w}'^T \mathbf{X}^{(1)})}\right)\right] + \cdots - \left[0 \cdot \ln\left(\frac{1}{1+\exp(-\mathbf{w}'^T \mathbf{X}^{(8)})}\right)\right] = 3.0158793\ldots$$

Since $\mathbf{w}'$ has a lower cross-entropy, it is a better fit for this data.

(j) After one interation,

$$\mathbf{w} = \mathbf{w} + \frac{\alpha}{N}\mathbf{x}^T\left(y - \sigma(\mathbf{x}\mathbf{w})\right)$$

$$\begin{bmatrix} 0.66 \\ -2.24 \\ -0.18 \end{bmatrix}$$

$$+\frac{0.01}{N}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0.49 & 1.69 & 0.04 & 1 & 0.16 & 0.25 & 0.49 & 0.04 \\ 0.09 & 0.04 & 0.64 & 0.16 & 0.09 & 0 & 0 & 0.01 \end{bmatrix}\left(\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} - \sigma\left(\begin{bmatrix} 1 & 0.49 & 0.09 \\ 1 & 1.69 & 0.04 \\ 1 & 0.04 & 0.64 \\ 1 & 1 & 0.16 \\ 1 & 0.16 & 0.09 \\ 1 & 0.25 & 0 \\ 1 & 0.49 & 0 \\ 1 & 0.04 & 0.01 \end{bmatrix}\begin{bmatrix} 0.66 \\ -2.24 \\ -0.18 \end{bmatrix}\right)\right)$$

$$= \begin{bmatrix} 0.661\ldots \\ -2.239\ldots \\ -0.181\ldots \end{bmatrix}$$

(k) These data points would contribute a medium amount to the new $\mathbf{w}$.

(l) These data points would contribute a low amount to the new $\mathbf{w}$.

(m) These data points would contribute a high amount to the new $\mathbf{w}$.

(n)

$$-\left[0 \cdot \ln\left(\frac{1}{1+\exp\left(-\mathbf{w_g}^T\mathbf{X^{(1)}}\right)}\right)\right] + \cdots - \left[0 \cdot \ln\left(\frac{1}{1+\exp\left(-\mathbf{w_g}^T\mathbf{X^{(8)}}\right)}\right)\right] = 4.2519\ldots$$

This is a very minor decrease from 4.2520. This makes sense because gradient ascent is supposed to make the model fit better and have less error.

4

- lasso

$$-\lambda\left(|w_1| + \cdots + |w_d|\right) + \sum_{i=1}^{N} y^{(i)} \ln\left(h(x)\right) + \left(1 - y^{(i)}\right) \ln\left(1 - h(x)\right)$$

- ridge

$$-\lambda\left(w_1^2 + \cdots + w_d^2\right) + \sum_{i=1}^{N} y^{(i)} \ln\left(h(x)\right) + \left(1 - y^{(i)}\right) \ln\left(1 - h(x)\right)$$

- gradient

$$\mathbf{x}^T(y - \sigma(\mathbf{x}\mathbf{w})) - 2\lambda I'\mathbf{w}$$

- Optimal $\lambda = 0.6$. It did help, or else the best lambda would be 0. The training error would get worse (no more overfitting), but the test/validation data would get better.

```python
# TODO Q09
# Write the gradient ascent function
def Gradient_Ascent(X_train_1, y_2d_train, learning_rate, num_iters, L=0):
    # Number of training examples.
    N = X_train_1.shape[0]
    # Initialize w(<np.ndarray>). Zeros vector of shape X_train.shape[1],1
    w = np.zeros((X_train_1.shape[1], 1))
    # Initiating list to store values of likelihood(<list>) after few iterations.
    likelihood_values = []
    for i in range(num_iters):
        y_hat = hypothesis(X_train_1, w)
        error = y_2d_train - y_hat

        gradient = np.dot(X_train_1.T, error) - 2*L*(w-w[0,0])
        # Updating Parameters
        w = w + learning_rate / N * gradient
        if (i % 100) == 0:
            likelihood_values.append(likelihood(X_train_1,y_2d_train,w,N))

    return w, likelihood_values
```

Figure 2: Adding Ridge regression

```python
from sklearn.model_selection import KFold
def cross(L):
    nfold = 5
    kf = KFold(n_splits=nfold, shuffle=True)
    X_unscaled = cancer.data
    X_unscaled = np.hstack((np.ones((X_unscaled.shape[0], 1)), X_unscaled))
    y_orig = cancer.target
    likelihoods = []
    for train, test in kf.split(X_unscaled):
        X_tr = X_unscaled[train,:]
        y_tr = y_orig[train]
        X_ts = X_unscaled[test, :]
        y_ts = y_orig[test]

        scaler_KFold = preprocessing.StandardScaler()
        X_tr = scaler_KFold.fit_transform(X_tr)
        X_ts = scaler_KFold.transform(X_ts)

        y_2d_ts = y_ts.reshape((y_ts.shape[0], 1))
        y_2d_tr = y_tr.reshape((y_tr.shape[0], 1))

        w_kfold, w_likelihood = Gradient_Ascent(X_tr, y_2d_tr, 0.001, 100000, L)
        likelihoods.append(likelihood(X_ts, y_2d_ts, w_kfold, X_ts.shape[0]))

    return max(likelihoods)


lambdas = []
div = 10
for lambda_ in range(0, 100):
    lambdas.append(cross(lambda_/div))
    print(lambda_/div, lambdas[lambda_])

print(f"index: {lambdas.index(max(lambdas))}, value:{max(lambdas)}")
```

Figure 3: 5 fold

Basically the code iterates over a range of lambdas and does cross validation on each, and then we see which one has the best errors.

Interestingly, if the iterations for gradient ascent were too low, I would actually get $\lambda = 0$ as the best error. I got $\lambda = 0.6$ from increasing the iterations and letting the code run for like 30 minutes.

5

(a) Predictors: average pitch; response: gender

(b) Predictors: matrix of where the strokes were; response: letter or number written

6

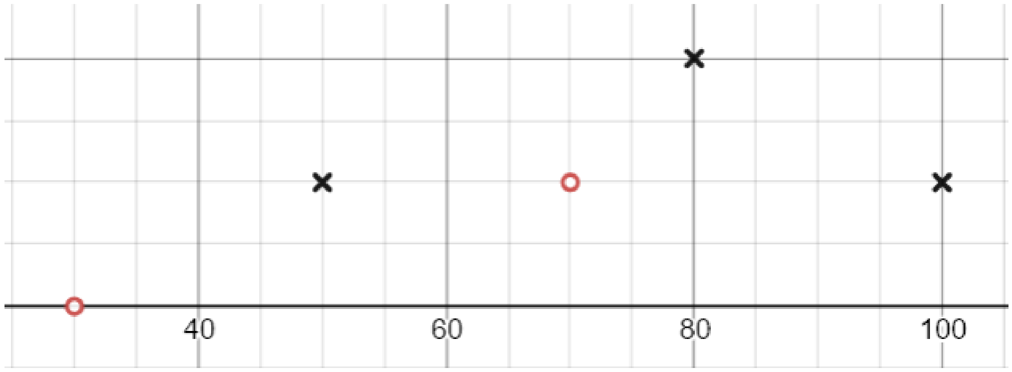

Figure 4: o: no donation, x: donation

(a)

(b) I will choose the horizontal line $x_2 = 0.5 \implies 0.5 + 0 \cdot x_1 + w_2$

$$\mathbf{w}^T = [-0.5, 0, 1]$$

(c) The least likely would be the one that is incorrect. In this case, it's same 3, or the one that earns 70k, follows 1 site, and did not donate. No calculator was needed for this!

(d) They would not change the values of $\hat{y}$, but they would change the likelihoods. If the prediction was correct, then the likelihood would increase. If the predictions were incorrect, then the likelihood would decrease.