1 Contradiction between Upper and Lower Bound

Here is my jugdment of the truth value of the four statements:

1. False.
2. True.
3. False.
4. True.

2 Validation

Let $\mathbb{B} = \{-1, 1\}$ be the binary output space. Let $\mathcal{X}$ be some input space. Let $\mathcal{H}$ be any hypothesis set such that $|\mathcal{H}| = 100$, and $h : \mathcal{X} \to \mathbb{B}$ for all $h \in \mathcal{H}$. Let ℓ be the zero-one loss. Let S be some set of i.i.d labeled samples from $\mathcal{X} \times \mathbb{B}$, drawn according to some probability distribution p on $\mathcal{X} \times \mathbb{B}$.

As I understand the question, we now have to determine the minimal size of S such that with probability at least 99% for all $h \in \mathcal{H}$, we have that

$$L(h) \leq \hat{L}(h, S) + 0.01 \tag{1}$$

which is equivalent to

$$L(h) - \hat{L}(h, S) \leq 0.01 \tag{2}$$

Another interpretation of the assignment text would be that we had to determine the minimal size of S such that with probability at least 99% for all $h \in \mathcal{H}$:

$$|L(h) - \hat{L}(h, S)| \leq 0.01 \tag{3}$$

However, we are normally not that interested in lower bounding the expected loss, since we do not normally worry about the expected loss being much lower than what we estimate from the empirical loss. Therefore, I think that line (2), where we only upper bound the expected loss, is a more natural interpretation of the assignment text than line (3), where we both upper and lower bound the expected loss. Therefore, I will now try to determine the minimal size of S such that with probability at least 99% for all $h \in \mathcal{H}$:

$$L(h) - \hat{L}(h, S) \leq 0.01 \tag{4}$$

From formulation (3.4) of **theorem 3.2** in Yevgeny's lecture notes, we know that since ℓ is bounded in $[0, 1]$ and $|\mathcal{H}| = 100$, then with probability at least $1 - \delta$ for all $h \in \mathcal{H}$:

$$L(h) - \hat{L}(h, S) \leq \sqrt{\frac{\ln \frac{100}{\delta}}{2n}} \tag{5}$$

where n is the size of S. If set $\delta = 0.01$, we get that with probability at least 99% for all $h \in \mathcal{H}$:

$$L(h) - \hat{L}(h, S) \leq \sqrt{\frac{\ln \frac{100}{0.01}}{2n}} \tag{6}$$

By simple calculations, we get that for all $n \in \mathbb{N}$:

$$\sqrt{\frac{\ln \frac{100}{0.01}}{2n}} \leq 0.01 \tag{7}$$

if and only if

$$\frac{\ln \frac{100}{0.01}}{2n} \leq (0.01)^2 \tag{8}$$

if and only if

$$\frac{\ln \frac{100}{0.01}}{2(0.01)^2} \leq n \tag{9}$$

Since

$$46051 < \frac{\ln \frac{100}{0.01}}{2(0.01)^2} < 46052 \tag{10}$$

we therefore have that if $46052 \leq n$, then

$$\sqrt{\frac{\ln \frac{100}{0.01}}{2n}} \leq 0.01 \tag{11}$$

By this and line (6), we therefore have that if $46052 \leq n$, then with probability at least 99% for all $h \in \mathcal{H}$:

$$L(h) - \hat{L}(h, S) \leq 0.01 \tag{12}$$

What about when $n \leq 46051$? In this case, we know from line (7-9) and (10) that

$$0.01 < \sqrt{\frac{\ln \frac{100}{0.01}}{2n}} \tag{13}$$

This means that if $n \leq 46051$, then we cannot anymore use **theorem 3.2** from Yevgeny's lecture to conclude that with probability at least 99% for all $h \in \mathcal{H}$:

$$L(h) - \hat{L}(h, S) \leq 0.01 \tag{14}$$

The fact that we cannot use a specific theorem to proof a statement does not mean, however, that that the statement is not true. Therefore, I do not think that it makes sense to say that 46052 is the minimal size of S such that line (14) is true. 46052 is the minimal size of S such that - with the assumptions, I listed in the beginning of this section, and the tools we have been taught in this course - we are able to proof that line (14) is true[1].

3 Kick it!

3.1 Question 1

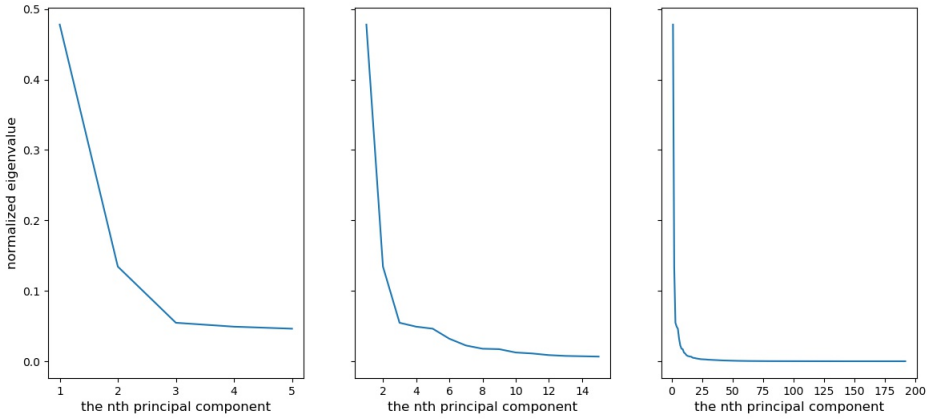
I use the Python library *pandas* to read in the data and compute the frequency of each of classes in the training dataset. Here is a table of the frequencies:

[1] Or at least, the minimal size such that I was able to proof that line (14) is true.

class	class index	frequency
player team 1	0	0.257
player team 2	1	0.488
goalkeeper team 1	2	0.027
goalkeeper team 2	3	0.059
referee	4	0.084
group	5	0.070
error	6	0.016

3.2 Question 2

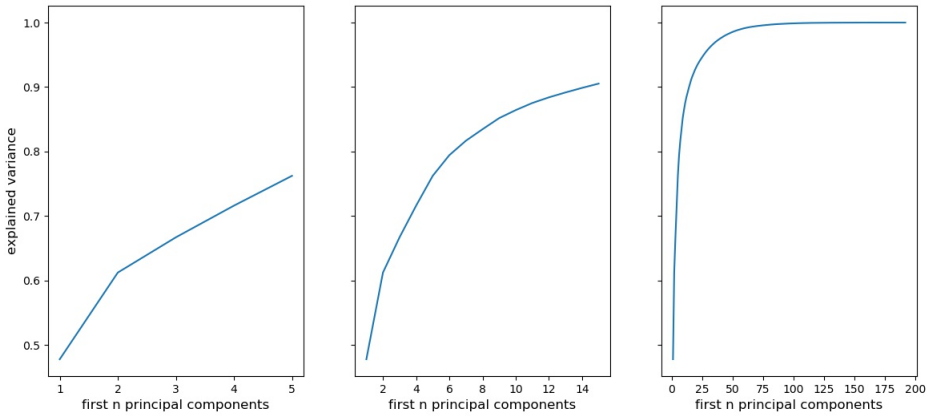
I use the *PCA* object from the module *decomposition* in the Python library *sklearn* to perform the principal component analysis. I use the module *pyplot* from the Python library *matplotlib* to make the plots. Here are some plots of the eigenspectrum of the first 5, 15 and all the 192 principal components:



As can be seen, I have decided to use the normalized eigenvalue of each principal component - that is the eigenvalue of the principal component divided by the sum of the eigenvalues of all the principal components - as the y-values in my plots.

