Lab 3

Bayesian Learning and Boosting

Arvid Holmäng
Klas Lindgren

Assignment 1

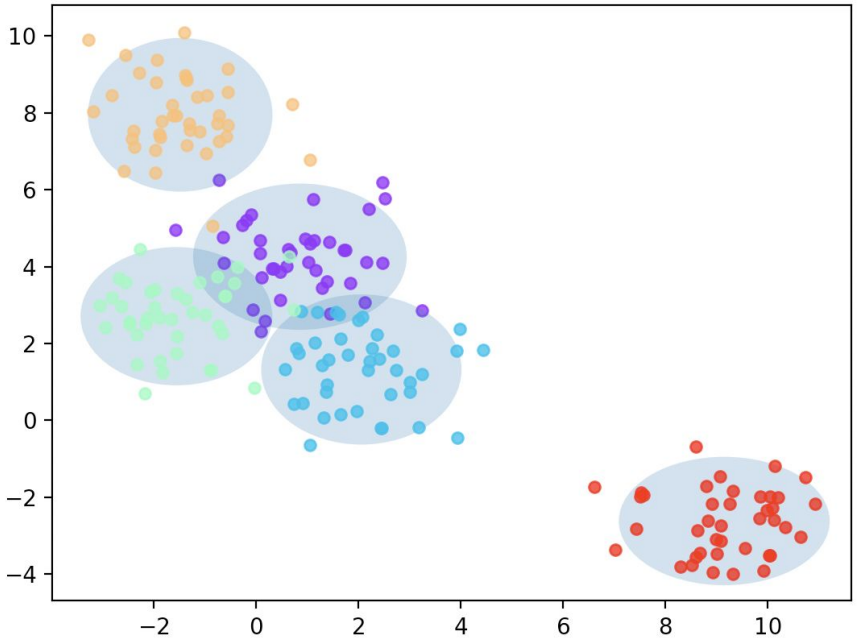
(1) **Write a function, mlParams(X,labels), that computes the ML- estimates of μk and Σk for the different classes in the dataset.**

(2) **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**

μ (Mu) (C × d-array) contains the class means (i.e. means of the associated class attributes)

Σ (Sigma) (C × d × d-array) contains the class covariances.

- The blue ellipses around classes is the covariance of the dataset *(how much attributes x,y change together).*



Equations for μ (mu) & Σ (Sigma)

$$\boldsymbol{\mu}_k = \frac{\sum_{\{i|c_i=k\}} \mathbf{x}_i}{N_k} \quad (8)$$

$$\boldsymbol{\Sigma}_k(m,m) = \frac{1}{N_k} \sum_{\{i|c_i=k\}} (\mathbf{x}_i(m) - \boldsymbol{\mu}_k(m))^2, \text{ and } \boldsymbol{\Sigma}_k(m,n) = 0 \text{ for } m \neq n. \quad (10)$$

Assignment 2

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

(2) **Write a function classifyBayes(X,prior,mu,sigma) that computes the discriminant 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.**

1) (12) →

```
# Uppgift 2: Write a function computePrior(labels) that estimates and returns the class prior in X (ignore the W argument).
# prior för en klass är # förekomster av en viss klass / totala förekomster av alla klasser
# equation 12 in assignment PDF
for c in classes:
    Nk=np.where(c == labels)[0] #labels related to a class
    n = len(labels) #labels in total
    print (len(Nk))
    prior[c] = np.size(Nk) / n

# ========================
```

2) (11) →

```
for c in range(Nclasses):
    det = np.linalg.det(sigma[c])
    determinant = -0.5 * np.log(det)
    for i in range(Npts):
        logProb[c][i] = determinant - 0.5 * np.matmul(np.matmul(X[i]-mu[c], np.linalg.inv(sigma[c])), (X[i]-mu[c]).transpose()) + np.log(prior[c])
# ========================
```

Equations that acts as basis for computePrior (12) and classifyBayes (11)

$$p(k) = \frac{N_k}{N}. \quad (12)$$

$$\begin{aligned}\delta_k(\mathbf{x}^*) &= \ln(p(k|\mathbf{x}^*)) = \ln(p_k(\mathbf{x}^*|k)) + \ln(p(k)) - \ln\sum_{l\in C} p_l(\mathbf{x}^*|l) \\ &= -\frac{1}{2}\ln(|\mathbf{\Sigma}_k|) - \frac{1}{2}(\mathbf{x}^* - \boldsymbol{\mu}_k)\mathbf{\Sigma}_k^{-1}(\mathbf{x}^* - \boldsymbol{\mu}_k)^{\mathrm{T}} - \frac{d}{2}\ln(2\pi) + \ln(p(k)) - \ln\sum_{l\in C} p_l(\mathbf{x}^*|l) \\ &= -\frac{1}{2}\ln(|\mathbf{\Sigma}_k|) - \frac{1}{2}(\mathbf{x}^* - \boldsymbol{\mu}_k)\mathbf{\Sigma}_k^{-1}(\mathbf{x}^* - \boldsymbol{\mu}_k)^{\mathrm{T}} + \ln(p(k)) + \mathcal{C}.\end{aligned} \quad (11)$$

Assignment 3

Actions:

(1) Test the accuracy for the vowels and iris datasets.

Questions:

(1) When can a feature independence assumption be reasonable and when not?

(2) 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?

1) The assumption of feature independence is unreasonable if it is known that the features substantially depend on one another.

   However, if there is little to no reliance, the assumption makes sense.

   For instance, if a squares side and area are two properties, we know that the area greatly depends on the diameter. The independence assumption is not reasonable in this situation.

2) **Changing classifier:** First boundary successfully divides class 0 and 1, second boundary fails to divide class 1 and 2. Changing classifier to e.g., a SVM (low complexity) could help with the division and possibly create better predictions.

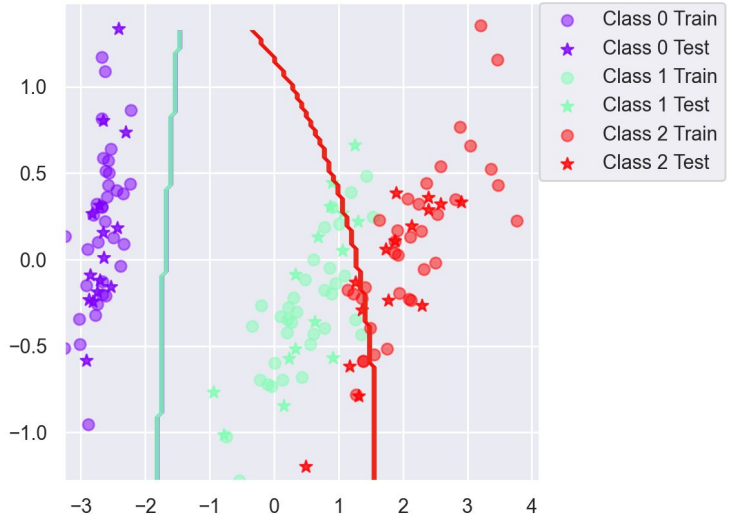
   Manipulating the data: We could remove data points that are far from the other points in the same class and thus reduce variance of the class.

```
testClassifier(BayesClassifier(), dataset='iris', split=0.7)

Testing Bayes Classifier on Iris dataset:
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
```

```
testClassifier(BayesClassifier(), dataset='vowel', split=0.7)

Testing Bayes Classifier on Vowel dataset:
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
```



Assignment 4

Actions:

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

Equations

$$\boldsymbol{\mu}_k = \frac{\sum_{\{i|c_i=k\}} \omega_i \mathbf{x}_i}{\sum_{\{i|c_i=k\}} \omega_i} \tag{13}$$

$$\boldsymbol{\Sigma}_k(m,m) = \frac{1}{\sum_{\{i|c_i=k\}} \omega_i} \sum_{\{i|c_i=k\}} \omega_i (\mathbf{x}_i(m) - \boldsymbol{\mu}_k(m))^2, \text{ and } \boldsymbol{\Sigma}_k(m,n) = 0 \text{ for } m \neq n. \tag{14}$$

```
for i in range(Nclasses):
    # Computing mu
    idx = np.where(labels == i)[0]
    mu[i] = sum(X[idx, :] * W[idx, :]) / sum(W[idx, :])

    # Computing sigma
    ximu = X[idx, :] - mu[i]
    mean = sum(np.square(ximu) * W[idx, :]) / sum(W[idx, :])
    sigma[i] = np.diag(mean)
```

Assignment 5

(1) **Is there any improvement in classification accuracy? Why/why not?**

(2) **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?**

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

1. There is **significant improvement** in classification accuracy!
   a. We have implemented Adaboosting →
      We have weights to every datapoint →
      Weight determines relative importance for datapoints in the classification
2. The boundaries have a **stronger fit with the datapoints**.
   a. The boundary between class 0 and 1 is straighter.
   b. The boundary 1 and 2 leans to the right.
   c. The boosted version makes the model more complex, since it takes the weights into account
3. **Yes!**
   a. Since the boosting will act as a more complex model, we can make up for not using a more advanced model in the basic classifier.
   b. **How?**
      With the boosting we can increase the accuracy. But this will lead to a more complex models which often increases variance and reduces bias.
   c. **Risks:**
      Boosting tend to overfit if the boosting is used to much!

