Problem Set 1 for Machine Learning 15 Fall

Jingyuan Liu
AndrewId: jingyual
jingyual@andrew.cmu.edu

1 Probability and Statistics Review

1.1 Exponential Families

(a) Show that Multinomial distribution, Multi-variate Gaussian distribution and Dirichlet distribution are members of the exponential families.

Multinomial distribution The probability mass function of multinomial distribution :

$$f(x_1, ..., x_k; p_1, ..., p_k) = \frac{\Gamma(\sum_i x_i + 1)}{\prod_i \Gamma(x_i + 1)} \cdot \prod_i^K p_i^{x_i} \tag{1}$$

Here the $x_i \in$ X, $p_i \in \theta$, and the Γ is the Gamma function:

$$\Gamma(n) = (n-1)! \quad , \qquad \Gamma(t) = \int_0^\infty x^{t-1} e^{-x} dx \tag{2}$$

To prove it belongs to exponential families, we can transfer it into:

$$f(X; \theta) = \frac{\Gamma(\sum_i x_i + 1)}{\prod_i \Gamma(x_i + 1)} \cdot \prod_i^K p_i^{x_i} \tag{3}$$

$$= h(x) \cdot p_1^{x_1} \cdot p_2^{x_2} ... p_i^{x_i} ... p_k^{x_k} \tag{4}$$

$$= h(x) \cdot exp(\sum_k x_i \cdot ln(p_i)) \tag{5}$$

Here $ln(x)$ is $log_e x$. Therefore, we can get

$$h(X) = \frac{\Gamma(\sum_i x_i + 1)}{\prod_i \Gamma(x_i + 1)} \tag{6}$$

$$\eta(\theta) = (ln(p_1), ln(p_2), ..., ln(p_k))^T \tag{7}$$

$$T(X) = (x_1, x_2, ..., x_k)^T \tag{8}$$

$$A(\theta) = 0 \tag{9}$$

Simply, we can derive for other specific distributions and prove that they also belong to exponential families:

Gaussian Distribution, $\{N(\mu, \sigma^2) : \mu \in \mathbb{R}, \sigma > 0\}$ is:

$$f(X, \theta) = \frac{1}{\sigma\sqrt{2\pi}} exp(-\frac{(x-\mu)^2}{2\sigma^2}) \tag{10}$$

$$= \frac{1}{\sqrt{2\pi}} exp\{\frac{\mu}{\sigma^2}x - \frac{1}{2\sigma^2}x^2 - \frac{\mu^2}{2\sigma^2} - ln\sigma\} \tag{11}$$

Therefore, we can prove that Gaussian distribution is in exponential families:

$$h(X) = \frac{1}{\sqrt{2\pi}} \tag{12}$$

$$\eta(\theta) = (\frac{\mu}{\sigma^2}, \frac{1}{2\sigma^2}) \tag{13}$$

$$T(X) = (x, -x^2) \tag{14}$$

$$A(\theta) = \frac{\mu}{2\sigma^2} + ln\sigma \tag{15}$$

Gamma Distribution is:

$$f(X; \theta) = \frac{1}{B(\alpha)} \prod_{i=1}^{K} x_i^{\alpha_i - 1} \tag{16}$$

$$B(\alpha) = \frac{\prod_{i=1}^{K} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{K} \alpha_i)} \tag{17}$$

After similar derivation with multinomial, we can get:

$$h(X) = 1 \tag{18}$$

$$\eta(\theta) = (\alpha_1 - 1, \alpha_2 - 1, ..., \alpha_i - 1, ..., \alpha_k - 1)^T \tag{19}$$

$$T(X) = (ln(x_1), ln(x_2), ..., ln(x_i), ..., ln(x_k))^T \tag{20}$$

$$A(\theta) = -ln(B(\alpha)) \tag{21}$$

(b) Computer the posterior

Based on Bayes Rule, we know that:

$$p(\theta \mid x) = \frac{p(x \mid \theta) \cdot p(\theta)}{p(x)} \tag{22}$$

The $p(\theta \mid x)$ is the posterior, the $p(x \mid \theta)$ is the likelihood, the $p(\theta)$ is the prior, and the $p(x)$ is the evidence. Given dataset $D = \{x_i\}_{i=1}^{N}$, the evidence is observed, so we can get:

$$p(\mathcal{X}, \mathcal{V} \mid X) \propto p(X \mid \theta) \cdot p(\theta \mid \mathcal{X}, \mathcal{V}) \tag{23}$$

