DD2421 HT20 Machine Learning

Lab 3: Bayesian Learning and Boosting

Erfan Wu (erfan@kth.se)

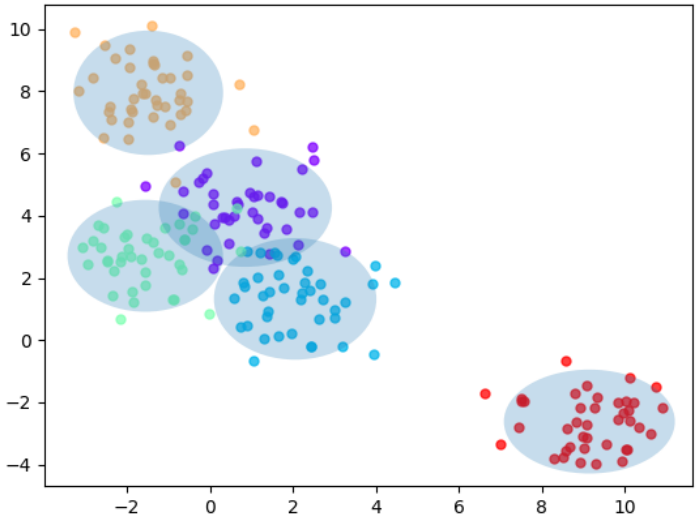
Yage Hao (yage@kth.se)

1. **Write a function, mlParams(X,labels), that computes the ML-estimates of μk and Σk for the different classes in the dataset.**

   Use the provided function, genBlobs(), that returns Gaussian distributed data points together with class labels, to generate some test data. Compute the ML-estimates for the data and plot the 95%-confidence interval using the function plotGaussians.

ANSWER

```
# TODO: fill in the code to compute mu and sigma!
# ===========================
for i,classname in enumerate(classes):
    idx = np.where(labels==classname)[0]
    X_classname = X[idx] #for a classname, get their corresponding xi
    _classname = W[idx] #corresponding wi
    mu[i,:] = np.sum(X_classname)/np.sum(X)
    sigma[i,:,:] = np.dot(np.transpose(X_classname-mu[i,:]),
                    (X_classname-mu[i,:]))/np.sum(X)
# ===========================
```



2. **Write a function computePrior(labels) that estimates and returns the class prior in X (ignore the W argument).**

   Write a function classifyBayes(X,prior,mu,sigma) that computes the dis- criminant function values for all classes and data points, and classifies each point to belong to the max discriminant value. The function should return a length N vector containing the predicted class value for each point.

ANSWER

Complete computePrior(labels) function.

```
    # TODO: compute the values of prior for each class!
    # ==========================
    for i, classname in enumerate(classes):
        idx = np.where(labels==classname)[0]
        W_classname = W[idx]
        prior[i] = np.sum(W_classname) / np.sum(W)
    # ==========================
```

Complete classifyBayes(X,prior,mu,sigma) function.

```
    # TODO: fill in the code to compute the log posterior logProb!
    # ==========================
    for i in range(Nclasses):
        for j in range(Npts):
            p1 = -0.5*np.log(np.linalg.det(sigma[i]))
            diffX = X[j] - mu[i]
            p2 = -0.5*np.dot(np.dot(diffX, np.linalg.inv(sigma[i])),
                 np.transpose(diffX))
            p3 = np.log(prior[i])
            logProb[i,j] = p1+p2+p3
    # ==========================
```

3. **When can a feature independence assumption be reasonable and when not?**

ANSWER

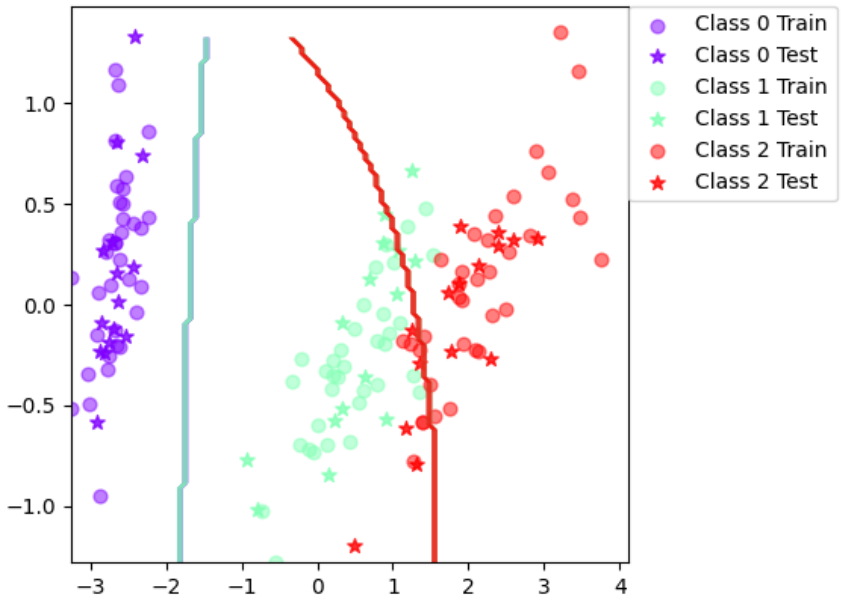
When the data is conditional independent, or mostly independent, such an assumption can be safely made. Otherwise, it is not good to make the assumption and it will be better to change another method.

