1 EM algorithm

We used two ways of initialization for EM algorithm, I fixed the value of sigma for the first way of initialization. The data instance of the two ways of initialization are just as below:

The First Way of Initialization miu

miu[0] : 4.73023835416; miu[1] : 25.8779122573; miu[2] : 25.265258018

The Initialization of the pi

pi[0] : 0.2933859716832062; pi[1] : 0.16552117890628634; pi[2] : 0.5410928494105075

The Initialization of the sigma

sigma[0] : 1.0 ; sigma[1] : 1.0 ; sigma[2] : 1.0

The Second Way of Initialization miu

miu[0] : 22.36976428933; miu[1] : 24.394333201750005; miu[2] : 5.438396501659

The Initialization of the pi

pi[0] : 0.40793159086502095; pi[1] : 0.3350396859943732; pi[2] : 0.25702872314060576

The Initialization of the sigma

sigma[0] : 114.96039271174348 ; sigma[1] : 146.95000092256834 ; sigma[2] : 168.3828473701766

The Log Likelihood Is: -18852.330445484153

I executed the EM algorithm for 6 times, and get 6 pair of results:

The outcome of the first initialize strategy: (fix sigma)

The Log Likelihood Is: -62594.92482705112

The Log Likelihood Is: -15101.00501087722

The Log Likelihood Is: -15101.00501087722

The Log Likelihood Is: -15101.00501087722

The Log Likelihood Is: -15101.00501087722

The Log Likelihood Is: -61703.88713633171

The outcome of the second initialize strategy: (not fix sigma)

The Log Likelihood Is: -18852.330445484153

The Log Likelihood Is: -18775.64760091111

The Log Likelihood Is: -18774.913336820104

The Log Likelihood Is: -18831.794605916944

The Log Likelihood Is: -18852.250314322966

The Log Likelihood Is: -21150.680883124016

Conclusions:

We can conclude that both the two ways are sensitive to the initial settings of parameters. For the first way (fixed sigma), we can see sometimes that the EM algorithm will not converge. If we get the proper initialization values for the parameters, the EM algorithm will converge to the local optimum value. For the second way (not fixed sigma), we can see sometimes that the EM algorithm will not converge too. The outcome of the EM algorithm also depends on the initialization value. So in order to get the better outcome, we need to initialize the parameters for several times. Based on the outcomes we get, we can see that the first way of initialization is better.

2. Support vector machines.

Training with default parameters:

Command for training: svm-train.exe training.new

The outcome of the training:

Default kernel parameter: (2) radial basis function

optimization finished, #iter = 99; nu = 0.801753; obj = -30.091940, rho = -0.076980

nSV = 71, nBSV = 22; Total nSV = 71

Command for testing: svm-predict.exe validation.new training.new.model validition.out

The outcome of the testing: **accuracy** = 77.1429% (27/35) (classification)

Training with different kernel parameters:

1. The kernel parameter is: 0 (linear kernel)

Command for traing: svm-train.exe -t 0 training.new

The outcome of the training:

optimization finished, #iter = 579; nu = 0.017662; obj = -0.627017, rho = 1.172955

nSV = 40, nBSV = 0; Total nSV = 40

Command for testing: svm-predict.exe validation.new training.new.model validition.out

The outcome of the testing: **accuracy** = 85.7143% (30/35) (classification)

2. The kernel parameter is: 1(polynomial)

Command for traing: svm-train.exe -t 1 training.new

The outcome of the training:

optimization finished, #iter = 162; nu = 0.022567; obj = -0.801149, rho = 0.404372

nSV = 57, nBSV = 0; Total nSV = 57

Command for testing: svm-predict.exe validation.new training.new.model validition.out

The outcome of the testing: accuracy = **accuracy** = 74.2857% (26/35) (classification)

3. The kernel parameter is: 3 (sigmoid: tanh)

Command for traing: svm-train.exe -t 3 training.new

The outcome of the training:

optimization finished, #iter = 37; nu = 0.957746; obj = -65.367107, rho = -0.492870

nSV = 68, nBSV = 68; Total nSV = 68

Command for testing: svm-predict.exe validation.new training.new.model validition.out

The outcome of the testing: **accuracy** = 75.7143% (16/35) (classification)

The perceptron algorithm

I used weka to classify the data, and the accuracy about perceptron is 77.6479 %.

Compare to the outcome of the LIBSVM used kernel, we can conclude that we use the radial basis function, polynomial kernel and linear kernel, we can conclude that the outcome of the LIBSVM is better; but we use the sigmoid kernel, we can conclude that the outcome of the perceptron is better.

This because when we use the SVM, we also find the max margin of the dataset, but if the dataset can be classified, we can find many different parameters for the perceptron, but we cannot make sure that which perceptron is the best one. So the outcome of the LIBSVM may be better than perceptron.