Since principal component analysis sorts the components in descending order according to their eigenvalue, we see that the normalized eigenvalues of the components gets smaller and smaller. Thus, the first principal component almost makes up half of the sum of all the eigenvalues. The cumulative sum of the normalized eigenvalues of the first n principal components is also known as the fraction of explained variance by these components[2]. Here we see a plot of this quantity for our dataset:

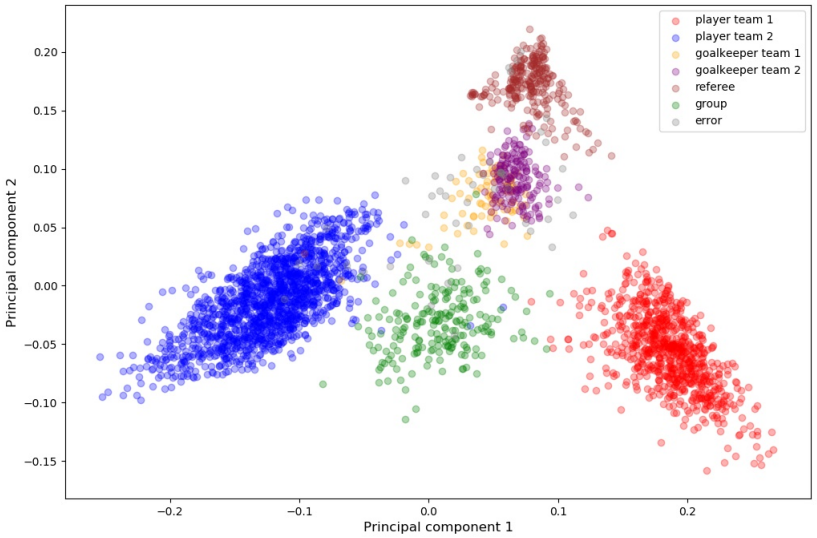
[2]Christian Igel: A Short Course in Data Mining, page 32.



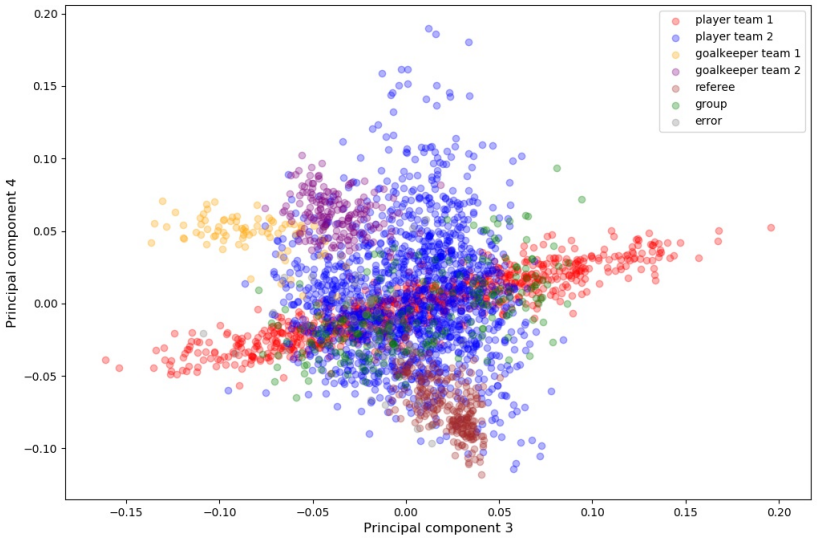
We see that the first 2 components already "explains more than 60%" of the variance in the dataset. From the plots and the following table, we that the first 15 components are just enough to "explain more than 90%" of the variance:

First n princ. comp.	Explained variance
1	0.478010
2	0.612426
3	0.667064
4	0.716177
5	0.762410
6	0.794413
7	0.816811
8	0.834599
9	0.851799
10	0.864249
11	0.875287
12	0.884031
13	0.891612
14	0.898710
15	0.905348

Here is a scatterplot of the training dataset projected onto the first two principal components with different colors for different classes:



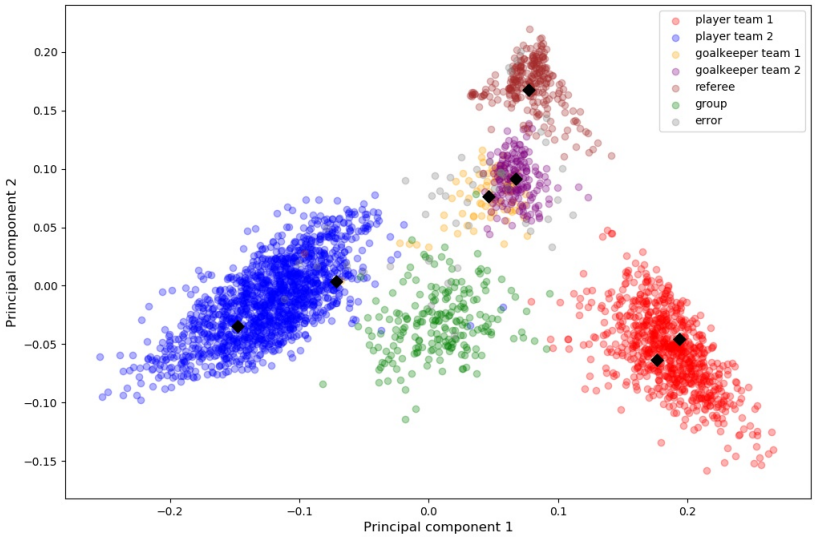
We see that although we represent the originally 192 dimensional data by only 2 dimensions, we are still able to tell the players from different teams, the referees and the groups apart. However, quite a few of the goalkeepers from the two teams are placed on top of each other, which mean that the two different kinds of goalkeepers do not seem to be clearly distinguishable by only the first two of the principal components of the data. However, if we plot the data projected onto the third and fourth principal components, we see that the goal keepers from the different teams might be distinguishable with respect to these two principal components:



Therefore, we might hope that we are able to distinguish all the classes except for the errors, if we project the originally 192 dimensional data onto a 4 dimensional space consisting of the first four principal components. All in all, this seems to be a pretty impressive example on how principal component analysis can be effectively used to reduce the dimensionality of data with a large number of features without loosing too much of the relevant structure.

3.3 Question 3

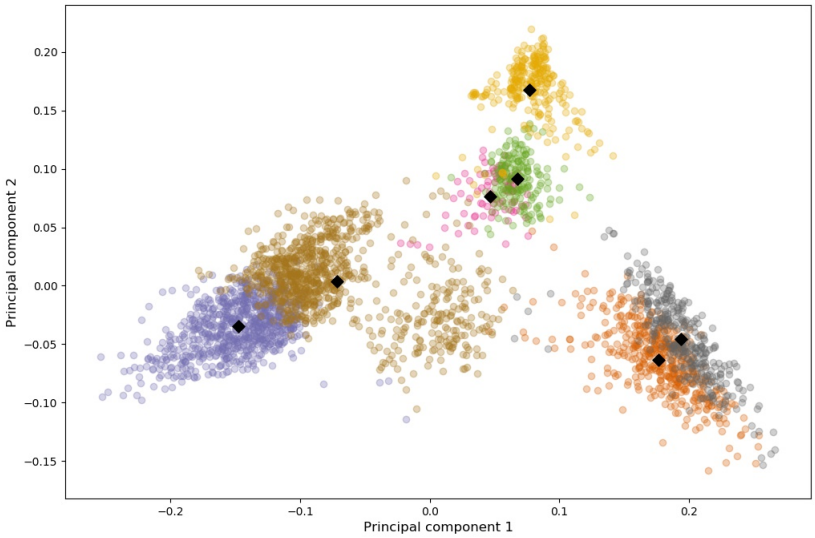
I use the *KMeans* object from the module *cluster* and the *Standardscaler* object from the module *preprocessing* in the Python library *sklearn* to perform the k means clustering. I use the module *pyplot* from the Python library *matplotlib* to make the plots. First, I normalized the original 192 dimensional data[3]. After the normalizatoin, I used 7 means clustering[4] to determine 7 centroids in the 192 dimensional space, which I then used the inverse function of the normalization to transform back. Hereafter, I projected them onto the first two principal components found in the last section. At last, I superimposed the projections of the centroids on the scatter plot from the last section, where they are seen as black diamonds:



Let us see a plot of the clusters according the 7 means clustering, projected onto the first two principal components:

[3] I did this, because kmeans clustering depends on norms between the points in the 192 dimensional input space, and therefore it is good practice to normalize the features.

[4] initialized as described in the assignment text

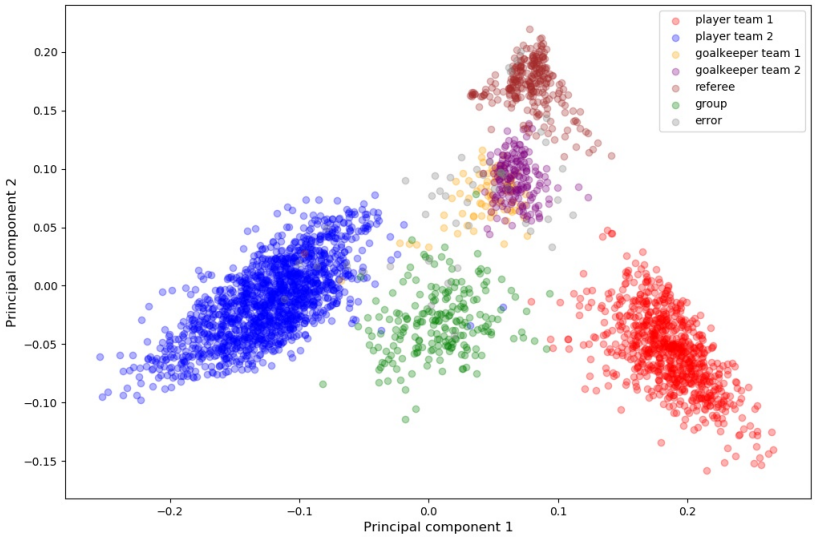


In the context of our specific problem, we could choose to judge the performance of 7 means clustering by how close the clusters, it has found, is to the classes that we are in fact interested in. In that case, it has not done a very good job. It does seem to have found clusters that correspond to the two goalkeeper classes and the referee class. However, it has complete missed the group and the error classes, and it has also split each of the player classes into two parts. I suspect that these flaws could be mainly due to two facts: 1) When we look at the principal components plots in the last section, the error class does not seem to be a cluster in the sense of all of its points generally being closer to each other than to points of the other classes. This means that it does make sense to hope that k means clustering will detect this cluster. 2) The sizes of the classes are very different, which I suspect could cause the optimums of the distortion function[5] to be quite different from the means of the classes. We can look at it like this: If we only had one quite small and one quite large class, then it maybe would not pay off in terms of minimizing the distortion function to put a centroid in the middle of the small class, if we instead could instead spread two centroids across the large class, thereby making all of its many points have a closer distance to a centroid. I suspect that this is what it causing 7 means clustering to prefer having multiple centroids in the two player classes and no centroid in the group class.

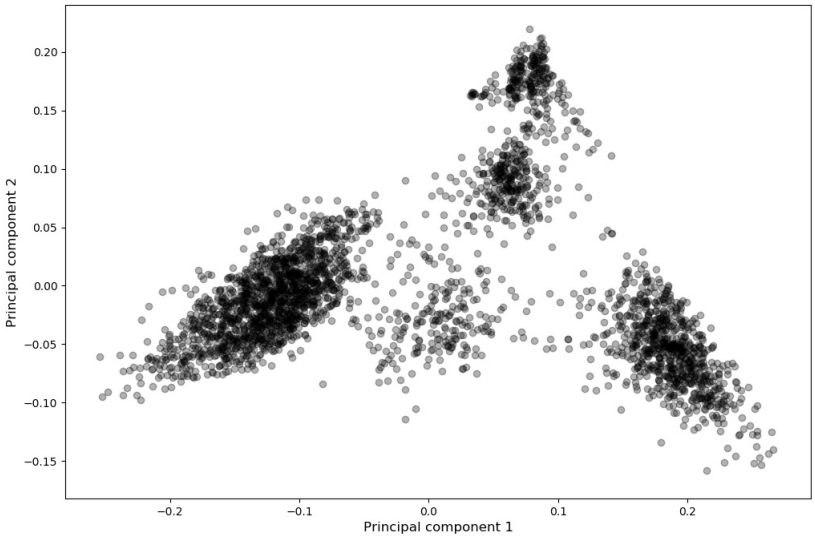
If this diagnosis is partly correct, then a possible fix to the k means clustering in the context of our problem would be to complete remove the error class from our training set, randomly remove some parts of the player classes to make the class sizes more equal, and lastly set the algorithm to only look for 6 clusters now that we have removed the error class.

However, we have to remember that now we are judging k means clustering almost as though it was a supervised learning algorithm. It is not. It is unsupervised, and therefore its world does not look like this

[5]The objective function of k means clustering



but more like this



and all it is doing is to minimize some specific function in such a world with no labels and only features. Therefore, it would also be quite stupid of us humans to afterwards come with our labels and expect that the algorithm has found clusters that match them. All in all, this example serves to show that we have to be very careful with expecting unsupervised methods to come up clusters - or other outputs - that exactly correspond to what we consider relevant.

3.4 Question 4

When we apply machine learning techniques to the sports analytics task, as well as any other task, there is the risk of what John Langford[6] calls parameter tweak overfitting: First, we choose some of our final model's parameters or hyperparameters, based on how the model perform on our test set. Second, we use the performance of the final model on the test set as an estimate of its expected performance, without taking into account that we have used the test set for some of the training of the model as well. This means that we overrepresent how well we expect the model to perform in the future, which is what Langford defines as overfitting. This can happen in subtle ways, if we do not take great care of keeping the test set zealed off from all interference with the training process. In a new area of applying machine learning, such as sports analytics, we might not have a very strong idea about what models tend to work best, and therefore we will maybe start our analysis by doing a bit of exploring of the dataset. Maybe we will do a quick train-test-split, and train and test some simple linear models and also a couple of simple neural networks to see what seems to work best. Okay, the neural networks seem to be best, let's go with them and start the real analysis with another train-test-split of the data. Now we have comitted overfitting by parameter tweaking. We have implicitly tweaked a parameter of our final model, namely which model class to use at all, and afterwards we have hidden the parameter by forgetting it, although we maybe have used data from the final test set to tweak it.

Another clever way that we could overfit in sport analytics is by what Langford calls Human-in-the-loop overfitting. One way this could happen would be in combination with what he calls dataset selection. Consider the example of classification of the protagonists in soccer, which we have worked with in this section of the exam. Here, a human from the sports team administrative department maybe could have preprocessed the data by choosing not to pass on some video material, where the camera is filming the audience of the soccer game, to the machine learning team, because he considers this video material irrelevant to the task. If the same preprocessing does not take place, when the algorithm has to analyse future data sets, then maybe the algorithm will do much worse, because it will confused by the video material of the audience. We will have overrepresented how well we expect the algorithm to perform, because a human interfered with data selection for the the training and testing in a way, so the data became unrepresentable of the real world data, the model will act on in the future.

3.5 Question 5

I have decided to use random forests as my non-linear, multi-label classification method. The reason for this is that it is very simple to use, since we do not have to worry about overfitting as long as we average over a lot of decision trees. I used 50 decision trees and trained each tree on 14[7] randomly sampled features of a sampled subset[8] of the total training set. I use the cross entropy measure to determine the best splits of the data. I used the *RandomForestClassifier* object from the module *ensemble* in the Python library *sklearn* to perform the random forests classification. I get the following training and test accuracy:

Training score	Test score
1.0	0.93

[6] Langford: Clever methods of overfitting, http://hunch.net/?p=22

[7] The ceiling of the squareroot of 192; the number of features in the full training set.

[8] I could not figure out how to specify the size of the subset in the random forests implementation that I used.

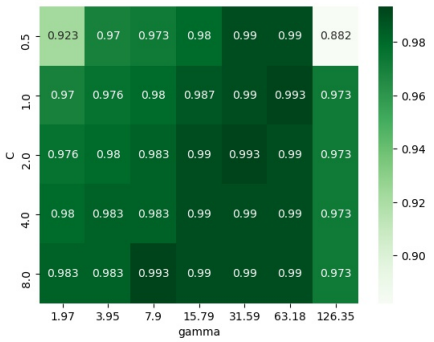
3.6 Question 6

I used the *SVC* object from the module *svm* and the *GridSearchCV* object from the module *model selection* module in the Python library *sklearn* to perform the support vector machine classification and grid search selection of hyper parametres, respectively. I used the module *heatmap* function from the Python library *seaborn* to make the plots.

I started by computing the Jaakkola heuristic for γ and found that:

Jaakoola heuristic for gamma
15.794

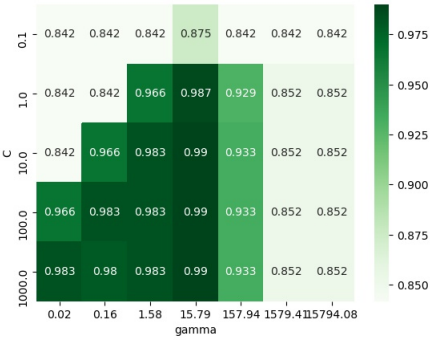
After that I used 3-fold cross validation to choose the hyperparametres C and γ from a grid of all combinations of $C \in \{b^{-1}, 1, b, b^2, b^3\}$ and $\gamma \in \{\gamma_{Jaakkola} \cdot b^i | i \in \{-3, ..., 3\}\}$, where $b = 2$. Here is a heatmap over the resulting cross validation scores:



Here is what I find to be the optimal combination of C and γ and their accuracy scores in the cross validation and on the training and test sets:

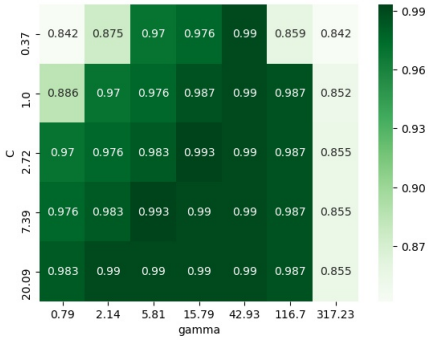
C	gamma	training score	validation score	test score
1.0	63.176	0.997	0.993	0.987

I repeated the process for b equal to 10 and Eulers constant. Here are the results for b equal to 10:



C	gamma	training score	validation score	test score
10.0	15.794	1.0	0.99	0.944

and here are the results for b equal to Eulers constant:



C	gamma	training score	validation score	test score
2.718	15.794	0.997	0.993	0.948

4 Decision Trees and Random Forests

4.1 Question 1

Let $\mathbb{B} = \{-1, 1\}$ be the binary output space. Let $\mathcal{H}_{stumps}$ be the hypothesis class of all one-level decision trees that takes $x \in \mathbb{R}$ as input and gives $y \in \mathbb{B}$ as output. Let me now proof that the VC-dimension of $\mathcal{H}_{stumps}$ is equal to 2.

Let me start by proving that $2 \leq d_{VC}(\mathcal{H}_{stumps})$. From the book **Learning by Data**, page 51, I know that I just need to find one example of a set of 2 points in $\mathbb{R}$ that can be shattered by $\mathcal{H}_{stumps}$.

A set $A = \{x_1, ..., x_n\}$ is shattered by $\mathcal{H}$, if and only if for all binary labelings $\{(x_1, y_1), ..., (x_n, y_n)\}$, $y_1, ..., y_n \in \mathbb{B}$ of A, there exists an hypothesis $h \in \mathcal{H}$ such that h gives this labeling of A, that is, such that $h(x_1) = y_1$, ... , $h(x_n) = y_n$.

Now let $A = \{x_1, x_2\}$ be any subset of $\mathbb{R}$, where $x_1 < x_2$. All possible binary labelings of the points in A can be split into two cases: Either x_1 and x_2 share the same label, or else they don't.

Let us start by assuming that x_1 and x_2 share the same label l. In that case, we can pick $h_{\theta,y} \in \mathcal{H}_{stumps}$ such that $\theta < x_1$ and $y = l$. By the definition of $h_{\theta,y}$ given in the assignment text, we now have that $h_{\theta,y}(x_1) = h_{\theta,y}(x_2) = l$, since both x_1 and x_2 are larger than θ. We have therefore found a hypothesis in $\mathcal{H}_{stumps}$ that gives us the correct labeling.

Assume now instead that x_1 is labeled with l_1, and that x_2 is labeled with l_2, and that $l_1 \neq l_2$. In that case, we can pick $h_{\theta,y} \in \mathcal{H}_{stumps}$ such that $\theta = \frac{l_1+l_2}{2}$ is the midpoint between x_1 and x_2, and $y = l_2$. By the definition of $h_{\theta,y}$, we now have that $h_{\theta,y}(x_1) = -l_2 = l_1$ and that $h_{\theta,y}(x_2) = l_1$. We have therefore found a hypothesis in $\mathcal{H}_{stumps}$ that gives us the correct labeling.

We have now shown that for all binary labeling of the points in A, we can find a hypothesis in $\mathcal{H}_{stumps}$ that gives us this labeling. Since the assumptions about A are true of for instance the set $\{-10, 10\}$, we have thereby shown that $\mathcal{H}_{stumps}$ can shatter this set. Therefore, there exists a set of two points in $\mathbb{R}$ that can be shattered by $\mathcal{H}_{stumps}$, and we now know that $2 \leq d_{VC}(\mathcal{H}_{stumps})$.

Let me now proof that $d_{VC}(\mathcal{H}_{stumps}) < 3$. From the book **Learning by Data**, page 52, I know that I need to show that no sets of 3 points in $\mathbb{R}$ can be shattered by $\mathcal{H}_{stumps}$.

Assume therefore for contradiction that there exists a set of 3 points from $\mathbb{R}$ that can be shattered by $\mathcal{H}_{stumps}$. Call this set A. Since A is a set, we know that its elements are unique. Since the real numbers are a totally ordered set, we know that we can index the three, unique elements in A such that

$$A = \{x_1, x_2, x_3\}, \quad x_1 < x_2 < x_3 \tag{15}$$

One way of labeling A is $(x_1, -1), (x_2, 1)$ and $(x_3, -1)$. We can visualize A with this labeling like this:

where blue and red corresponds to the label -1 and 1, respectively.

Since A can be shattered by $\mathcal{H}_{stumps}$, we know that there exist a hypothesis $h_{\theta,y} \in \mathcal{H}_{stumps}$ that gives this labeling of A. In other words, there exist a hypothesis $h_{\theta,y} \in \mathcal{H}_{stumps}$ such that

$$[h_{\theta,y}(x_1) = -1] \ \wedge \ [h_{\theta,y}(x_2) = 1] \ \wedge \ [h_{\theta,y}(x_3) = -1] \tag{16}$$

Assume for contradiction that $\theta \leq x_2$. Since x_2 and x_3 are now both larger than θ, we know that $h_{\theta,y}$ must assign the same label to them. This contradicts line (16). Therefore $\theta \leq x_2$ cannot be true.

Assume now for contradiction that $x_2 < \theta$. Since x_1 and x_2 are now both strictly smaller than θ, we know that $h_{\theta,y}$ must assign the same label to them. This contradicts line (16). Therefore $x_2 < \theta$ cannot be true.

We have now reached the conclusion that θ and x_2 are two real numbers, where it is not true that

$$\theta \leq x_2 \ \vee \ x_2 < \theta \tag{17}$$

This is a contradiction of the fact that the real numbers are a totally ordered set. The contradiction follows from directly from our assumption that there exist a set of 3 points from $\mathbb{R}$ that can be shattered by $\mathcal{H}_{stumps}$. Therefore, this assumption must be false. Therefore, it is true that no set of 3 points from $\mathbb{R}$ can be shattered by $\mathcal{H}_{stumps}$. Therefore, $d_{VC}(\mathcal{H}_{stumps}) < 3$.

All in all, I have now shown that $2 \leq d_{VC}(\mathcal{H}_{stumps})$ and $d_{VC}(\mathcal{H}_{stumps}) < 3$. Since the VC-dimension is a natural number, this implies that $d_{VC}(\mathcal{H}_{stumps}) = 2$.

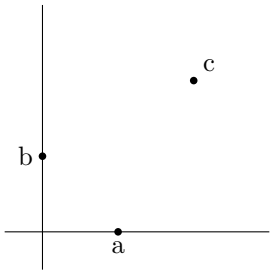
4.2 Question 2

Let $\mathcal{H}$ be the hypothesis class of all one-level decision trees that takes $x \in \mathbb{R}^2$ as input and gives $y \in \mathbb{B}$ as output. I denote each hypothesis in $\mathcal{H}$ with $h_{(\theta,d),y}$, where $\theta \in \mathbb{R}$, $d \in \{1, 2\}$ and $y \in \{-1, 1\}$. A given hypothesis $h_{(\theta,d),y}$ is defined by $h_{(\theta,d),y}(x) = y$, if and only if $\theta \leq x_d$, and otherwise $h_{(\theta,d),y}(x) = -y$.

Let me now proof that $3 \leq d_{VC}(\mathcal{H})$. To do this, I just need to find one example of a set of 3 points in $\mathbb{R}^2$ that can be shattered by $\mathcal{H}_{stumps}$. Consider the set

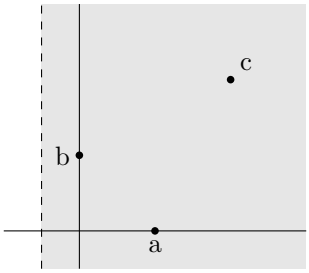
$$A = \{(1,0),(0,1),(2,2)\} = \{(a_1,a_2),(b_1,b_2),(c_1,c_2)\} = \{a,b,c\} \tag{18}$$

I will now proof that this set can be shattered by $\mathcal{H}$. When we visualize A in the plane, it looks like this



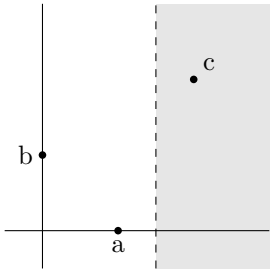
Let me now show that for all possible binary labelings of A, there exists a hypothesis in $\mathcal{H}$ that labels A that way. All possible binary labelings of the points in A can be split into two cases: Either a, b and c all share the same label, or else only two of them share a label, and the last one has a different label.

Let us start by assuming that a, b and c all share the same label l. In that case, we can pick $h_{(\theta,d),y} \in \mathcal{H}$ such that $(\theta, d) = (-0.5, 1)$ and $y = l$. This way the hypothesis will split $\mathbb{R}^2$ like this



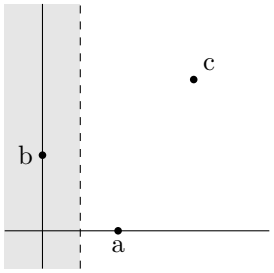
where the grey area is area labeled with l, and the white area is the area labeled with $-l$. We see that the labeling is correct.

Let us now assume that only two of a, b and c share a label, and the last one has a different label. Let us first assume that a and b share the label l_1 and c has the label l_2, where $l_1 \neq l_2$. In that case, we can pick $h_{(\theta,d),y} \in \mathcal{H}$ such that $(\theta, d) = (1.5, 1)$ and $y = l_2$. This way the hypothesis will split $\mathbb{R}^2$ like this



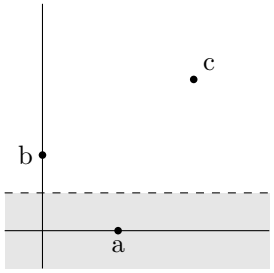
where the grey area is area labeled with l_2, and the white area is the area labeled with $-l_2 = l_1$. We see that the labeling is correct.

Let us now instead assume that a and c share the label l_1 and b has the label l_2, where $l_1 \neq l_2$. In that case, we can pick $h_{(\theta,d),y} \in \mathcal{H}$ such that $(\theta, d) = (0.5, 1)$ and $y = l_1$. This way the hypothesis will split $\mathbb{R}^2$ like this



where the grey area is area labeled with l_2, and the white area is the area labeled with $-l_2 = l_1$. We see that the labeling is correct.

Let us now instead assume that b and c share the label l_1 and a has the label l_2, where $l_1 \neq l_2$. In that case, we can pick $h_{(\theta,d),y} \in \mathcal{H}$ such that $(\theta, d) = (0.5, 2)$ and $y = l_1$. This way the hypothesis will split $\mathbb{R}^2$ like this



where the grey area is area labeled with l_2, and the white area is the area labeled with $-l_2 = l_1$. We see that the labeling is correct.

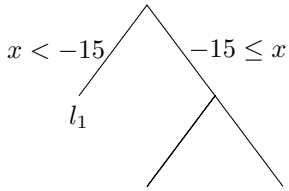
We have now shown that for all labeling of the points in A with labels -1 or 1, we can find a hypothesis in $\mathcal{H}$ that gives us this labeling. We have therefore that there exists a set of 3 points in $\mathbb{R}^2$ that can be shattered by $\mathcal{H}$, which implies that $3 \leq d_{VC}(\mathcal{H}_{stumps})$.

4.3 Question 3

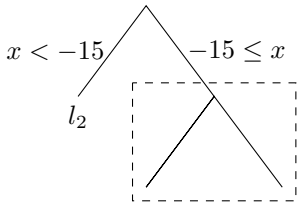
Let $\mathcal{H}$ be the hypothesis class of all two-level, incomplete decision trees with only three leafs that takes a real number as input and gives -1 or 1 as output. Let me now proof that $3 \leq d_{VC}(\mathcal{H})$. To do this, I just need to construct one example of a set of 3 points in $\mathbb{R}$ that can be shattered by $\mathcal{H}_{stumps}$. Consider the set

$$A = \{-20, -10, 10\} = \{a, b, c\} \tag{19}$$

Let $(-20, l_1), (-10, l_2), (10, l_3)$ be some arbitrary binary labeling of A. Let $\mathcal{J}$ be the subset of $\mathcal{H}$ with all the decision trees on the form



That is, for all the decision trees in $\mathcal{J}$ the threshold of the first decision node is equal to -15, and all inputs that are strictly below -15 are classified with the label l_1. It clear that all decision trees in $\mathcal{J}$ will label the point $a = -20$ correctly with the label l_1. Neither of the points $b = -10$ and $c = 10$ are below -15, and therefore all decision trees in $\mathcal{J}$ will send them both to the right in the first decision node, whereafter they will be classified according to the threshold of the remaining decision stump:



The question is therefore now if there, given some arbitrary binary labeling of $\{b, c\} = \{-10, 10\}$, exists a hypothesis h in the class of all decision stumps such that h labels b and c correctly. In question 1, I have already proved that this is the case.

