Machine Learning Exercises: Set 3

Roger Garriga Calleja

February 27, 2017

Problem 9 (Rademacher averages.) Let A be a bounded subset of $\Re^n$. Define the Rademacher average

$$R_n(A) = \mathbb{E} \sup_{a \in A} \frac{1}{n} \left| \sum_{i=1}^{n} \sigma_i a_i \right| \tag{1}$$

where $\sigma_1, \ldots, \sigma_n$ are independent random variables with $\mathbb{P}(\sigma_i = 1) = \mathbb{P}(\sigma_i = -1) = \frac{1}{2}$ and $a_1, \ldots, a_n$ are the components of the vector a. Let $A, B \subset \Re^n$ be bounded sets and let $c \in \Re$ be a constant. Prove the following "structural" results:

$$R_n(A \cup B) \leqslant R_n(A) + R_n(B), \quad R_n(c \cdot A) = |c| R_n(A), \quad R_n(A \oplus B) \leqslant R_n(A) + R_n(B), \tag{2}$$

where $c \cdot A = \{ca : a \in A\}$ and $A \oplus B = \{a + b : a \in A, b \in B\}$. Moreover, if absconv($A$)$= \left\{ \sum_{j=1}^{N} c_j a^{(j)} : N \in \mathbb{N}, \sum_{j=1}^{N} |c_j| \leqslant 1, a^{(j)} \in A \right\}$ is the absolute convex hull of A, then

$$R_n(A) = R_n(\text{absconv}(A)). \tag{3}$$

The Rademacher complexity of the union will be

$$R_n(A \cup B) = \mathbb{E} \sup_{e \in A \cup B} \frac{1}{n} \left| \sum_{i=1}^{n} \sigma_i e_i \right|, \tag{4}$$

where $e_1, \ldots, e_n$ are the components of the vector e. Now, since either $e \in A$ or $e \in B$,

$$\sup_{e \in A \cup B} \frac{1}{n} \left| \sum_{i=1}^{n} \sigma_i e_i \right| = \sup \left\{ \sup_{a \in A} \frac{1}{n} \left| \sum_{i=1}^{n} \sigma_i a_i \right|, \sup_{b \in B} \frac{1}{n} \left| \sum_{i=1}^{n} \sigma_i b_i \right| \right\}. \tag{5}$$

So, as both supremums are positive,

$$\sup_{e \in A \cup B} \frac{1}{n} \left| \sum_{i=1}^{n} \sigma_i e_i \right| \leqslant \sup_{a \in A} \frac{1}{n} \left| \sum_{i=1}^{n} \sigma_i a_i \right| + \sup_{b \in B} \frac{1}{n} \left| \sum_{i=1}^{n} \sigma_i b_i \right|. \tag{6}$$

Finally, using the linearity of the expectation,

$$R_n(A \cup B) = \mathbb{E} \sup_{e \in A \cup B} \frac{1}{n} \left| \sum_{i=1}^{n} \sigma_i e_i \right| \leqslant \mathbb{E} \left\{ \sup_{a \in A} \frac{1}{n} \left| \sum_{i=1}^{n} \sigma_i a_i \right| + \sup_{b \in B} \frac{1}{n} \left| \sum_{i=1}^{n} \sigma_i b_i \right| \right\} = \tag{7}$$

$$= \mathbb{E} \sup_{a \in A} \frac{1}{n} \left| \sum_{i=1}^{n} \sigma_i a_i \right| + \mathbb{E} \sup_{b \in B} \frac{1}{n} \left| \sum_{i=1}^{n} \sigma_i b_i \right| = R_n(A) + R_n(B). \tag{8}$$

Let's prove now that $R_n(c \cdot A) = |c| R_n(A)$:

$$R_n(c \cdot A) = \mathbb{E} \sup_{e \in c \cdot A} \frac{1}{n} \left| \sum_{i=1}^{n} \sigma_i e_i \right|, \tag{9}$$

since $\forall e \in c \cdot A$, $e = c \cdot a$ for some $a \in A$,

$$R_n(c \cdot A) = \mathbb{E} \sup_{a \in A} \frac{1}{n} \left| \sum_{i=1}^{n} \sigma_i c a_i \right| = |c| \mathbb{E} \sup_{a \in A} \frac{1}{n} \left| \sum_{i=1}^{n} \sigma_i a_i \right| = |c| R_n(A). \tag{10}$$