3. Boosting

The first row is DecisionStump
The second row is J48
The third row is Logistic regression
The first dataset1: breast-cancer-wisconsin
Number of iterations: 30

Base learner	Vanilla	Bagging	Boosting
Classifier1	7.5823 %	6.867 %	4.578 %
Classifier2	5.4363 %	4.0057 %	3.4335 %
Classifier3	3.4335 %	3.4335 %	3.4335 %

Number of iterations: 100

Base learner	Vanilla	Bagging	Boosting
Classifier1	7.5823 %	7.2961 %	4.2918 %
Classifier2	5.4363 %	3.5765 %	3.1474 %
Classifier3	3.4335 %	3.4335 %	3.4335 %

Number of iterations: 150

Base learner	Vanilla	Bagging	Boosting
Classifier1	7.5823 %	7.1531 %	4.1488 %
Classifier2	5.4363 %	3.2904 %	2.7182 %
Classifier3	3.4335 %	3.4335 %	3.4335 %

The second dataset2: Image Segmentation data

Number of iterations: 30

Base learner	Vanilla	Bagging	Boosting
Classifier1	71.4286 %	43.5498 %	71.4286 %
Classifier2	3.2468 %	2.3377 %	1.3853 %
Classifier3	4.1126 %	3.9345 %	3.8381 %

Number of iterations: 100

Base learner	Vanilla	Bagging	Boosting
Classifier1	71.4286 %	43.5931 %	71.4286 %
Classifier2	3.2468 %	2.3377 %	1.5152 %
Classifier3	4.1126 %	3.9345 %	3.6371 %

Number of iterations: 150

Base learner	Vanilla	Bagging	Boosting
Classifier1	71.4286 %	43.5065 %	71.4286 %
Classifier2	3.2468 %	2.3377 %	1.645 %
Classifier3	4.1126 %	3.9345 %	3.5357 %

The third dataset3: Pima Indians Diabetes Data Set

Number of iterations: 30

Base learner	Vanilla	Bagging	Boosting
Classifier1	28.125 %	26.6927 %	25.2604 %
Classifier2	26.1719 %	25 %	27.2135 %
Classifier3	22.7865 %	22.6563 %	22.7865 %

Number of iterations: 100

Base learner	Vanilla	Bagging	Boosting
Classifier1	28.125 %	27.3438 %	24.6094 %
Classifier2	26.1719 %	25 %	25.3906 %
Classifier3	22.7865 %	22.6563 %	22.7865 %

Number of iterations: 150

Base learner	Vanilla	Bagging	Boosting
Classifier1	28.125 %	27.3438 %	24.349 %
Classifier2	26.1719 %	24.6094 %	26.0417 %
Classifier3	22.7865 %	22.7865 %	22.7865 %

1. Which algorithms + data set combination is improved by Bagging?

Based on the outcome above table, we can conclude for Bagging, all the algorithms + data set combinations are improved.

2. Which algorithms + data set combination is improved by Boosting?

Based on the outcome above table, we can conclude for Boosting,

The first dataset1: breast-cancer-wisconsin

DecisionStump + 30; J48 + 30;

DecisionStump + 100; J48 + 100;

DecisionStump + 150; J48 + 150;

The second dataset2: Image Segmentation data

J48 + 30; Logistic regression + 30;

J48 + 100; Logistic regression + 100;

J48 + 150; Logistic regression + 150;

The third dataset3: Pima Indians Diabetes Data Set

DecisionStump + 30; J48 + 30;

DecisionStump + 100; J48 + 100;

DecisionStump + 150; J48 + 150;

These algorithms + data set are improved

3. Can you explain these results in terms of the bias and variance of the learning algorithms applied to these domains? Are some of the learning algorithms unbiased for some of the domains? Which ones?

Bias: measures the accuracy or quality of the algorithm; high bias means a poor match.

Variance: measures the precision or specificity of the match, a high variance means a weak match.

There also a tradeoff between bias and variance, bagging reduces variance by averaging, bagging has little effect on bias. And we also can average and reduce bias through boosting.

Yes, we can see some algorithms are unbiased for some of domains, for instance, the decision tree is unbiased for the discrete domains, and some linear learning algorithms are unbiased for some continuous domains. Because of this reason, we choose the data set from UCI, I just choose three datasets only with numeric attributes.

4 K-means clustering on images

The command line of the Kmeans

Step1: javac KMeans.java

Step2: java KMeans <imageInput.jpg> <k> <imageOutput.jpg> for instance:java KMeans Koala.jpg 2 Koala2.jpg

1. The outcome of the Kmeans for Koala.jpg:

The comprssion rate: K = 2 is 0.031252544

Kmeans Running Time = 446ms

The comprssion rate: K = 5 is 0.093756355

Kmeans Running Time = 770ms

The comprssion rate: K = 10 is 0.12501271

Kmeans Running Time = 1216ms

The comprssion rate: K = 15 is 0.12501907

Kmeans Running Time = 2345ms

The comprssion rate: K = 20 is 0.15627544

Kmeans Running Time = 2055ms

****

2. The outcome of the Kmeans for Penguins.jpg:

The comprssion rate: K = 2 is 0.031252544

Kmeans Running Time = 2619ms

The comprssion rate: K = 5 is 0.093756355

Kmeans Running Time = 750ms

The comprssion rate: K = 10 is 0.12501271

Kmeans Running Time = 1137ms

The comprssion rate: K = 15 is 0.12501907

Kmeans Running Time = 1594ms

The comprssion rate: K = 20 is 0.15627544

Kmeans Running Time = 2092ms

****

The average compression rate for K=2: 0.031252544

The average compression rate for K=5: 0.093756355

The average compression rate for K=10: 0.12501271

The average compression rate for K=15: 0.12501907

The average compression rate for K=20: 0.15627544

And we can conclude that there is a tradeoff between the quality of the image and the compression rate of the image, we can see the number of the clusters k is 15 or 10 is a good value of K.