All in all, we now know that for all binary labelings of A there exists a hypothesis h in $\mathcal{J}$ such that h gives that labeling. Since $\mathcal{J}$ is a subset of $\mathcal{H}$, we therefore know that there exists a set A of 3 real numbers such that A is shattered by $\mathcal{H}$. Therefore, we have that $3 \leq d_{VC}(\mathcal{H})$.

4.4 Question 4

Let $d_{k,m}$ be the VC-dimension of the hypothesis class $\mathcal{H}_{k,m}$ of decision trees with k leaves for m-dimensional input data. From **Theorem 3.7** in Yevgeny's lecture notes, it follows directly that with probability at least $1 - \delta$ for all $h \in \mathcal{H}_{k,m}$

$$L(h) \leq \hat{L}(h, S) + \sqrt{\frac{8 \ln \left(\frac{2\left[(2n)^{d_{k,m}} + 1\right]}{\delta} \right)}{n}} \tag{20}$$

The only quantity in the bound that grows, when the size k of the trees and the dimensionality m of input space grow, is the VC-dimension $d_{k,m}$. Therefore, it is only the complexity penalty term

$$\sqrt{\frac{8 \ln \left(\frac{2\left[(2n)^{d_{k,m}} + 1\right]}{\delta} \right)}{n}} \tag{21}$$

that grows with bigger k and m. However, with bigger k and m, the minimal empirical error $\hat{L}(h, S)$ across all hypothesis $h \in \mathcal{H}_{k,m}$ will also tend to get smaller, since we include more and

more complex hypothesis in $h \in \mathcal{H}_{k,m}$. Therefore, it is far from trivial whether the minimal of the upper bounds

$$\hat{L}(h,S) + \sqrt{\frac{8 \ln\left(\frac{2\left[(2n)^{d_{k,m}}+1\right]}{\delta}\right)}{n}} \tag{22}$$

across all hypothesis $h \in \mathcal{H}_{k,m}$ tested on a given set S will get smaller or larger with bigger k and m.

4.5 Question 5

Let S be a training set of size n, where n is a even number. Let RF be a random forests consisting of B decision trees $\mathcal{D} = \{h_1, ..., h_B\}$, where each tree h_i is trained on a subset S_i of $\frac{n}{2}$ points sampled without replacement from S. Let $\overline{S_i}$ denote the subset of S, which was not used to train h_i.

Let me now bound the expected loss $L(h_1)$ of the first tree h_1 in terms of the empirical loss $\hat{L}(h_1, \overline{S_1})$. Since $|\overline{S_1}| = \frac{n}{2}$ and we are not trying to bound the expected loss of more than a single hypothesis being tested on $\overline{S_1}$, we can simply use **Theorem 3.1** from Yevgeny's lecture notes to conclude that with probability at least $1-\delta$:

$$L(h_1) \leq \hat{L}(h_1, \overline{S_1}) + \sqrt{\frac{\ln \frac{1}{\delta}}{n}} \tag{23}$$

It is clear that there is nothing special about h_1, but that this bound holds for all $h_i \in \mathcal{D}$, if we only test h_i and not the other hypothesis in $\mathcal{D}$. However, if we want to test all the hypothesis $h_i \in \mathcal{D}$, then the above bound will not hold with probability at least $1-\delta$ simultaneously for all of them, since there will be a probability of $1-\delta$ for each of the tested hypothesis to break the bound. However, we can do the exact same trick as in the proof of **Theorem 3.2** from Yevgeny's lecture notes and take the union bound:

$$\mathbb{P}\left\{\exists h_i \in \mathcal{D} : L(h_i) \geq L(h_i, \overline{S_i}) + \sqrt{\frac{\ln \frac{B}{\delta}}{n}}\right\} \tag{24}$$

$$\leq \sum_{i=1}^{B} \mathbb{P}\left\{L(h_i) \geq L(h_i, \overline{S_i}) + \sqrt{\frac{\ln \frac{1}{\frac{\delta}{B}}}{n}}\right\} \tag{25}$$

$$\leq \sum_{i=1}^{B} \frac{\delta}{B} = \delta \tag{26}$$

We therefore have that with at least probability $1-\delta$ for all $h_i \in \mathcal{D}$:

$$L(h_i) \leq L(h_i, \overline{S_i}) + \sqrt{\frac{\ln \frac{B}{\delta}}{n}} \tag{27}$$

Let me now proof that the expected error $L(RF)$ of the random tree can be bounded by the expected error of its decision trees in the following way:

$$L(RF) \leq \frac{2}{B} \sum_{i=1}^{B} L(h_i) \tag{28}$$

Consider first the case, where the random forests makes a wrong prediction of the label y of a point x. In this case, we have that

$$\ell(RF(x), y) = 1 \tag{29}$$

Since the random forests predicting y given x implies that more than half of its trees have made the same prediction, we must have that more than half of the trees were also wrong in their predictions. Therefore, we must have that

$$0.5 \leq \frac{1}{B} \sum_{i=1}^{B} \ell(h_i(x), y_i) \tag{30}$$

where y_i is the prediction of the tree h_i given the input x. From this, it clearly follows if $\ell(RF(x), y) = 1$, then

$$\ell(RF(x), y) \leq \frac{2}{B} \sum_{i=1}^{B} \ell(h_i(x), y_i) \tag{31}$$

If $\ell(RF(x), y) = 0$, then this is clearly also true: The right hand side is always at least 0, since it consists of a positive constant multiplied with a sum of quantities always at least 0. If we now take the expectation on both sides of line (31)[9], we get

$$\mathbb{E}[\ell(RF(x), y)] \leq \mathbb{E}\left[\frac{2}{B} \sum_{i=1}^{B} \ell(h_i(x), y_i)\right] = \frac{2}{B} \sum_{i=1}^{B} \mathbb{E}[\ell(h_i(x), y_i)] \tag{32}$$

By definition, this means that

$$L(RF) \leq \frac{2}{B} \sum_{i=1}^{B} L(h_i) \tag{33}$$

which is what I wanted to show. Since the inequality on line (27) holds with probability at least $1 - \delta$ for all $h_i \in \mathcal{D}$, we can now conclude that with probability at least $1 - \delta$:

$$L(RF) \leq \frac{2}{B} \sum_{i=1}^{B} \left[L(h_i, \overline{S_i}) + \sqrt{\frac{\ln \frac{B}{\delta}}{n}}\right] \tag{34}$$

5 Rotation of the inputs

In this section, I will start by proving that rotation of the input space does not affect k nearest neighbours with euclidean distance metric or support vector machines with Gaussian radial

[9] We can do this, since if one random variable is always greater than another random variable, then the same holds true for their expectations.

kernels. Hereafter, I will give an example that shows that random forests can be affected by rotation of the inputs. The intution behind the proofs is actually quite simple, namely that k nearest neighbours and svms with Gaussian radial kernels only use values from the input space through their euclidean distances and not anything else. Therefore, these algorithms are unaffected by rotation of the input space, since this operation does not affect the euclidean distances between the vectors of the space. Random forests, on the other hand, does not only use values from the input space through their euclidean distances. Therefore, it makes sense that random forests can be affected by rotation of the inputs, and we will show that there in fact exists an example, where this is the case. This was the intuition. The rest is an attempt to make this intuition precise.

Let $\mathbb{B} = \{-1, 1\}$ be the binary label space. Consider the input space $\mathbb{R}^d$ for some $d \in \mathbb{N}$. Let A be some learning algorithm that as input takes some hyperparametres θ and a training set S sampled from $\mathbb{R}^d \times \mathbb{B}$, and as output gives some classifier $h : \mathbb{R}^d \to \mathbb{B}$. Let e be the euclidean distance on $\mathbb{R}^d$. If the *only* way that values v, w from the input space $\mathbb{R}^d$ enters A is through their euclidean distance $e(v, w) = ||v - w||$, then I say that A is a **euclidean learner**.

