

Exercise 7.1

In this exercise we are interested about Rademacher complexities. It suffices to examine the term

$$\hat{R}_m(\mathcal{F}, S) = E_r \sup_{f \in \mathcal{F}} \left| \frac{2}{m} \sum_i^m r_i f(z_i) \right|. \quad (80)$$

Denote $\text{conv}(\mathcal{F})$ as \mathcal{F}_c and $\text{absconv}(\mathcal{F})$ as F_a .

i) $\hat{R}_m(\mathcal{F}_a, S) \geq \hat{R}_m(\mathcal{F}, S)$, as in \mathcal{F}_a we are allowed to choose $\bar{v} = \bar{e}_1$ (without loss of generality), where e_1 is the standard basis vector (i.e. the first slot is 1 and the rest are 0). That is, we can atleast choose all members from \mathcal{F} by F_a (and by F_c too).

ii)

$$\hat{R}_m(\mathcal{F}_a, S) = E_r \sup_{f \in \mathcal{F}_a} \left| \frac{2}{m} \sum_i^m r_i f(z_i) \right| \quad (81)$$

$$\leq E_r \sup_{f \in \mathcal{F}, \bar{v}} \left| \frac{2}{m} \sum_i^m r_i \sum_j^n v_j f(z_i) \right| \quad (82)$$

$$\leq E_r \sup_{f \in \mathcal{F}, \bar{v}} \sum_j^n v_j \left| \frac{2}{m} \sum_i^m r_i f(z_i) \right| \quad (83)$$

$$\leq \sup_{\bar{v}} \sum_j^n v_j E_r \sup_{f \in \mathcal{F}} \left| \frac{2}{m} \sum_i^m r_i f(z_i) \right| \quad (84)$$

$$\leq \sup_{\bar{v}} \sum_j^n v_j \hat{R}_m(\mathcal{F}, S) \quad (85)$$

$$= 1 \hat{R}_m(\mathcal{F}, S), \quad (86)$$

where we used the definition of functions in F_a and Jensen for abs (convex. by triangle ineq.) to get out $\sum v$. The hint given in the exercise sheet could also be used. Finally, due to definition, $1 = \sum v$.

We have shown that $R_m(\mathcal{F}) = R_m(\mathcal{F}_a)$. The equality to $\text{conv}(\mathcal{F}_c)$ follows from noting that $\mathcal{F}_c \subseteq \mathcal{F}_a$.

Note: an interesting practical corollary of this result is that in the Rademacher complexity sense, it appears that we can take weighted (linear) combinations of models without increasing the complexity cost. Some practically successful methods, such as boosting, can be seen as optimizing (roughly) similar kind of convex hull hypothesis by an incremental, greedy search.

□

Exercise 7.2

a) Note that we are throwing dice and that the throws are independent ($\Rightarrow E[C|A] = E[C]$, for example). It follows that

$$E[E[X|A, B]|A] = E[E[A + B + C|A, B]|A] \quad (87)$$

$$= E[A + B + E[C]|A] \quad (88)$$

$$= A + E[B] + E[C]. \quad (89)$$

and similarly $E[A + B + C|A] = A + E[B] + E[C]$. Just keep in mind which variables have been "fixed" in the given condition and can be thus seen as constants. Numerically, $E[A] = E[B] = E[C] = \frac{1}{6}(1 + 2 + 3 + 4 + 5 + 6) = 3.5$, so the sum is $A + 7$ in both cases.

b) Need to show $E[E[X|A, B]|A] = E[X|A]$. This is sometimes called the *law of total probability for conditional expectations*. Lets follow the hint given in the exercise sheet and denote $Y(a, b) = E[X|A = a, B = b]$. By definition,

$$Y(a, b) = \frac{\sum_x x Pr(X = x, A = a, B = b)}{\sum_x Pr(X = x, A = a, B = b)} \quad (90)$$

and

$$E[Y|A = a] = \frac{\sum_y y Pr(Y = y, A = a)}{\sum_y Pr(Y = y, A = a)} \quad (91)$$

$$= \frac{\sum_y y \sum_b Pr(Y = y, A = a, B = b)}{\sum_y Pr(Y = y, A = a)} \quad (92)$$

