Machine learning HW2 report

b02502108
陶昇永

1. logistic regression function:

```
#----------------------sigmoid-------------------------------
def sigmoid(z):
	return 1/(1+math.exp(-z))
#----------------------variables-----------------------------
b,dLdb = 0,0
w = []
for i in range(58):
	w.append(0.1)
dLdw = []
for i in range(58):
	dLdw.append(0)
learningrate = 1
Gt = []
Gtb = 0
for i in range(58):
	Gt.append(0)
count = 0
#--------------------logistic regression-----------------------------------------
while True:
	RMSE = 0
	count = count + 1
	for i in range (4000):
		yhat = float(data[i][58])
		X = np.array([float(j) for j in data[i][0:58]])
		W = np.array(w)
		ypredict = sigmoid(np.dot(X,W)+b)
		if ypredict >= 0.5:
			ypredict1 = 1
		else:
			ypredict1 = 0
		RMSE = RMSE + abs(yhat – ypredict1)	#計算錯估次數
		dLdb = dLdb + (yhat – ypredict)		#updating parameter
		Gtb = Gtb + dLdb**2				#adagrad parameter
		for j in range(1,58):
			dLdw[j] = dLdw[j] + (yhat - ypredict)*X[j]
			Gt[j] = Gt[j] + dLdw[j]**2
	print w,b,RMSE
	if (RMSE < 200):					#stop condition
		print w,b
		break
	b = b + learningrate*dLdb/(Gtb**0.5)		#updating parameter
	dLdb = 0
	for i in range(1,58):
		w[i] = w[i] + learningrate*dLdw[i]/(Gt[i]**0.5)
		dLdw[i] = 0
#------------------------------finish-----------------------------------------------
```

解釋作法：

寫法基本上和 gradient descent 一樣，只是要將 ypredict 算出來後丟入 sigmoid function，另外，我在

上面的程式碼中也利用在 train set 錯估的總次數，來監控 train 的進度，另外，這次的 loss function 微分值也和 gradient descent 不相同，在 weight 方面為 dLdw[i] = (yhat - ypredict)*X[i]，而在 bias 方面，則為 (yhat – ypredict) 。而上面的程式碼中，也包含了 adagrad 的實作。

2.method 2 :
我是使用 naive bayes classification 作為第 2 種方法，其中的機率模型選擇高斯分佈的模型，讓每個參數互相獨立，即，任兩組參數的 covariance = 0，P(x|c1) = P(x1|c1)*P(x2|c1)*....P(xk|c1)，並依照 P(c1|x) = P(x|c1)*P(c1)/(P(x|c1)*P(c1)+P(x|c2)*P(c2))來分類，若 P(c1|x)>0.5，就屬於第 1 類。
用這種分法得到的結果，比用 logistic 的方法要差一些，我覺得是因為將 covariance 省略基本上是一種比較簡化的模型，如果這樣去求，即暗示每個參數之間無任何關係，這顯然不太合理，因為在 mail 中，許多字是會同時出現的，不過，這樣會讓計算方便很多，而正確率其實也不差(0.83%左右)，所以也算是一個實用的方法！

3.這次的作業實作 logistic regression，因為經過了 sigmoid function 的關係，使 ypredict 總是介於 0~1 之間，造成 update 參數的幅度剛開始就比較低，也讓 regression 的速度快上許多。但是，過了一段時間後就會發現，Loss function 一直振盪，無法下降，應該代表的是 update 的幅度在後期反而變得太大了，而無法降下來，因此，adagrad 變成了必要的修正方法，讓 update 的幅度可以愈來愈小。不過我實作的時候，即使加了 adagrad，最終仍會在某個地方振盪，導致 loss 無法下降的情況。

4.討論的對象：朱晉德、吳雨葳、呂弈臻、黃馨平、施敬彥

5.我在 kaggle 投稿的最高分(0.93)是用 logistic regression 做出來的，而 method 2 則是約在 0.83 分左右。