How does the decision boundary look for the Iris dataset? How could one improve the classification results for this scenario by changing classifier or, alternatively, manipulating the data?

ANSWER

The decision boundary looks not good, especially between class1 and class2. It has a high bias and a low variance and does not distinguish the data well. One could improve the decision boundary by changing the classifier or use boosting (see our case in assignment 5). Otherwise, similar approaches in the SVM could be used, such as add dimensions to the data.

Iris:



```
Trial: 0 Accuracy 84.4
Trial: 10 Accuracy 95.6
Trial: 20 Accuracy 93.3
Trial: 30 Accuracy 86.7
Trial: 40 Accuracy 88.9
Trial: 50 Accuracy 91.1
Trial: 60 Accuracy 86.7
Trial: 70 Accuracy 91.1
Trial: 80 Accuracy 86.7
Trial: 90 Accuracy 91.1
Final mean classification accuracy  89 with standard deviation 4.16
```

Vowel:

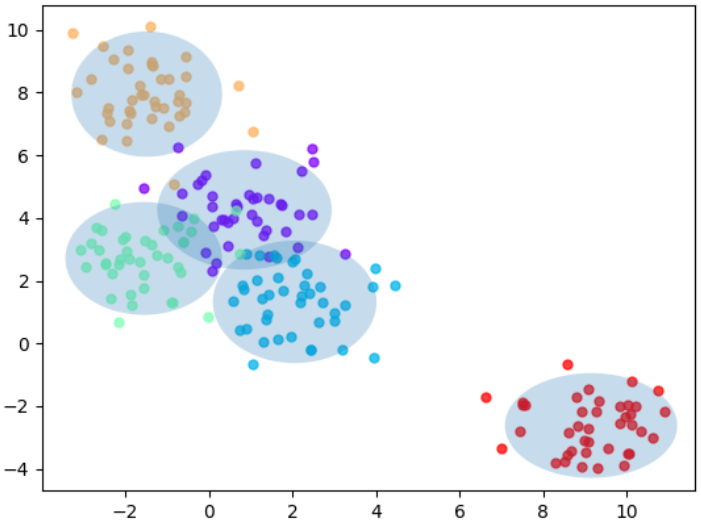
```
Trial: 0 Accuracy 61
Trial: 10 Accuracy 66.2
Trial: 20 Accuracy 74
Trial: 30 Accuracy 66.9
Trial: 40 Accuracy 59.7
Trial: 50 Accuracy 64.3
Trial: 60 Accuracy 66.9
Trial: 70 Accuracy 63.6
Trial: 80 Accuracy 62.3
Trial: 90 Accuracy 70.8
Final mean classification accuracy  64.7 with standard deviation 4.03
```

4. **Extend the old mlParams function to mlParams(X, labels, W) that handles weighted instances.**

ANSWER

```
# TODO: fill in the code to compute mu and sigma!
# =========================
for i,classname in enumerate(classes):
    idx = np.where(labels==classname)[0]
    X_classname = X[idx] #for a classname, get their corresponding xi
    W_classname = W[idx] #corresponding wi
    mu[i,:] = np.dot(np.transpose(W_classname),X_classname)/np.sum(W_classname)
    sigma[i,:,:] = cov_mat(X_classname, W_classname)
# =========================
```

We handle weighted instances by multiplying each feature xi with its corresponding updated weight wi.