Iris

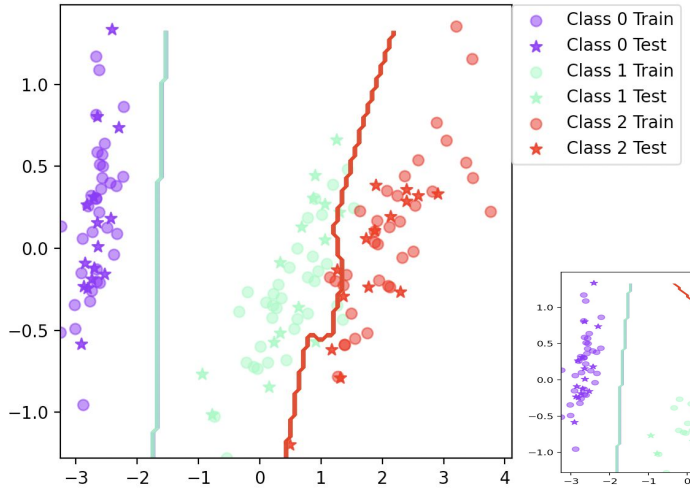
```
Testing BOOSTED Bayes Classifier on Iris dataset:
Trial: 0 Accuracy 95.6
Trial: 10 Accuracy 100
Trial: 20 Accuracy 93.3
Trial: 30 Accuracy 91.1
Trial: 40 Accuracy 97.8
Trial: 50 Accuracy 93.3
Trial: 60 Accuracy 93.3
Trial: 70 Accuracy 97.8
Trial: 80 Accuracy 95.6
Trial: 90 Accuracy 93.3
Final mean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

89 before

Vowel

```
Testing BOOSTED Bayes Classifier on Wowel dataset:
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
```

64.7 before



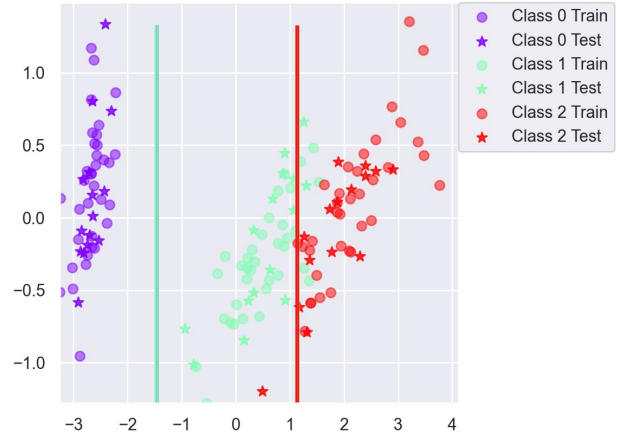
Assignment 6

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.

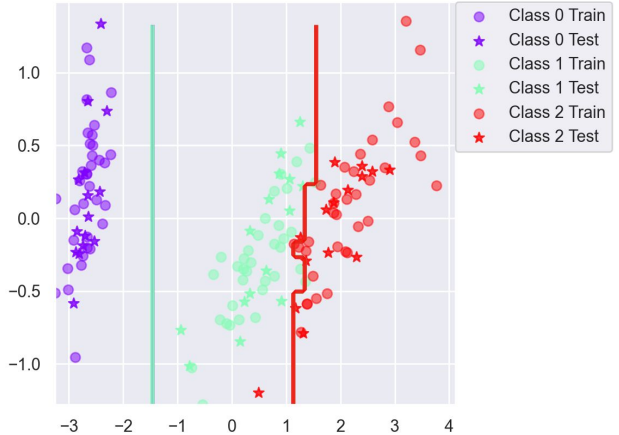
(1) Is there any improvement in classification accuracy? Why/why not?

(2) 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?

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




```
Testing Decision Tree Classifier on Iris dataset:
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
```




```
Testing BOOSTED Decision Tree Classifier on Iris dataset:
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
```

```
================================================================
=== DECTREE + VOWEL ============================================
================================================================
Testing Decision Tree Classifier on Vowel dataset:
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