$$= \frac{\sum_b \sum_y y Pr(Y = y, A = a, B = b)}{Pr(A = a)} \quad (93)$$

$$= \frac{\sum_b Y(a, b) Pr(A = a, B = b)}{Pr(A = a)} \quad (94)$$

$$= \frac{\sum_b \sum_x x Pr(X = x, A = a, B = b)}{Pr(A = a)} \quad (95)$$

$$= \frac{\sum_x x Pr(X = x, A = a)}{\sum_x Pr(X = x, A = a)} \quad (96)$$

$$= E[X|A = a]. \quad (97)$$

In the derivation we used the definition of $Y(a, b)$ to get an expression for $Y(a, b)Pr(A = a, B = b)$ which we inserted to (94). We also used the knowledge that $\sum_x P(X = x, A) = P(A)$ (i.e. we can safely add a dimension if we sum over it, and we can also cut away such a dimension - just by summing over all the possibilities).

□

Exercise 7.3

a)

The equivalence classes of \sim are the "basic events" of \mathcal{F} . Our requirement for all Y means that each Y can tell the difference between any two basic events, but not between two members of the same event.

Fix some arbitrary Y_1 and Y_2 . Given $w \in \Omega$, let $[w] \in \mathcal{F}$ be its equivalency class w.r.t. \sim . Thus Y_1 and Y_2 are constant over $[w]$, $\forall a$. Let now X be a function (not necessarily \mathcal{F} -measurable), and let C_i be the r.v. $E[X|Y_i], i \in \{1, 2\}$. Now

$$C_i = E[X|Y_i = y_i] \quad (98)$$

$$= \frac{\sum_x xP(X = x, Y_i = y_i)}{\sum_x P(X = x, Y_i = y_i)} \quad (99)$$

$$= \frac{\sum_{w' \in \Omega} X(w')\delta(y_i, Y_i(w'))P(w')}{\sum_{w' \in \Omega} \delta(y_i, Y_i(w'))P(w')} \quad (100)$$

$$= \frac{\sum_{w' \in [w]} X(w')P(w')}{P([w])}, \quad (101)$$

where δ is the Kronecker delta. Hence, C_i does not depend on i and C_1, C_2 are the same r.v.

b)

Note that in the fixed version of the exercise sheet we assume that X_i is a function of Z_0, \dots, Z_i .

For $i \in \{0, \dots, n\}$, and $w \in \Omega$, define

$$\{w\}_i = \{w' \in \Omega | Z_j(w) = Z_j(w'), \forall j \leq i\}, \quad (102)$$

which can be interpreted as the set of cases that are equivalent to w w.r.t. the side-information available up to index i .

Due to assumptions, $\{w' | Z_j(w) = a\}$ is \mathcal{F}_j -measurable for all j . Since $\mathcal{F}_j \subseteq \mathcal{F}_i$ when $j \leq i$, it follows that $\{w\}_i$ is \mathcal{F}_i -measurable for all $w \in \Omega$. Also, since X_i is determined by Z_0, \dots, Z_i , X_i is constant inside $\{w\}_i$.

Define Y as in part (a) and $[w]_i$ as the equivalency class ("basic event") of w w.r.t. \mathcal{F}_i similarly.

Since $[w]_i$ is the smallest \mathcal{F}_i -measurable set containing w , we have $[w]_i \subseteq \{w\}_i$ (think of set $\{w' | Z_i(w') = a\}$).

Further, each $\{w\}_i$ is a disjoint union

$$\{w\}_i = [w^1]_i \cup \dots \cup [w^k]_i, \quad (103)$$

for some w^1, \dots, w^k . Thus, for $w \in \Omega$ let $S(w) = \{w^1, \dots, w^k\}$ so that (103) holds.

Assume now that $E[X_{i+1}|Y] = X_i$, or, for any $w \in \Omega$,

$$X_i(w) = \frac{\sum_x xP(X_{i+1} = x, Y = Y(w))}{\sum_x P(X_{i+1} = x, Y = Y(w))}. \quad (104)$$

Lets start to expand the formula with the side information,

$$E[X_{i+1}|Z_0 = Z_0(w), \dots, Z_i = Z_i(w)]$$

$$= \frac{\sum_x xP(X_{i+1} = x, \bar{Z} = \bar{Z}(w))}{\sum_x P(X_{i+1} = x, \bar{Z} = \bar{Z}(w))} \quad (105)$$

$$= \frac{\sum_x x \sum_{w' \in S_i(w)} P(X_{i+1} = x, Y = Y(w'))}{\sum_x P(X_{i+1} = x, \bar{Z} = \bar{Z}(w))} \quad (106)$$

$$= \frac{\sum_{w' \in S_i(w)} \sum_x xP(X_{i+1} = x, Y = Y(w'))}{\sum_x P(X_{i+1} = x, \bar{Z} = \bar{Z}(w))} \quad (107)$$