5. **Is there any improvement in classification accuracy? Why/why not?**

ANSWER

Iris:

```
Trial: 0 Accuracy 95.6
Trial: 10 Accuracy 100
Trial: 20 Accuracy 93.3
Trial: 30 Accuracy 91.1
Trial: 40 Accuracy 97.8
Trial: 50 Accuracy 93.3
Trial: 60 Accuracy 93.3
Trial: 70 Accuracy 97.8
Trial: 80 Accuracy 95.6
Trial: 90 Accuracy 93.3
Final mean classification accuracy  94.7 with standard deviation 2.82
```

	Classification accuracy	Standard deviation
Previous basic classifier	89	4.16
Now with boosting	94.7	2.82

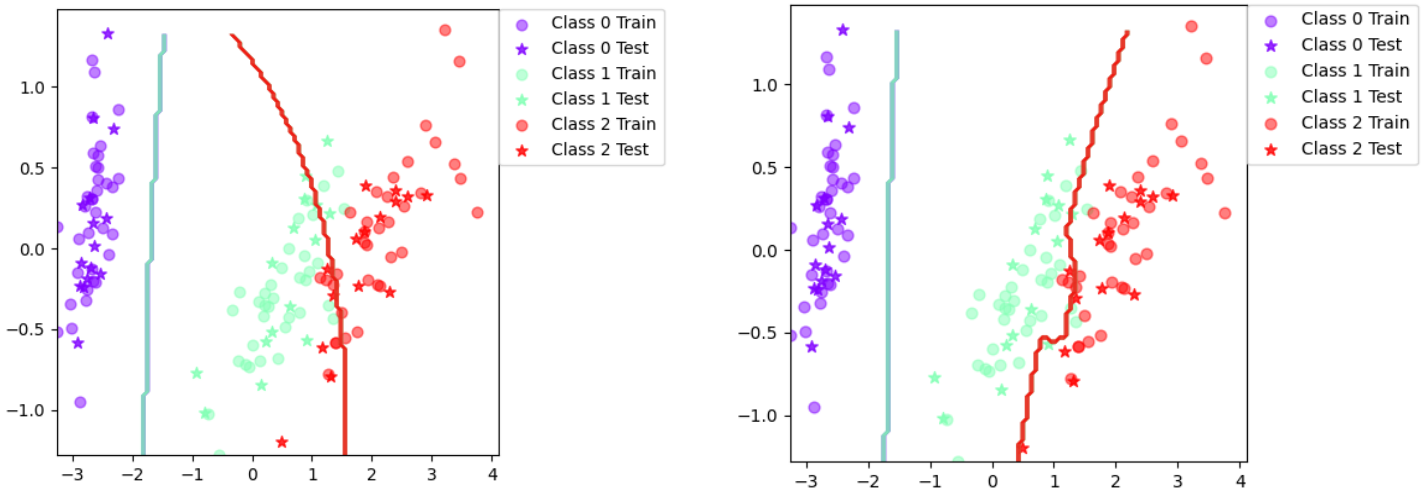
Vowel:

```
Trial: 0 Accuracy 76.6
Trial: 10 Accuracy 86.4
Trial: 20 Accuracy 83.1
Trial: 30 Accuracy 80.5
Trial: 40 Accuracy 72.7
Trial: 50 Accuracy 76
Trial: 60 Accuracy 81.8
Trial: 70 Accuracy 82.5
Trial: 80 Accuracy 79.9
Trial: 90 Accuracy 83.1
Final mean classification accuracy  80.2 with standard deviation 3.52
```

	Classification accuracy	Standard deviation
Previous basic classifier	64.7	4.03
Now with boosting	80.2	3.52

Plot the decision boundary of the boosted classifier on iris and compare it with that of the basic. What differences do you notice? Is the boundary of the boosted version more complex?

ANSWER



Left: non-boost, right: boost

The classification boundary of the boost classifier is more complex especially between class 1 and class 2. The Adaboost algorithm focuses more on misclassified data points so the boundary will adjust itself and reach a strong classification power.

Can we make up for not using a more advanced model in the basic classifier (e.g. independent features) by using boosting?

ANSWER

Yes. The basic classifier here is simply a naive bayesian classifier. By applying adaboost, several simple classifiers with weights combined to a complex final classifier which has good classification accuracy and small variance and it also avoid overfitting.

6. **Test the decision tree classifier on the vowels and iris data sets. Repeat but now by passing it as an argument to the BoostClassifier object. Answer questions 1-3 in assignment 5 for the decision tree.**

   Is there any improvement in classification accuracy? Why/why not?