Testing BOOSTED Decision Tree Classifier on Vowel dataset:
Trial: 0 Accuracy 85.7
Trial: 10 Accuracy 90.9
Trial: 20 Accuracy 87.7
Trial: 30 Accuracy 93.5
Trial: 40 Accuracy 83.1
Trial: 50 Accuracy 79.9
Trial: 60 Accuracy 90.9
Trial: 70 Accuracy 85.7
Trial: 80 Accuracy 83.1
Trial: 90 Accuracy 85.1
Final mean classification accuracy  86.5 with standard deviation 3.18
```

Assignment 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.

(1) Is there any improvement in classification accuracy? Why/why not?
(2) 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?
(3) Can we make up for not using a more advanced model in the basic classifier (e.g. independent features) by using boosting?

1. There is some improvement!
    a. There is barely any improvement in classification accuracy for the ***iris dataset.***
        i. The accuracy is already high!
    b. There is a significant improvement in classification accuracy for the ***vowel dataset.***
        i. A more complex dataset requires a more complex classifier. Boosting improves this!

2. The boundaries of the boosted classifier have a **marginally** stronger fit with the datapoints.
    a. The boundary between class 0 and 1 **looks identical**.
    b. The boundary 1 and 2 looks **more complex**.

3. Yes, especially in the more complex vowel-dataset!

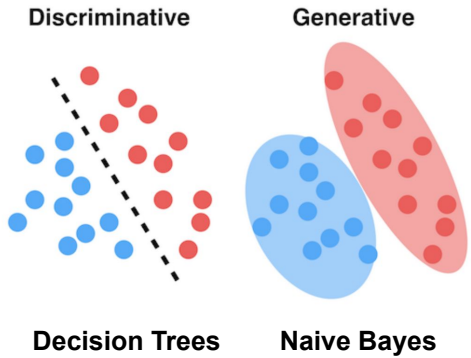
Assignment 7

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:

(1) Outliers: How the classifier is affected by outliers/noise.
(2) Irrelevant inputs: How the classifier is affected by features that are irrelevant
(3) Predictive power: How the classifier is affected by predictive power of certain classifications
(4) Mixed types of data: How the classifier is affected by the number of classes, etc.
(5) Scalability: How the classifier is affected by high dimensions, large datasets, etc.

General information…

- **Decision Trees** are discriminative models, whereas **Naive Bayes** is a generative model.
- With large datasets **Decision Trees** will get overfitted. Boosting can remove this (noise gets pruned). However, **Decision Tree** pruning may neglect some key values in training data, which can decrease the accuracy.
- **Naive Bayes** works well with small datasets, although more data will result in more accurate calculations
- **Decision Trees** have easy to use features to identify the most significant dimensions, handle missing values, and deal with outliers.
- **Naive Bayes** method is not affected by the curse of dimensionality and large feature sets.
- **Decision Trees** are more flexible and easy.
- **Naive Bayes** expects all features to be independent.
- A major advantage to **Naive Bayes** classifiers is that they are not prone to overfitting, thanks to the fact that they “ignore” irrelevant features. They are, however, prone to poisoning, a phenomenon that occurs when we are trying to predict a class C but features uncommon to C appear, causing a misclassification.
- A disadvantage with **Decision Trees** is that the algorithm separates the samples linearly. Imagine a continuous variable x, a possible split would be x=k. Depending on their nature, the data might not be linearly separable, and thus decision trees are unsuitable for these applications.
- **Naive Bayes** is better for multi-class classification, **Decision Trees** might be better for binary classification.



Assignment 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:

(1) Outliers: How the classifier is affected by outliers/noise.
(2) Irrelevant inputs: How the classifier is affected by features that are irrelevant
(3) Predictive power: How the classifier is affected by predictive power of certain classifications
(4) Mixed types of data: How the classifier is affected by the number of classes, etc.
(5) Scalability: How the classifier is affected by high dimensions, large datasets, etc.

To summarize…

(1) Decisions trees are preferred.
(2) Both Decision Trees and Naive Bayes are good at dealing with irrelevant inputs.
(3) Boosting algorithms preferred - consider other points for which of the boosted algorithms to choose.
(4) Our assumption is that Naive bayes in general will be more appropriate for mixed types of data (high number of classes) - picks class with highest probability.
(5) Because of the independence assumption, Naive Bayes classifiers are highly scalable and can quickly learn to use high dimensional features with limited training data.

Naive Bayes is a good choice for small datasets. It is not prone to overfitting. Not very sensitive to irrelevant input but is **sensitive to (1) outliers** e.g., uncommon features for a certain class.

Decision Trees can be selected if data is simple and linearly separable (appropriate for data in low dimensions). Decisions trees is the best choice with respect to **(2) irrelevant inputs -** as the inputs with most information gain is placed at the top of the tree, in combination with pruning, irrelevant outputs are removed. Decision Trees are also **not sensitive to outliers (1)** since the partitioning happens based on the proportion of samples within the split ranges and not on absolute values. The decision trees divide items by lines, so it does not difference how far is a point from lines. *(Example: "Guess the number").*

Boosted (Decision Trees or Naive Bayes) for **(3) predictive power**. Boosted models tend to have higher accuracy. However, depending on the dataset, either Boosted Naive Bayes or Boosted Decision Trees might be preferred.