Both k nearest neighbours and svms with Gaussian radial kernels are examples of euclidean learners. If we start by looking at the pseudocode[10] of k nearest neighbours with euclidean distance metric, we see the first thing it does, when it has to classify a new input v, is exactly to calculate $e(v, x_i)$ for all $(x_i, y_i) \in S$. After that, it does not look at the input values $x_1, ..., x_n$ from S or the value of v, but only on the values[11] $e(v, x_1), ..., e(v, x_n)$. Therefore, k nearest neighbours with euclidean distance metric is a euclidean learner.

Regarding non-linear svms, we know in general that values from the input space only enters the algorithm through the kernel function k of the svm[12]. This fact is the entire foundation of the kernel trick on which non-linear svms are built. The Gaussian radial kernel k for input space $\mathbb{R}^d$ and hyperparameter $\gamma \in \mathbb{R}_+$ is defined as

$$\forall x, z \in \mathbb{R}^d : \; k(x, z) = \exp(-\gamma ||x - z||^2) = \exp(-\gamma (e(x, z))^2) \tag{35}$$

Since the argumets x and z of this kernel only enters the kernel through their euclidean distance, a svm with Gaussian radial kernel is an euclidean learner.

Let me now argue that all euclidean learners are unaffected by rotation of the input space. Let A be an euclidean learner, as I have defined it. Let $S = \{(x_1, y_1), ..., (x_n, y_n)\}$ be a training set with input values x_i from $\mathbb{R}^d$ and labels y_i from $\mathbb{B}$. Let $A_\theta(S)$ denote the classifier $h : \mathbb{R}^d \to \mathbb{B}$, which is outputted by the algorithm A when given the hyperparameters θ and the training set S as input. Let $\mathbf{R}$ be any d-dimensional rotation matrix, as defined in the assignment text. Let $\mathbf{R}S = \{(\mathbf{R}x_1, y_1), ..., (\mathbf{R}x_n, y_n)\}$ be the training set with rotated input values. Let me now argue that for all $w \in \mathbb{R}^d$ we have that

$$[A_\theta(S)](w) = [A_\theta(\mathbf{R}S)](\mathbf{R}w) \tag{36}$$

In other words, let me argue that for all $w \in \mathbb{R}^d$, the result from training A_θ on the original training set S and thereafter using it to classify the input value w, is equal to the result from training A_θ on the rotated training set $\mathbf{R}S$ and thereafter using it to classify the rotated input value $\mathbf{R}w$.

[10] See Yevgeny's lecture notes, page 4.
[11] In the pseudocode in the lecture notes, these values are called $d_1, ..., d_n$.
[12] Christian Igel's Machine Learning: Kernel-based Methods, page 29, page 32.

By the standard rules from linear algebra and the properties of rotation matrices, we get that for all $v, w \in \mathbb{R}^d$

$$(e(\mathbf{R}v, \mathbf{R}w))^2 \tag{37}$$
$$= ||\mathbf{R}v - \mathbf{R}w||^2 \tag{38}$$
$$= (\mathbf{R}v - \mathbf{R}w)^T(\mathbf{R}v - \mathbf{R}w) \tag{39}$$
$$= ((\mathbf{R}(v - w)))^T(\mathbf{R}(v - w)) \tag{40}$$
$$= ((v - w)^T\mathbf{R}^T)(\mathbf{R}(v - w)) \tag{41}$$
$$= (v - w)^T(\mathbf{R}^T\mathbf{R})(v - w) \tag{42}$$
$$= (v - w)^T(\mathbf{R}^{-1}\mathbf{R})(v - w) \tag{43}$$
$$= (v - w)^T(v - w) \tag{44}$$
$$= ||v - w||^2 \tag{45}$$
$$= (e(v, w))^2 \tag{46}$$

By this, it follows directly that for all for all $v, w \in \mathbb{R}^d$

$$e(\mathbf{R}v, \mathbf{R}w) = e(v, w) \tag{47}$$

In other words, we have that if we use a rotation matrix $\mathbf{R}$ to rotate the inpute space $\mathbb{R}^d$, then the euclidean distance between all elements of the space remain unchanged. This directly implies that for all $w \in \mathbb{R}^d$ and $(x_1, y_n), ..., (x_n, y_n)$ in S, the pairwise euclidean distances between all elements in $\{x_1, ..., x_n, w\}$ are exactly the same as the pairwise euclidean distances between all elements in $\{\mathbf{R}x_1, ..., \mathbf{R}x_n, \mathbf{R}w\}$. From this, it follows that

$$[A_\theta(S)](w) = [A_\theta(\mathbf{R}S)](\mathbf{R}w) \tag{48}$$

since the only difference between the left and the right side of this equation is that the values from input space on the left side are $\{x_1, ..., x_n, w\}$, and on the right side are $\{\mathbf{R}x_1, ..., \mathbf{R}x_n, \mathbf{R}w\}$. However, since A is a euclidean learner, these input values only enters A through their pairwise euclidean distances, which I have just argued are the same on both sides.

By line (48), it follows directly that for all loss functions $\ell : \mathbb{B}^2 \to \mathbb{R}$ and labeled input values $(w, z) \in \mathbb{R}^d \times \mathbb{B}$, we have that

$$\ell[[A_\theta(S)](w),\ z] = \ell[[A_\theta(\mathbf{R}S)](\mathbf{R}w),\ z] \tag{49}$$

Let $T = \{(w_1, z_1), ..., (w_m, z_m)\}$ be some test set sampled from $\mathbb{R}^d \times \mathbb{B}$. Let $T = \{(\mathbf{R}w_1, z_1), ..., (\mathbf{R}w_m, z_m)\}$ be the test set with rotated input values. Given the loss function ℓ, the empirical error of $A_\theta(S)$ on T is defined as

$$\hat{L}[A_\theta(S), T] = \frac{1}{m}\sum_{i}^{m} \ell[[A_\theta(S)](w_i), z_i] \tag{50}$$

From this definition and line (49), it follows that

$$\hat{L}[A_\theta(S), T] = \hat{L}[A_\theta(\mathbf{R}S), \mathbf{R}T] \tag{51}$$

All in all, I have now shown that for all euclidean learners A, all choices of hyperparameters θ for A, and all training and test sets S and T sampled from $\mathbb{R}^d \times \mathbb{B}$, we have that the empirical

loss from training A_θ on S and then testing it on T is equal to the empirical loss from training A_θ on the rotated training set $\mathbf{R}S$ and then testing it on the rotated test set $\mathbf{R}T$.

Since I have also argued that k nearest neighbours with euclidean distance metric, and svms with Gaussian radial kernels are euclidean learners, I have hereby shown that they are both unaffected by rotation[13] of the input space in the sense that their performance - measured by the empirical error on test sets - is the same, whether the input space is rotated or not.

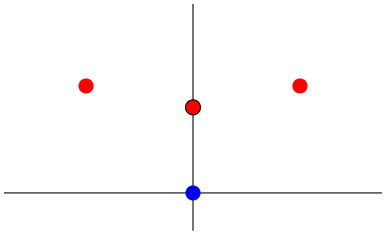
Let me now give an example that shows that random forests can be affected by rotation of the input space. Let D be a binary classification tree that accepts input from $\mathbb{R}^2$. Let the training set S from $\mathbb{R}^2 \times \mathbb{B}$ be defined as

$$S = \left\{ \left(\left(-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2} \right), 1 \right), \left(\left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2} \right), 1 \right), (0,0), -1) \right\} \tag{52}$$

Let the test set T from $\mathbb{R}^2 \times \mathbb{B}$ be defined as

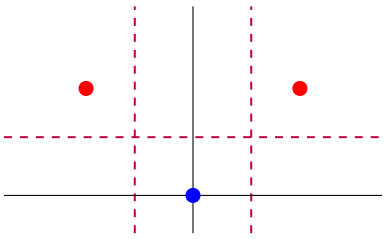
$$T = \left\{ \left(\left(0, -\frac{2\sin(\pi/4)}{5} + \frac{2\cos(\pi/4)}{5} \right), 1 \right) \right\} \tag{53}$$

S and T looks like this:



where red corresponds to the label 1, blue corresponds to the label -1, and the red dot with the black border is the test point.