And the observed data is supposed to be iid., therefore, the likelihood :

$$p(X \mid \theta) = \prod_{i=1}^{N} p(x_i) = \prod_{i=1}^{N} h(x_i) exp(\theta^T T(x_i) - A(\theta)) \tag{24}$$

Therefore, the posterior is in the form of :

$$p(\mathcal{X}, \mathcal{V} \mid X) \propto \prod_{i=1}^{N} h(x_i) exp(\theta^T T(x_i) - A(\theta)) \cdot f(\mathcal{X}, \mathcal{V}) exp(\theta^T \mathcal{X} - \mathcal{V} A(\theta)) \tag{25}$$

$$= \prod_{i=1}^{N} h(x_i) \cdot f(\mathcal{X}, \mathcal{V}) \cdot exp(\theta^T (\sum_{i=1}^{N} T(x_i) + \mathcal{X}) - (N + \mathcal{V}) A(\theta)) \tag{26}$$

Therefore, we can see that the posterior takes the same form as the prior.

1.2 Maximum Likelihood Estimation

a. Prove $A'(\hat{\theta}_{ML}) = \frac{1}{N} \sum_i T(X_i)$

Given tge datasets, we can get the log likelihood function:

$$log(p(X)) = \prod_{i=1}^{N} log(p(x_i)) = \sum_{i=1}^{N} log(h(x_i)) + \theta \sum_{i=1}^{N} x_i - N A(\theta) \tag{27}$$

We want to maximize the likelihood, which is the same as log likelihood function considering that log is a convex function, therefore we can get :

$$\frac{\partial log(p(x))}{\partial \theta} = \sum_{i=1}^{N} T(x_i) - N A'(\theta) \tag{28}$$

Then at the maximal point, the partial would be 0, so we can prove it:

$$A'(\hat{\theta}_{ML}) = \frac{1}{N} \sum_i T(X_i) \tag{29}$$

b. Compute the uniform distribution maximal likelihood

As we can see from the uniform distribution, we can get the log likelihood :

$$log(p(x \mid \theta)) = \sum_{i=1}^{N} log(\frac{1}{\theta}) \tag{30}$$

Therefore, to maximize the log likelihood, we can get the partial descent :

$$\frac{\partial log(p(x \mid \theta))}{\partial \theta} = -\sum_{x=1}^{N} \theta^{-1} = -\frac{N}{\theta} < 0 \tag{31}$$

We can see that the likelihood is decreasing with the increase of θ. So when the θ get the minimual value, the likelihood can get the max value. Therefore, θ_{ML} should be $\frac{1}{x_{max}}$.

2 Decision Boundary of Naive Bayes

2.1 Compute posterior and decision boundary

The posterior for $p(Y = 1 \mid X)$ is :

$$p(Y = 1 \mid X) = \frac{p(X \mid Y = 1) \cdot p(Y = 1)}{p(X)} \tag{32}$$

$$= \frac{\prod_{i=1}^{d} p(X_i \mid Y = 1) \cdot p(Y = 1)}{\sum_y \prod_{i=1}^{d} p(X_i \mid Y) \cdot p(Y)} \tag{33}$$

$$= \frac{\prod_i^d h(x_i) exp(\theta_{i1} T_i(x_i) - A_i(\theta_{i1}))\pi}{\prod_i^d h(x_i)(exp(\theta_{i1} T_i(x_i) - A_i(\theta_{i1}))\pi + exp(\theta_{i0} T(x_i) - A_i(\theta_{i0}))(1-\pi))} \tag{34}$$

$$= \prod_i^d \frac{1}{1 + (\frac{1-\pi}{\pi}) exp(T(x_i)(\theta_{i0} - \theta_{i1}) - (A_i(\theta_{i0}) - A_i(\theta_{i1})))} \tag{35}$$

$$= \prod_i^d \sigma((\theta_{i0-\theta_{i1}}) T(X_i) - (A_i(\theta_{i0}) - A_i(\theta_{i1})) - k) \tag{36}$$

Here, k is:

$$k = exp(ln(\frac{1-\pi}{\pi})) \tag{37}$$

And posterior for $p(Y = 1 \mid X)$ is similar to this. To compute the decision boundary, we do not need to care the denominer, considering that given a observed dataset, the $p(x)$ remains the same. So we can get:

$$P(X \mid Y = 1) \cdot p(Y = 1) = P(X \mid Y = 0) \cdot p(Y = 0) \tag{38}$$

$$log(\prod_i h(x_i) exp(\theta_{i1} T(x_i) - A_i(\theta_{i1}))\pi) = log(\prod_i h(x_i) exp(\theta_{i0} T(x_i) - A_i(\theta_{i0})(1-\pi))) \tag{39}$$

$$\sum_i \theta_{i1} T_{i1}(x_i) - A_i(\theta_{i1}) + log(\pi) = \sum_i \theta_{i0} T_{i0}(x_i) - A_i(\theta_{i0}) + log(1-\pi) \tag{40}$$

For those x which satisfy the equation 40, we would get the decision boundary.

2.2 Gaussian linear boundary

As the equation (40) shows, and that $P(X_i \mid Y)$ is a Gaussian Distribution, we could get:

$$\sum_i \frac{\mu_{i1}}{\sigma_{i1}^2} x - \frac{1}{2\sigma_{i1}^2} x^2 - k_{i1} = \sum_i \frac{\mu_{i0}}{\sigma_{i0}^2} x - \frac{1}{2\sigma_{i0}^2} x^2 - k_{i0} \tag{41}$$

Here k is constants changing with i, but is not related to x or parameters. Considering that σ is not related with label, we can get:

$$\sum_i \frac{\mu_{i1}}{\sigma_{i1}^2} x - k_{i1} = \sum_i \frac{\mu_{i0}}{\sigma_{i0}^2} x - k_{i0} \tag{42}$$

Therefore, we could get that the decision boundary is linear in terms of X.

3 KNN Classification

3.1 Question (a)

I think it is impossiable to build a decision tree as descripted that exactly behave the same as the 1-NN classifier.

As the following figure 3.1 shows, we could see 1-NN would split the region into several parts, and in different parts, the classification result is different.While for decision tree, it could only split the area horizontly or vertically. Therefore, the classifiers based on these two models migh not behave exactly the same.

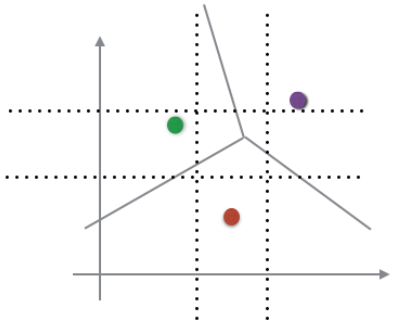

Figure 1: 1-NN Classifier vs "Binary" Decision Tree

3.2 Question (b)

We could set a special example to explain. Suppose we have only one dimension continuous variable and two points, with x_0 = 0, x_1 = 1. And we choose k = 2.

Then, we use the model to do classification. Suppose we have a new x, it could fall into three areas. We could notice that, when x falls between x_0 and x_1, the probabilty is $\frac{K}{NV} \cdot v'$, which equals 1. Then we could also find that the probability that x falls before x_0 and after x_1 is also positive, therefore, the overall probability is bigger than 1, which is unreasonable.

3.3 Question (c)

Considering that $p(e \mid x)$ is the limit that N is going to limit, so we could see here, x' is x in the limited case. Therefore, we could deriviate:

$$p_N(e \mid x) - p(e \mid x) = p(c_1 \mid x)p(c_2 \mid x') + p(c_2 \mid x)p(c_1 \mid x') - p(c_1 \mid x)p(c_2 \mid x) + p(c_2 \mid x)p(c_1 \mid x) \tag{43}$$

$$= p(c_1 \mid x)(p(c_2 \mid x') - p(c_2 \mid x)) + p(c_2 \mid x)(p(c_1 \mid x') - p(p_1 \mid x)) \tag{44}$$

If we use the distance as describe in the note, then we could get:

$$p_N(e \mid x) - p(e \mid x) = a(d_{c2}(x, x')) + (1 - a)d_{c1}(x, x') \tag{45}$$

Where $a = p(c_1 \mid x)$. $d_{c2}(x, x') = -d_{c1}(x, x')$, because $p(c_2 \mid x) + p(c_1 \mid x) = 1$. So :

$$E((p_N - p)^2 \mid x) = (d(x, x') - 2ad(x, x'))^2 \tag{46}$$

Therefore, in this case, we would have the minimal value for the upperbound.

4 Decision Trees

4.1 Decision tree on two features

Question (a)

Removing x' from the trainning dataset would change the learnt decision tree. I think there would be two main reasons:

(1) First, removing a "column" from dataset would change the conditional probability $p(X \mid Y)$, so it might influence the information gain for a certain feature, and influence us on choosing the best feature to split.

(2) Second, it would also influence us to decide the split result on a label. Removing x' from dataset may influence us that which is the majority case after splitting the feature.

Question (b) and Question (c) I think that in both the (b) and (c) case, the decision tree could correctly classify these vectors. The upper bound for both case is log_2^n, where n is the total number of the data points. My method to prove it is the same for both cases.

Let's use figures to show the meaning of it. If we have a total number of N points, and we want fit all data point. Given our classification tree model, we could see that each time we build one more layer, then we could split the total region into two times subregions. So in total, if we have K layers, then we could split the region into 2^k subregions.

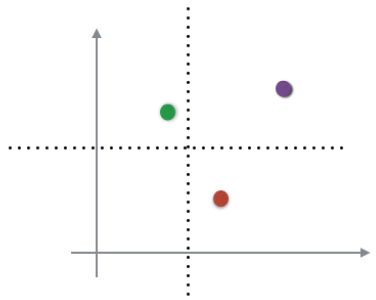

Figure 2: k = 2, 4 splitted subregions

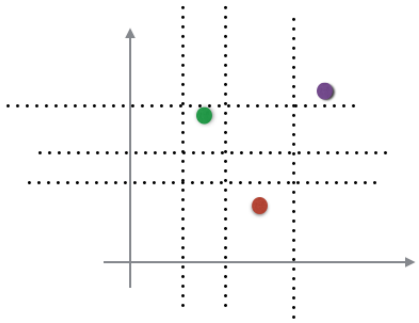

Figure 3: k = 4, 16 splitted subregions

Therefore, for N points and k layers, we could always build 2^k subregions. The upper bound is that N points falls into each subregions, so we need to assign each subregion a different label. Therefore:

$$2^k = N \qquad k = log_2(N) \tag{47}$$

4.2 C4.5 Algorithm

I implemented ID3 and C4.5 Decision tree via Weka Interface, which is a data mining software. The ID3 tree was in Weka but not suitable to do the classification and I exported the trainning output. The C4.5 was J48 in Weka library. The two trees are as follows:

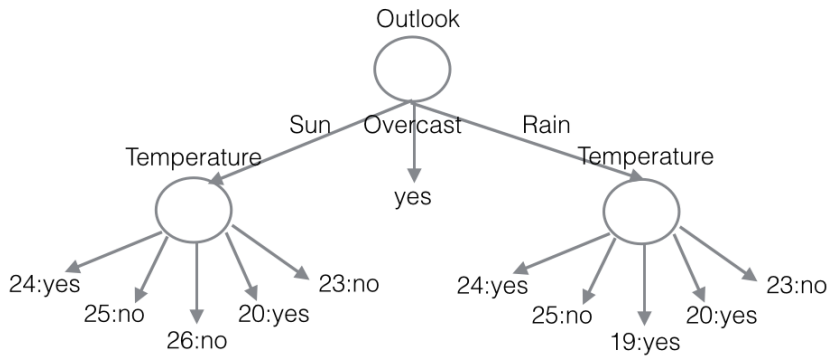

Figure 4: ID3 Decision Tree

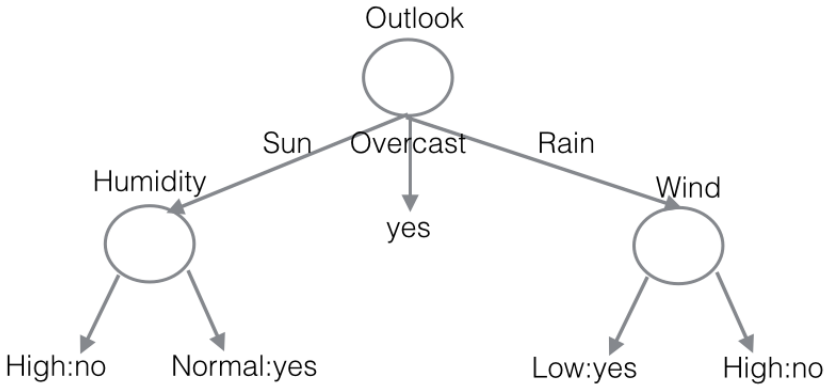

Figure 5: C4.5 Decision Tree

As we can see from these two figures, the ID3 prefer the features with larger V, like temperature, while C4.5 prefer the features with smaller V.

5 Naive Bayes

5.1 Result Analysis

I implemented Naive Bayes using C++. It costs more time than I planned... Feeling sad, I only finished the task one, preprocessment and classification. I did not do experiment with code. Next time, I would choose python or allocate more time for c++.

The result :

(1) For Train Datasets:

Neg Precision: 0.98375, Pos Precision: 0.9475, Overall Precision: 0.967125

(2) For Test Datasets:

Neg Precision: 0.34, Pos Precision: 0.28, Overall Precision: 0.31

As we can see, the trainning precision is fairly high while the testing precision is very low. The implemented model should be overfitted.

5.2 Running code

I design a Data class to preprocess data, a Model class to train the model, and the Evaluation clas to evaluate the result. To run the code, you would need to change the rootPaht in the data.cc to your local file path to load data from your disk. After this change, type make and run the ./main, it will run the overall program and then output the neg precision and pos precision seperately.

6 Collaboration

I discussed with Zheng Chen from MIIS Program with Question 3 and Question 4. Specifically, we discussed how to understand the layer relationship with the splitted subregions. Therefore we could solve Question 3.a, 4.1.b, and 4.1.c.

7 Code

(1) Makefile

```
OBJS = main.o data.o
CC = g++
CFLAGS = -O3 -std=c++0x -pg -Wall -c

main : main.o evaluation.o model.o data.o
	$(CC) main.o evaluation.o model.o data.o -o main

main.o : main.cc evaluation.o model.o data.o
	$(CC) $(CFLAGS) main.cc

evaluation.o : model.h evaluation.h evaluation.cc
	$(CC) $(CFLAGS) evaluation.cc

model.o : data.h model.h model.cc
	$(CC) $(CFLAGS) model.cc

data.o : data.h data.cc
	$(CC) $(CFLAGS) data.cc




clean:
	\rm *.o main
```

(2) Data class

```
#include <vector>
#include <string>
#include <map>
#include <set>

using namespace std;

/* Data class is used to preprocess file data*/
class Data {
    public:
        Data(string fileFolder, string sentWords);
```

```
        /* store words from files */
        string sentWordsPath;
        string fileFolderPath;
        vector<string> fileNames;
        vector<set<int>> fileWords;
        map<string, int> sentWord2IndexMap;
        map<int, string> sentIndex2WordMap;

        /* related functions */
        // load sentiment words
        void loadSent();
        // load file words, type: 0 keep only sentiwords; 1 all words
        void loadFile(int type);

    private:
        void tokenizeLine(string line, vector<string> &result);

};
```

```
#include <fstream>
#include <iostream>
#include <string>
#include <dirent.h>
#include <unistd.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <boost/algorithm/string/split.hpp>
#include <boost/algorithm/string/classification.hpp>

#include "data.h"

using namespace std;
using namespace boost::algorithm;

const static string fileRoot
= "~/Documents/cmu/15fall/10701/cmu701/assign/assign1/code/data/files/";
const static string sentRoot
= "~/Documents/cmu/15fall/10701/cmu701/assign/assign1/code/data/sent/";


/* Initilization, read file holder path and sentiment word path*/
Data::Data(string fileFolder, string sentWords) {
    Data::sentWordsPath = sentRoot + sentWords;
    Data::fileFolderPath = fileRoot + fileFolder;

    DIR *dir;
    struct dirent *ent;
    const char *fileFolderPathTemp = Data::fileFolderPath.c_str();

    if ((dir = opendir(fileFolderPathTemp)) != NULL) {
        while ((ent = readdir(dir)) != NULL) {
            string tempFileName(ent->d_name);
            if (tempFileName.length() > 2) {
                Data::fileNames.push_back(fileFolderPath + tempFileName);
            }
        }
        closedir(dir);
    } else {
        cout << "could not open folder" << endl;
    }

}

/* Read sentiment words and store in sent words related map*/
```

```
void Data::loadSent() {
    string readline;
    int index = 0;
    ifstream myfile (Data::sentWordsPath);
    if (myfile.is_open()) {
        while (getline(myfile, readline)) {
            if (!Data::sentWord2IndexMap.count(readline)) {
                Data::sentWord2IndexMap[readline] = index;
                Data::sentIndex2WordMap[index] = readline;
                index++;
            }

        }
    }
    myfile.close();
}

/* Read trainning files from folder */
/* Type 0, only keeps sentiwords, Type 1, keeps all words*/
void Data::loadFile(int type) {
    string readline;

    for (string filename : Data::fileNames) {
        ifstream myfile (filename);
        set<int> tempFileIndex;
        vector<string> tempWords;
        if (myfile.is_open()) {
            // store all the sentiwords for each doc
            if (type == 0) {
                while (getline(myfile, readline)) {
                    Data::tokenizeLine(readline, tempWords);
                    for (string word : tempWords) {
                        if (Data::sentWord2IndexMap.count(word)) {
                            tempFileIndex.insert(
                                    Data::sentWord2IndexMap[word]);
                        }
                    }
                }
            }
        } else {
            cout << "open file error! " << endl;
        }
        myfile.close();
        Data::fileWords.push_back(tempFileIndex);
    }
}


/* Parse doc, tokenize doc to vector string*/
void Data::tokenizeLine(string line, vector<string> &result) {
    split(result, line, is_any_of(" "));
}
```