Let's prove now that $R_n(A \oplus B) \leqslant R_n(A) + R_n(B)$:

$$R_n(A \oplus B) = \mathbb{E} \sup_{e \in A \oplus B} \frac{1}{n} \left| \sum_{i=1}^{n} \sigma_i e_i \right|, \tag{11}$$

since $\forall e \in A \oplus B$, $e = a + b$ for some $a \in A$ and $b \in B$, applying the triangle inequality

$$R_n(A \oplus B) = \mathbb{E} \sup_{a \in A, b \in B} \frac{1}{n} \left| \sum_{i=1}^{n} \sigma_i (a_i + b_i) \right| = \mathbb{E} \sup_{a \in A, b \in B} \frac{1}{n} \left| \sum_{i=1}^{n} \sigma_i a_i + \sum_{i=1}^{n} \sigma_i b_i \right| \leqslant \tag{12}$$

$$\leqslant \mathbb{E} \sup_{a \in A, b \in B} \frac{1}{n} \left\{ \left| \sum_{i=1}^{n} \sigma_i a_i \right| + \left| \sum_{i=1}^{n} \sigma_i b_i \right| \right\} = \tag{13}$$

$$= \mathbb{E} \sup_{a \in A} \frac{1}{n} \left| \sum_{i=1}^{n} \sigma_i a_i \right| + \mathbb{E} \sup_{b \in B} \frac{1}{n} \left| \sum_{i=1}^{n} \sigma_i b_i \right| = R_n(A) + R_n(B). \tag{14}$$

Finally, let's prove that $R_n(A) = R_n(\text{absconv}(A))$ (to simplify notation I'll call $\text{absconv}(A) = H(A)$):

$$R_n(H(A)) = \mathbb{E} \sup_{e \in H(A)} \frac{1}{n} \left| \sum_{i=1}^{n} \sigma_i e_i \right|, \tag{15}$$

where $e_i \in H(A)$. It is clear that $R_n(A) \leqslant R_n(H(A))$ because $A \subset H(A)$. Now, once fixed N and the c_j's, we define $H_{N,c_j\text{'s}}(A) = \left\{ \sum_{j=1}^{N} c_j a^{(j)} : a^{(j)} \in A \right\} = c_1 A \oplus c_2 A \oplus \cdots \oplus c_N A$. Since $\sum_{j=1}^{N} |c_j| \leqslant 1$, using the third property we obtain that

$$R_n(H_{N,c_j\text{'s}}(A)) \leqslant R_n(A), \ \forall N, c_j\text{'s}. \tag{16}$$

And since $H(A) = \bigcup_{N,c_j\text{'s}} H_{N,c_j\text{'s}}(A)$, using the argument that the supremum of elements belonging to a union of sets is the supremum over the elements of all the subsets, we get that $R_n(H(A)) \leqslant R_n(A)$. As $R_n(H(A)) \leqslant R_n(A)$ and $R_n(A) \leqslant R_n(H(A))$, we conclude that $R_n(A) = R_n(H(A)) = R_n(\text{absconv(A)})$.

Problem 10: A circle in the plane is a set of the form $C_{c,r} = \{x \in \Re^2 : \|x - c\| \leqslant r\}$ for some $c\Re^2$ and $r \geqslant 0$.
Determine the VC dimension of the class $\mathcal{A} = \{C_{c,r} : c \in \Re^2, r \geqslant 0\}$ of all circles.
What is the VC dimension of the class $\mathcal{A}_\infty = \{C_{c,1} : c \in \Re^2\}$ of all circles of radius 1?

We consider first the general circle in the plane $C_{c,r} = \{x \in \Re^2 : \|x - c\| \leqslant r\}$, where c is the center and r the radius. It is clear that we can shatter at least three points in the plane, by looking at the next graphic we can see the 8 possible subsets of three points.

