This assignment represents my own work. I did not work on this assignment with others. All coding was done by myself.

I understand that if I struggle with this assignment that I will reevaluate whether this is the correct class for me to take. I understand that the homework only gets harder.

CS 671: Homework 2

Alex Kumar

Question 5

In [ ]:
```
### Imports
import math as m
import csv
from sklearn.model_selection import train_test_split
```

In [ ]:
```
### 5.2 Read files and preprocess
# Make dictionary with key: word, value: index
d = {}
with open("dict.txt") as f:
    for word in f:
        w, i = word.split()
        d[w] = i


def dictMaker(line, d):
    # Make dict for x_i: key: word index, value: indicator
    temp_d = {}
    for word in line.split(" "):
        if word in d:
            temp_d[d[word]] = 1
    return temp_d

X, y = [], []
# Read each line and process data
with open("moviereview.tsv") as f:
    tsv = csv.reader(f, delimiter="\t")
    for line in tsv:
        temp_d = dictMaker(line[1], d)
        X.append(temp_d)                    # dict
        y.append(int(line[0]))              # label
```

In [ ]:
```
### 5.3 Binary logistic regression
T, nu, theta = 30, 0.1, {}

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random

for k in d.keys():
    # dict with key: word idx, value: init 0
    theta[d[k]] = 0
```

```
def dotprod(theta, x_i):
    # dot product of theta and x_i
    total = 0
    for idx in x_i.keys():
        total += theta[idx]
    return total

for i in range(T):
    print("########## EPOCH: ", i, " ###########")
    for i in range(len(x_train)):
        x_i, y_i = x_train[i], y_train[i]   # [x: dict, y: label]

        dotproduct = dotprod(theta, x_i)
        p_i = -y_i + ( (m.e**(dotproduct)) / (1 + m.e**(dotproduct)) )

        for j in x_i.keys():
            theta[j] = theta[j] - nu * p_i
```

```
########## EPOCH:  0  ###########
########## EPOCH:  1  ###########
########## EPOCH:  2  ###########
########## EPOCH:  3  ###########
########## EPOCH:  4  ###########
########## EPOCH:  5  ###########
########## EPOCH:  6  ###########
########## EPOCH:  7  ###########
########## EPOCH:  8  ###########
########## EPOCH:  9  ###########
########## EPOCH:  10  ###########
########## EPOCH:  11  ###########
########## EPOCH:  12  ###########
########## EPOCH:  13  ###########
########## EPOCH:  14  ###########
########## EPOCH:  15  ###########
########## EPOCH:  16  ###########
########## EPOCH:  17  ###########
########## EPOCH:  18  ###########
########## EPOCH:  19  ###########
########## EPOCH:  20  ###########
########## EPOCH:  21  ###########
########## EPOCH:  22  ###########
########## EPOCH:  23  ###########
########## EPOCH:  24  ###########
########## EPOCH:  25  ###########
########## EPOCH:  26  ###########
########## EPOCH:  27  ###########
########## EPOCH:  28  ###########
########## EPOCH:  29  ###########
```

In [ ]:

```
### Predictions
def updateConfusion(confusion, pred, y):
    if y == 1:                  # P
        if pred >= 0.5:           # TP
            confusion[0] += 1
        else:                   # FP
            confusion[1] += 1
    else:                       # N
```

```
        if pred < 0.5:              # TN
            confusion[3] += 1
        else:                     # FN
            confusion[2] += 1
    return confusion


confusion = [0, 0, 0, 0]        # [TP, FP, FN, TN]
for i in range(len(x_test)):
    x, y = x_test[i], y_test[i]

    dot = dotprod(theta, x)
    pred = m.e**(dot) / (1 + m.e**(dot))

    confusion = updateConfusion(confusion, pred, y)
print("[TP, FP, FN, TN]: ", confusion)
```

```
[TP, FP, FN, TN]:  [102, 8, 23, 107]
```

In [ ]: