CS 559: Homework Set 2

Alana Laryssa S A Santos

10/8/2015

1 First problem

First we need to define the joint density function:

for $D = \{x_1, x_2, ..., x_n\}$

$$p(D \mid \theta) = \prod_{k=1}^{n} p(x_k \mid \theta)$$

$$p(D \mid \theta) = \prod_{k=1}^{n} \theta e^{-\theta x_k}$$

We need to find the θ that maximizes $p(D \mid \theta)$

To simplify the derivative calculus, we can take the natural logarithm of both terms:

$$\ln p(D \mid \theta) = \ln \prod_{k=1}^{n} \theta e^{-\theta x_k}$$

$$\ln p(D \mid \theta) = \sum_{k=1}^{n} \ln \theta e^{-\theta x_k}$$

In order to find the optimal θ, we have to set $\ln p(D \mid \theta)$ to zero

$$\sum_{k=1}^{n} \ln \theta e^{-\theta x_k} = 0$$

$$\sum_{k=1}^{n} \frac{e^{-\theta x_k} + \theta . e^{-\theta x_k} . (-x_i)}{\theta e^{-\theta x_k}} = 0$$

$$\sum_{k=1}^{n} \frac{1 - \theta x_k}{\theta} = 0$$

$$\frac{n}{\theta} - \sum_{k=1}^{n} x_k = 0$$

$$\theta = \frac{n}{\sum_{k=1}^{n} x_k}$$

2 Second problem

The results reported were obtained training both methods with the same samples. The algorithms run 25 times and the mean and standard deviation below are with respect to the accuracy of each method.

2.1 Maximum-Likelihood Estimation

Mean: 0.7326
Standard deviation: 0.0161

2.2 Naive Bayes classifier

Mean: 0.7422
Standard deviation: 0.0189

Evaluating these results, I see that in the data provided both classifiers work almost in the same way. But when training a small number of samples, MLE get better results.

3 Third problem

The results below are with respect to the accuracy of KNN-classifier.

3.1 k = 1

Mean: 0.6512
Standard deviation: 0.0213

3.2 k = 5

Mean: 0.7052
Standard deviation: 0.0183

3.3 k = 11

Mean: 0.7254
Standard deviation: 0.0195

4 Fourth problem

Mean: 0.6490
Standard deviation: 0.0261

5 MatLab codes

5.1 MLE and Naive Bayes

```
data = dlmread('pima-indians-diabetes.data');
gtidx = 9;
```

```
acc_nb = [];
acc = [];

for times = 1:25

    rp = randperm(length(data));
    data = data(rp,:);

    split = 20;
    train_data = data(1:split, :);
    test_data = data(split+1:end, :);

    active_feat = 2:4;
    % active_feat = 3;

    %% Training NB
    mean0_nb = mean(train_data(train_data(:,gtidx)==0, active_feat));
    mean1_nb = mean(train_data(train_data(:,gtidx)==1, active_feat));

    var0_nb = var(train_data(train_data(:,gtidx)==0, active_feat));
    var1_nb = var(train_data(train_data(:,gtidx)==1, active_feat));

    prior0temp_nb = length(train_data(train_data(:,gtidx)==0));
    prior1temp_nb = length(train_data(train_data(:,gtidx)==1));

    prior0_nb = prior0temp_nb/(prior0temp_nb+prior1temp_nb);
    prior1_nb = prior1temp_nb/(prior0temp_nb+prior1temp_nb);


    %% Training MLE
    mean0 = mean(train_data(train_data(:,gtidx)==0, active_feat));
    mean1 = mean(train_data(train_data(:,gtidx)==1, active_feat));

    cov0 = cov(data(data(:,gtidx)==0,active_feat));
    cov1 = cov(data(data(:,gtidx)==1,active_feat));

    prior0temp = length(train_data(train_data(:,gtidx)==0));
    prior1temp = length(train_data(train_data(:,gtidx)==1));

    prior0 = prior0temp/(prior0temp+prior1temp);
    prior1 = prior1temp/(prior0temp+prior1temp);

    % testing
    correct_nb = 0;
    wrong_nb = 0;
```

```
correct = 0;
wrong = 0;

for i = 1:length(test_data)

    %% Testing NB
    lklhood0_nb = 1; lklhood1_nb = 1;
    for j = active_feat
        lklhood0_nb = lklhood0_nb * ...
        exp(-(test_data(i,j)-mean0_nb(j-1)).^2/ ...
        (2*var0_nb(j-1))./sqrt(var0_nb(j-1));
        lklhood1_nb = lklhood1_nb * ...
        exp(-(test_data(i,j)-mean1_nb(j-1)).^2/ ...
        (2*var1_nb(j-1))./sqrt(var1_nb(j-1));
    end

    post0_nb = lklhood0_nb*prior0_nb;
    post1_nb = lklhood1_nb*prior1_nb;

    if and(post0_nb > post1_nb, test_data(i,gtidx) == 0)
       correct_nb = correct_nb + 1;
    elseif and(post0_nb < post1_nb, test_data(i,gtidx) == 1)
       correct_nb = correct_nb + 1;
    else
       wrong_nb = wrong_nb + 1;
    end

    %% Testing MLE
    x = test_data(i,active_feat);
    lh0 = exp(-1/2*(x-mean0)*inv(cov0)*(x-mean0)')/sqrt(det(cov0));
    lh1 = exp(-1/2*(x-mean1)*inv(cov0)*(x-mean1)')/sqrt(det(cov1));

    post0 = lh0*prior0;
    post1 = lh1*prior1;

    if and(post0 > post1, test_data(i,gtidx) == 0)
       correct = correct + 1;
    elseif and(post0 < post1, test_data(i,gtidx) == 1)
       correct = correct + 1;
    else
       wrong = wrong + 1;
    end


end
```

```
    acc = [acc correct/(correct+wrong)];
    acc_nb = [acc_nb correct_nb/(correct_nb+wrong_nb)];

end
disp('Naive_Bayes')
[mean(acc_nb) std(acc_nb)]
disp('MLE')
[mean(acc) std(acc)]
```

5.2 k-NN

```
data = dlmread('pima-indians-diabetes.data');
total = [];
for k = [1 5 11]
    acc = [];
    for it = 1:20

        rp = randperm(length(data));
        data = data(rp,:);

        train_data = data(1:length(data)/2, :);
        test_data = data(length(data)/2+1:end, :);

        active_feat = 2:4;

        % testing
        correct = 0;
        wrong = 0;

        for i = 1:length(test_data)
            sample = test_data(i,active_feat);

            idx = knnsearch(train_data(:,active_feat), sample, 'K', k, 'NSMethod

            if sum(train_data(idx, 9)) > k/2
                class = 1;
            else
                class = 0;
            end

            if test_data(i,9) == class
                correct = correct + 1;
            else
                wrong = wrong + 1;
```

```
            end
        end

        acc = [acc correct/(correct+wrong)];
    end
    total = [total; acc];
end

meann = mean(total')
stdd = std(total')
```

5.3 Parzen windows

```
clc, clear all;

data = dlmread('pima-indians-diabetes.data');
acc = [];

window = 20;

for times = 1:20

    rp = randperm(length(data));
    data = data(rp,:);

    split = length(data)/2;
    train_data = data(split, :);
    test_data = data(split+1:end, :);

    active_feat = 2:4;
    % active_feat = 3;

    % training
    mean0 = mean(train_data(train_data(:,9)==0, active_feat));
    mean1 = mean(train_data(train_data(:,9)==1, active_feat));

    var0 = var(train_data(train_data(:,9)==0, active_feat));
    var1 = var(train_data(train_data(:,9)==1, active_feat));

    prior0temp = length(train_data(train_data(:,9)==0));
    prior1temp = length(train_data(train_data(:,9)==1));

    prior0 = prior0temp/(prior0temp+prior1temp);
    prior1 = prior1temp/(prior0temp+prior1temp);
```

```
    % testing
    correct = 0;
    wrong = 0;

    for i = 1:length(test_data)
        sample = test_data(i,active_feat);

        idx = rangesearch(train_data(:,active_feat), sample, window, 'NSMethod',
        idx = idx{1}';

        if sum(train_data(idx, 9)) > length(idx)/2
            class = 1;
        else
            class = 0;
        end

        if test_data(i,9) == class
            correct = correct + 1;
        else
            wrong = wrong + 1;
        end
    end

    acc = [acc correct/(correct+wrong)];
end

acc
[mean(acc) std(acc)]
```