If we train D on S, the first iteration of the training will try splitting the data with these thresholds[14]:



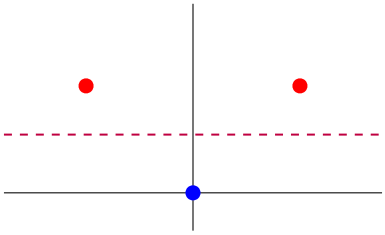
That is, the first node of D tries to split the data with the thresholds[15] $(\theta_1, d_1) = (0.5, 1)$,

[13] Actually, I do not use the fact that the rotation $\mathbf{R}$ matrix has determinant 1, which means that my proof works for all matrices in the d-dimensional orthogonal group, not only all matrices in the d-dimensional specially orthogonal group. Geometrically, this corresponds to including mirroring operations on $\mathbb{R}^d$, as well as rotations. This makes sense, since mirroring also preserves euclidean distances.

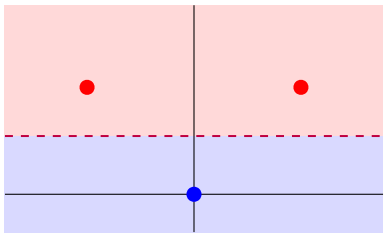
[14] I follow Christian Igel's slides about decision trees and random forests and say that each node in the decision tree tries to split the data along all dimensions in the data on each of the means of two succesive data points projected onto the given dimension [slide 14].

[15] I follow Christian Igel's slides about decision trees and random forests and parametrize each split by a pair (θ, d), where $\theta \in \mathbb{R}$ is the threshold value and $d \in \{1, 2\}$ is the dimension, which is used to split the data according to the threshold θ [slide 7].

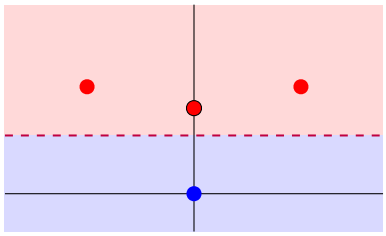
$(\theta_2, d_2) = (-0.5, 1)$ and $(\theta_3, d_3) = (0.5, 2)$. It is clear that by any impurity measure, the threshold $(0.5, 2)$:



gives rise to the purest split of the data. Since all the splitted subsets of the data are now pure, the algorithm terminates. It therefore ends up with a hypothesis that splits $\mathbb{R}^2$ like this:



where the red part is labeled with 1, and the blue part is labeled with -1. If we add the test point to the plot, we see that it is classified correctly by this hypothesis:



Now, let the $\mathbf{R}$ be the matrix

$$\mathbf{R} = \begin{bmatrix} \cos(7\pi/4) & -\sin(7\pi/4) \\ \sin(7\pi/4) & \cos(7\pi/4) \end{bmatrix} \tag{54}$$

It is known that $\mathbf{R}$ is the rotation matrix that rotates $\mathbb{R}^2$ 45 degrees clockwise. By using $\mathbf{R}$ on the input values of our training and test sets, we get

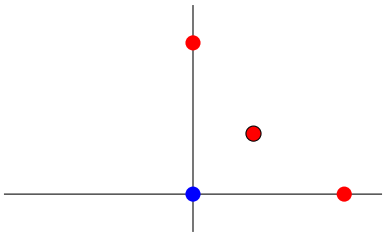
$$\mathbf{R}S = \left\{ \left(\mathbf{R}\left(-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2} \right), 1 \right), \left(\mathbf{R}\left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2} \right), 1 \right), (\mathbf{R}(0,0), -1) \right\} \tag{55}$$

$$= \left\{ ((0,1),1), ((1,0),1), ((0,0),-1) \right\} \tag{56}$$

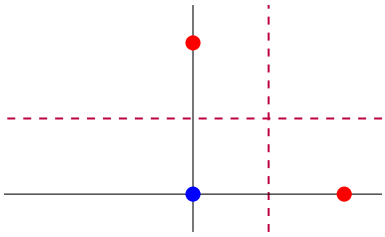
and

$$\mathbf{R}T = \left\{ \left(\mathbf{R}\left(0, -\frac{2\sin(\pi/4)}{5} + \frac{2\cos(\pi/4)}{5} \right), 1 \right) \right\} = \left\{ \left(\left(\frac{2}{5}, \frac{2}{5} \right), 1 \right) \right\} \tag{57}$$

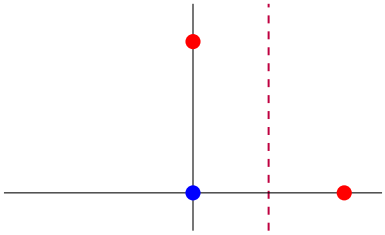
Visually, the rotated training and test sets look like this:



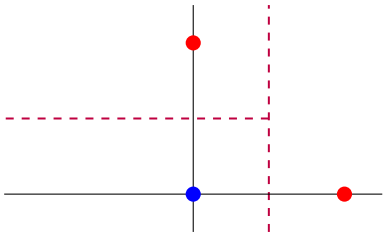
If we train D on $\mathbf{R}S$, the first node will try splitting the data with these thresholds:



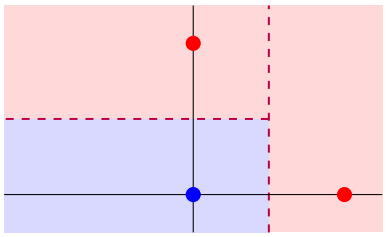
That is, the node will try the thresholds $(\theta_1, d_1) = (0.5, 1)$ and $(\theta_2, d_2) = (0.5, 2)$. Since the resulting split of data for the two thresholds are the same in terms of purity, the algorithm will choose a random of them, say $(\theta_1, d_1) = (0.5, 1)$:



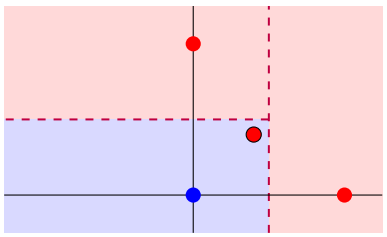
Since the split resulted in one pure subset of the data, this branch will turn into a leaf, and we will only continue the splitting on the other branch. On this branch, there is only one split to try, which will therefore be chosen:



Since all the splitted subsets of the data are now pure, the algorithm terminates. It therefore ends up with a hypothesis that splits $\mathbb{R}^2$ like this:



If we add the test point to the plot, we see that it is classified wrongly by this hypothesis:



I have hereby constructed an example of a training set S, a test set S, a rotation matrix $\mathbf{R}$ and a decision tree D such that

$$\hat{L}(D(S), T) \neq \hat{L}(D(\mathbf{R}S), \mathbf{R}T) \tag{58}$$

where $D(S)$ denotes the hypothesis, we get from training D on S. That is, I have constructed a decision tree that have a different performance when trained on a set S and thereafter tested on a set T than when trained on the rotated set $\mathbf{R}S$ and thereafter tested on the rotated set $\mathbf{R}T$. Hereby, I have shown that decision trees can be affected by rotation of their input spaces. Since the output of a random forests is a function of the outputs of all its decision trees, I have hereby also shown that a random forests can affected by rotation of its input space.