ANSWER

Iris - DecisionTree:

```
Trial: 0 Accuracy 95.6
Trial: 10 Accuracy 100
Trial: 20 Accuracy 91.1
Trial: 30 Accuracy 91.1
Trial: 40 Accuracy 93.3
Trial: 50 Accuracy 91.1
Trial: 60 Accuracy 88.9
Trial: 70 Accuracy 88.9
Trial: 80 Accuracy 93.3
Trial: 90 Accuracy 88.9
Final mean classification accuracy  92.4 with standard deviation 3.71
```

Iris - DecTree - Boost:

```
Trial: 0 Accuracy 95.6
Trial: 10 Accuracy 100
Trial: 20 Accuracy 95.6
Trial: 30 Accuracy 93.3
Trial: 40 Accuracy 93.3
Trial: 50 Accuracy 95.6
Trial: 60 Accuracy 88.9
Trial: 70 Accuracy 93.3
Trial: 80 Accuracy 93.3
Trial: 90 Accuracy 93.3
Final mean classification accuracy  94.6 with standard deviation 3.65
```

	Classification accuracy	Standard deviation
Previous basic classifier	92.4	3.71
Now with boosting	94.6	3.65

Vowel - DecisionTree:

```
Trial: 0 Accuracy 63.6
Trial: 10 Accuracy 68.8
Trial: 20 Accuracy 63.6
Trial: 30 Accuracy 66.9
Trial: 40 Accuracy 59.7
Trial: 50 Accuracy 63
Trial: 60 Accuracy 59.7
Trial: 70 Accuracy 68.8
Trial: 80 Accuracy 59.7
Trial: 90 Accuracy 68.2
Final mean classification accuracy  64.1 with standard deviation 4
```

Vowel - DecTree - Boost:

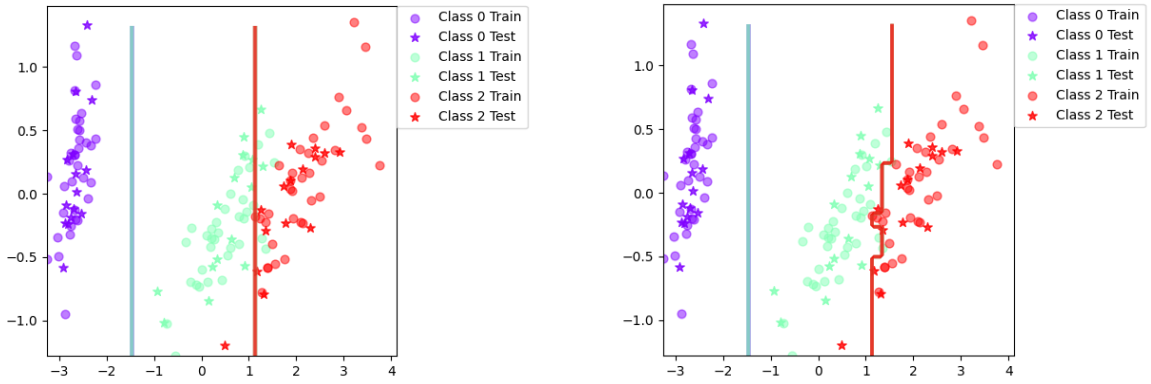
```
Trial: 0 Accuracy 86.4
Trial: 10 Accuracy 89.6
Trial: 20 Accuracy 86.4
Trial: 30 Accuracy 91.6
Trial: 40 Accuracy 80.5
Trial: 50 Accuracy 81.8
Trial: 60 Accuracy 86.4
Trial: 70 Accuracy 85.1
Trial: 80 Accuracy 86.4
Trial: 90 Accuracy 89.6
Final mean classification accuracy  86.5 with standard deviation 2.72
```

	Classification accuracy	Standard deviation
Previous basic classifier	64.1	4
Now with boosting	86.5	2.72

Plot the decision boundary of the boosted classifier on iris and compare it with that of the basic. What differences do you notice? Is the boundary of the boosted version more complex?

ANSWER

Iris - DecisionTree (L) - Boost (R):



The boundary of the basic decision tree classifier is linear while the boost classifier's is a combination of linear lines and is more complex.

Can we make up for not using a more advanced model in the basic classifier (e.g. independent features) by using boosting?

ANSWER

Yes we still obtained a more advanced classifier simply by using boosting based on adjusting on the basic decision tree classifier.

7. **If you had to pick a classifier, naive Bayes or a decision tree or the boosted versions of these, which one would you pick? Motivate from the following criteria:**

ANSWER

- **Outliers**

Boost. Since boosting will distribute a larger weight to the misclassification cases like outliers.

- **Irrelevant inputs: part of the feature space is irrelevant**

Decision tree with boosting. By applying pruning, the irrelevant features will not be considered by the model. And boosting will improve the accuracy and low the variance of results.

- **Predictive power**

Naive Bayes with boosting. Conclude from our computing above.

- **Mixed types of data: binary, categorical or continuous features, etc.**

Naive Bayes with boosting. It is easier for naive Bayes to handle the continuous features.

- **Scalability: the dimension of the data, D, is large or the number of instances, N, is large, or both.**

Decision tree with boosting. Naive Bayes assumes all features are independent to each other, thus may result in ‘curse of dimensionality’. Decision tree can handle large scales of data better with some tricks like pruning.