(3) Model class

```
#include <vector>
#include <map>
#include <set>

#include "data.h"

using namespace std;

/* Model class is used to train and test data */
class Model {
```

```
    public:
        Model(Data *posData, Data *negData);

        /* Parameters*/
        int type;
        Data *posData, *negData;
        map<int, int> posIndex2CountMap;
        map<int, int> negIndex2CountMap;
        map<int, double> posIndex2ScoreMap;
        map<int, double> negIndex2ScoreMap;


        /* Train model */
        void trainModel();

    private:
        /* Transfer data to Index2Count map */
        /* type = 0 neg, type = 1 pos*/
        void countData(int type);

};
```

```
#include <vector>
#include <map>
#include <string>
#include <iostream>

#include "model.h"

using namespace std;

/* Constructor */
Model::Model(Data *posData, Data *negData) {
    Model::posData = posData;
    Model::negData = negData;
}

void Model::trainModel() {
    Model::countData(0);
    Model::countData(1);
    cout << "training model" << endl;
    int debug = 0;
    for (map<int,int>::iterator posIt = Model::posIndex2CountMap.begin();
            posIt != Model::posIndex2CountMap.end(); posIt++) {
        debug++;
        int posValue = posIt->second;
        double posScore = 0.0;

        posScore = ((double)posValue + 0.1) / ((double)800 + 0.1);
        Model::posIndex2ScoreMap[posIt->first] = posScore;
    }

    for (map<int,int>::iterator negIt = Model::negIndex2CountMap.begin();
            negIt != Model::negIndex2CountMap.end(); negIt++) {
        int negValue = negIt->second;
        double negScore = 0.0;

        negScore = ((double)negValue + 0.1) / ((double)800 + 0.1);
        Model::negIndex2ScoreMap[negIt->first] = negScore;
    }
}

void Model::countData(int type) {
    // neg label
    if (type == 0)  {
```

```
        for (set<int> negSet : Model::negData->fileWords) {
            for (int negTemp : negSet) {
                if (!Model::negIndex2CountMap.count(negTemp)){
                    Model::negIndex2CountMap[negTemp] = 0;
                }
                else{
                    Model::negIndex2CountMap[negTemp]++;
                }
            }
        }
    } else {
        for (set<int> posSet : Model::posData->fileWords) {
            for (int posTemp : posSet) {
                if (!Model::posIndex2CountMap.count(posTemp)) {
                    Model::posIndex2CountMap[posTemp] = 0;
                }
                else {
                    Model::posIndex2CountMap[posTemp]++;
                }
            }
        }
    }
}
```

(4) Evaluation class

```
#include <vector>
#include <string>
#include <map>

#include "model.h"

class Evaluation {
    public:
        /* Type = 0 for neg, Type = 1 for pos*/
        Evaluation(Model *model, Data *testData, int type);

        int type;
        Data *testData;
        Model *model;
        double precision;

        void evalModel();
};
```

```
#include <vector>
#include <set>
#include <map>
#include <string>
#include <math.h>
#include <iostream>

#include "evaluation.h"

using namespace std;

/* Type = 0 for neg, Type = 1 for pos*/
Evaluation::Evaluation(Model *model, Data *testData, int type) {
    Evaluation::type = type;
    Evaluation::testData = testData;
    Evaluation::model = model;
}

void Evaluation::evalModel() {
    int right = 0;
```

```
    int debug = 0;
    for (set<int> tempSet : Evaluation::testData->fileWords) {
        debug++;
        double posTotalScore = 0.0;
        double negTotalScore = 0.0;
        for (map<string, int>::iterator it =
                Evaluation::model->posData->sentWord2IndexMap.begin();
                it != Evaluation::model->posData->sentWord2IndexMap.end();
                it++) {

            int trainKey = it->second;
            string trainWord = it->first;
            if (Evaluation::testData->sentWord2IndexMap.count(trainWord))
            {
                int testkey =
                    Evaluation::testData->sentWord2IndexMap[trainWord];
                if (tempSet.find(testkey) != tempSet.end()) {
                    posTotalScore += log2(
                        Evaluation::model->posIndex2ScoreMap[trainKey]);
                } else {
                    posTotalScore += log2((double)1 -
                        Evaluation::model->posIndex2ScoreMap[trainKey]);
                }
            } else {
                posTotalScore += log2((double)1 -
                    Evaluation::model->posIndex2ScoreMap[trainKey]);
            }
        }
        for (map<string, int>::iterator it =
                Evaluation::model->negData->sentWord2IndexMap.begin();
                it != Evaluation::model->negData->sentWord2IndexMap.end();
                it++) {

            int trainKey = it->second;
            string trainWord = it->first;
            if (Evaluation::testData->sentWord2IndexMap.count(trainWord))
            {
                int testkey =
                    Evaluation::testData->sentWord2IndexMap[trainWord];
                if (tempSet.find(testkey) != tempSet.end()) {
                    negTotalScore += log2(
                        Evaluation::model->negIndex2ScoreMap[trainKey]);
                } else {
                    negTotalScore += log2((double)1 -
                        Evaluation::model->negIndex2ScoreMap[trainKey]);
                }
            } else {
                negTotalScore += log2((double)1 -
                    Evaluation::model->negIndex2ScoreMap[trainKey]);
            }
        }
        if (type == 0) {
            if (negTotalScore - posTotalScore >= 0) {
                right++;
            }
        } else {
            if (posTotalScore - negTotalScore >= 0) {
                right++;
            }
        }
    }
    Evaluation::precision = (double)right /
        Evaluation::testData->fileWords.size();

}
```

(5) Main

```
#include <iostream>
#include <string>
#include <vector>

#include "evaluation.h"

using namespace std;

int main() {
    string sentPath = string("sent.txt");

    /* Preprocess data*/
    string negTrainPath = string("neg_train/");
    Data negTrainData(negTrainPath, sentPath);
    negTrainData.loadSent();
    negTrainData.loadFile(0);

    string posTrainPath = string("pos_train/");
    Data posTrainData(posTrainPath, sentPath);
    posTrainData.loadSent();
    posTrainData.loadFile(0);

    Data negTestData("neg_test/", sentPath);
    negTestData.loadSent();
    negTestData.loadFile(0);

    Data posTestData("pos_test/", sentPath);
    posTestData.loadSent();
    posTestData.loadFile(0);

    /* Train the model */
    Model model(&posTrainData, &negTrainData);
    model.trainModel();

    /* Evaluate the model */
    Evaluation negEval(&model, &negTestData, 0);
    Evaluation posEval(&model, &posTestData, 1);
    negEval.evalModel();
    posEval.evalModel();
    cout << "neg result!" << negEval.precision << endl;
    cout << "pos result!" << posEval.precision << endl;
}
```