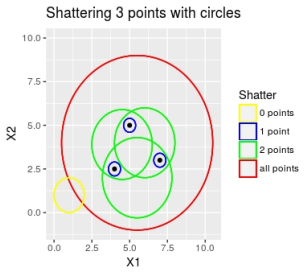

Figure 1: All the different ways to separate the points with circles.

However four points cannot be shattered. It is clear that if at least three of them are in the same line, it is impossible to shatter them (by convexity of the circle one cannot take the two extreme points without the ones in the middle).
Apart from that there are two different cases depending on their convex hull: either their convex hull is formed by connecting three of the points (triangular convex hull with a point inside), or their convex hull is formed by connecting the four points (quadrilateral convex hull).
In the first case, by convexity of the circle, one could not include the three vertices of the triangle and leave the other point out.
In the second case, consider the two diagonals that connect the opposite extremes of the quadrilater. One could not build a circle containing the two points connected by the longest diagonal that does not contain at least one of the other two points. (if the two diagonals were equal then one could not build a circle containing either of the pair of opposite points that does not contain any of the others). So, four points cannot be shattered, hence the VC-dimension is 3.

Regarding to the circles of radius 1 in the plane, $C_{c,1} = \{x \in \Re^2 : \|x - c\| \leqslant 1\}$. We can shatter also up to three points, but not four as we showed before.

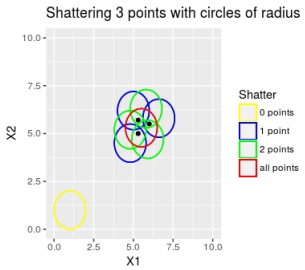

Figure 2: All the different ways to separate the points with circles of radius 1.

Problem 11: A half plane is a set of the form $H_{a,b,c} = \{(x, y) \in \Re^2 : ax + by \geqslant c\}$ for some real numbers a, b, c. Determine the n-shatter coefficient of the classes

$$\mathcal{A}_0 = \{H_{a,b,c} : a, b \in\} \quad \text{and} \quad \mathcal{A} = \{H_{a,b,c} : a, b, c \in \Re\} \tag{17}$$

First, notice that $H_{a,b,c}$ can be constructed as $H_{a,b,c}\{(x, y) \in \Re^2 : \sum_{i=1}^{3} a_i\varphi_i(x, y) \geqslant 0\}$, where $a_1 = a$, $a_2 = b$, $a_3 = c$, $\varphi_1(x, y) = x$, $\varphi_2(x, y) = y$, $\varphi_3(x, y) = 1$. So, the VC dimension of the two classes will be bounded by $\mathcal{A}_0 \leqslant 2$ and $\mathcal{A} \leqslant 3$. It is easy to see that $\mathcal{A}_0 = 2$ and $\mathcal{A} = 3$. With Sauer's lemma we can bound $S_{\mathcal{A}_0}(n) \leqslant (n + 1)^2$ and $S_{\mathcal{A}}(n) \leqslant (n + 1)^3$. But this does not give us the exact shatter coefficient.

For $\mathcal{A}_0$ observe that the points are separated by a line through the origin that cuts the plane in two, and for each separation we can choose between including the points on one side or the other. Let us consider two cases, when all n points can be contained into a half plane and when that is not possible.

On the first case, we would have a line that cuts the plane in one half that contains all the points and the other none. Then, we can order the points by the angle that form each them with the origin and start tilting the line to build all the different separations. When putting the line between x_1 and x_2 we obtain a half plane that contains x_1 and the other the rest of the points. Then, we can tilt more and put it between x_2 and x_3 so we would have a plane that contains x_1 and x_2 and the other the rest of the points. If we keep doing the process tilting the line until it is between each possible pair, we would obtain n lines that lead to different separations. For each separating we can either take the point of one half or the other, so $S_{\mathcal{A}_0}(n) = 2n$.
On the second case, we would obtain at least two less combinations of points (all the points and none of them).

For $\mathcal{A}$, the way to have the points spread so we can do the highest number of combinations is when they form the convex hull of the points. That way, we can separate each of them. Without loss of generality, let us pick one of the points as the first one, and numerate from it clockwise. Then, for each point i we put $n-1$ lines that cut the plane in a way that the first line creates the half plane that contains the point i, the second line creates the half plane that contains i and $i+1$, and so on until the last line that contains all the points but the $i - 1$ (considering that $i - 1 = n$ when $i = 1$). Following this procedure for all points we would obtain $n(n - 1)$ different subsets of points by intersection with the half planes. Apart from this we would have the subset of all the points and the subset without any point. In total $S_{\mathcal{A}}(n) = n(n - 1) + 2$.

Problem 12: Write a program that generates n uniformly distributed points $X_1, \ldots, X_n$ in the d-dimensional cube $[-2^{\frac{1}{d}}, 2^{\frac{1}{d}}]^d$. Assign labels such that $Y_i = 1$ if $X_i \in [-1, 1]^d$ and $Y_i = 0$ otherwise. (Thus, about half of the points have label 1.)
Train two different classifiers, both performing empirical risk minimization as follows.
The first classifier selects the smallest cube of form $[-a, a]^d$ (for some $a \leqslant 0$) that contains all points with label 1 and classifies with 1 inside the cube and with 0 outside.
The second classifier selects the smallest rectangle of form$[a_1, b_1] \times \cdot \times [a_d, b_d]$ (for arbitrary real numbers $a_i \leqslant b_i$, $i = 1, \ldots, d$) that contains all points with label 1 and classifies with 1 inside the rectangle and with 0 outside.
Try a wide range of values of d and n and plot the test error (measured on a large independent test set) for both classifiers. Explain what you see.

The hypercube's classifier risk is clearly lower than the hyperrectangle's classifier. This is a clear example of how overfit can hurt us. Our data classification was done by a hypercube of size 1, so estimating the classification with a hypercube is more accurate than doing it with a hyperrectangle. Observe that the hypercube's class is a subclass of the hyperrectangle's class, but in the second one we have more free parameters, so it will fit the data training data in a more complex way, reducing the volume of the region classified as 1. Therefore leading to more errors when testing the classifier's performance.

Also, it can be seen that increasing the number of points used in the training data set reduces very significantly the risk, in especial with the hyperrectangle's classifier. Observe that for the hypercube's classifier with only 100 the performance is quite good. While with the hyperrectangle we need at least 1000 points for low dimensions (3,5,10) and more than 10000 for higher ones (50,100) in order to have a fairly good classifier. That is because in the hypercube just with one of the components of a point being close enough to 1 we have a strong classifier, while in the hyperrectangle we need that most of the components are of any point to be close to 1 (which is much harder when the dimension increases, being even a totally coin toss classifier when dimension is high and the number of points is small).

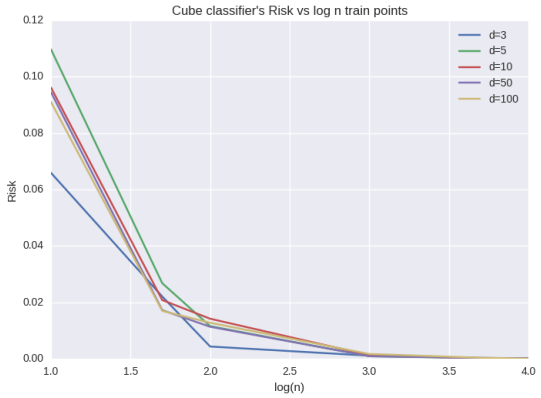

Figure 3: Risk of the hypercube classifier for different dimensions (d=3,5,10,50,100) and different training sample sizes (N=10,50,100,1000,5000,10000).

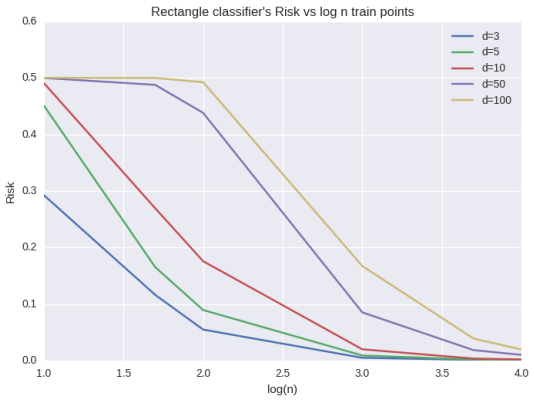

Figure 4: Risk of the hyperrectangle classifier for different dimensions (d=3,5,10,50,100) and different training sample sizes (N=10,50,100,1000,5000,10000).

Appendix

Ex 12:

```
import sklearn as sk
import numpy as np
from numpy import random as rand
from sklearn import neighbors
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import math as mat
```

```
def gen_data(n,d):
X=rand.uniform(low=-2**(1/d),high=2**(1/d),size=(n,d))

Y=[2]*n
for i in range(0,n):
if any(abs(n)>1 for n in X[i])==True:
Y[i]=0
else:
Y[i]=1
return(X,Y)
```

```
def cube_classifier(X,Y):
data=list(zip(X,Y))
data=pd.DataFrame(data)
data.columns=['X','Y']
```

```
data1=data[data['Y']==1]
a=np.amax(list(abs(data1['X'])))
return(a)



def rectangle_classifier(X,Y):
data=np.column_stack([X,Y])
data1=data[data[:,d]==1]

a=np.amin(data1[:,0:d],axis=0)
b=np.amax(data1[:,0:d],axis=0)
return(list(zip(a,b)))


def check_classcube(X_test,Y_test,class_cube):
m=len(X_test)
Y_emp_cube=[]
for i in range(0,m):
if any(abs(n)>class_cube for n in X_test[i])==True:
Y_emp_cube.append(0)
else:
Y_emp_cube.append(1)

check_cube=np.array(Y_test)!=np.array(Y_emp_cube)
wrong_cube=sum(check_cube)
freq_cube=wrong_cube/m
return(freq_cube)

def check_classrect(X_test,Y_test,class_rectangle):
m=len(X_test)
Y_emp_rect=[]
for i in range(0,m):
if any((X_test[i][j]<class_rectangle[j][0] or X_test[i][j]>class_rectangle[j][1])
for j in range(0,d))==True:
Y_emp_rect.append(0)
else:
Y_emp_rect.append(1)

check_rect=np.array(Y_test)!=np.array(Y_emp_rect)
wrong_rect=sum(check_rect)
freq_rect=wrong_rect/m

return(freq_rect)


m=100000
```

```
for d in [3,5,10,50,100]:

s=[d]*6
n=[10,50,100,1000,5000,10000]
lenn=len(n)
check_cube=[0]*lenn
check_rect=[0]*lenn
X_test, Y_test=gen_data(m,d)

for i in range(0,10):

train=list(map(gen_data,n,s))
X_train=[x[0] for x in train]
Y_train=[x[1] for x in train]
class_cube=list(map(cube_classifier,X_train,Y_train))
class_rect=list(map(rectangle_classifier,X_train,Y_train))
aux_cube=list(map(check_classcube,[X_test]*lenn,[Y_test]*lenn,class_cube))
check_cube=list(map(np.add,check_cube,aux_cube))
aux_rect=list(map(check_classrect,[X_test]*lenn,[Y_test]*lenn,class_rect))
check_rect=list(map(np.add,check_rect,aux_rect))

check_cube=list(map(np.divide,check_cube,[1000]*lenn))
check_rect=list(map(np.divide,check_rect,[1000]*lenn))
#plt.subplot(3,2,i)
plt.figure(0)
plt.plot(list(map(mat.log,n,[10]*lenn)), check_cube)
plt.figure(1)
plt.plot(list(map(mat.log,n,[10]*lenn)), check_rect)
#plt.ylim([0.15,0.5])

plt.figure(0)
plt.legend(['d=3','d=5', 'd=10', 'd=50', 'd=100' ],loc='upper right')
plt.xlabel('log(n)')
plt.ylabel('Risk')
plt.title("Cube classifier's Risk vs log n train points")


plt.figure(1)
plt.legend(['d=3','d=5', 'd=10', 'd=50', 'd=100', 'd=1000' ],loc='upper right')
plt.xlabel('log(n)')
plt.ylabel('Risk')
plt.title("Rectangle classifier's Risk vs log n train points")
```