Now using the definition of $X_i(w)$ above to replace the $\sum_x xP$ sum,

$$= \frac{\sum_{w' \in S_i(w)} X_i(w) \sum_x P(X_{i+1} = x, Y = Y(w'))}{\sum_x P(X_{i+1} = x, \bar{Z} = \bar{Z}(w))} \quad (108)$$

$$= \frac{\sum_{w' \in S_i(w)} X_i(w) P(Y = Y(w'))}{\sum_x P(X_{i+1} = x, \bar{Z} = \bar{Z}(w))} \quad (109)$$

$$= \frac{\sum_{w' \in S_i(w)} X_i(w) P(Y = Y(w'))}{\sum_{w' \in S_i(w)} P(Y = Y(w'))} \quad (110)$$

$$= X_i(w). \quad (111)$$

The last equation follows from noting that X_i is constant within $\{w\}_i$.

It may be possible to construct a more elementary proof. You can do that as an optional exercise if you like.

□

Exercise 7.4-7.5

In this exercise you were asked to calculate the bound of theorem 3.27 for the car dataset used in exercise 5, when marginalised perceptron was used as the learning algorithm. The bounds were then to be used to select a model.

Unfortunately, the results appear discouraging. With this setting, the bound given by the theorem appears to be monotonically decreasing as the margin size increases, atleast on the range of margins suggested by the exercise sheet. The starting margin ρ_0 (the maximum of the euclidean norms of the training vectors, around 40 on this data) gives the best bound, which is still over one. Clearly, such margin doesn't appear realizable except perhaps on pathological datasets. The marginalised perceptron could still work: its just updating its hypothesis all the time. However, the test set error for the choice of ρ_0 can be close to 0.5 on this data - the guessing accuracy for a label-balanced set. On the other hand, the test set error appears to achieve the minimum (≈ 0.07) with a wished margin $\rho \approx 0.17$, which gives a worse bound from 3.27. This suggests that the way we used the bound in this exercise doesn't appear to be a good way of model selection in this setting³.

Perhaps the lesson of this exercise is that a) theoretical bounds are often not practical b) if you use them, you should make sure you're using them correctly, and c) if you wish to select a model in practice, you should atleast verify that the bounds you're using agree with cross-validation or some other empirically reasonable error estimate.

For completeness, I have included the R code I used.

```
#####  
  
# 7.4-5  
  
library('pixmap'); # for plotting a greyscale image  
load('Lcars.RData'); # contains instances, labels  
  
# the actual learning script  
  
dims<-dim(instances); # (rows,cols,instance#)  
X<-matrix(data=0,nrow=length(labels),ncol=prod(dims[1:2]));  
# turn it to a matrix  
for(i in 1:length(labels)) {  
  X[i,]<-as.vector(instances[,i]);  
}  
  
# sample train,test indexes  
idxsleft<-1:length(labels);  
trainidxs<-sample(idxsleft,floor(2/3*dim(X)[1]),replace=FALSE);
```

³This statement is left vague on purpose, as it not too clear what the problem is: is it in the way we used the bound, the characteristics of the dataset/algorithm combination, some property of the bound itself, or perhaps several of these reasons at the same time, resulting in useless bound behaviour.

```

testidxs<-setdiff(idxsleft,trainidxs);

# split the data and labels to train and test sets
trainX<-X[trainidxs,];
testX<-X[testidxs,];
trainLab<-labels[trainidxs];
testLab<-labels[testidxs];

wstack<-NULL;          # our stored hypotheses
eta<-0.001;           # learning rate coeff.
maxIter<-20;          # don't ever bother with more iterations
B<-1;                 # keep the norm under this

tmp<-sqrt(apply(trainX^2,1,sum)); # ||x_t||_2 for all t
Xnorm<-max(tmp);       # max_t ||x_t||_2
XnormSum<-sqrt(sum(tmp^2));    # sqrt(sum_t(||x_t||_2^2)) for 3.27
exCount<-length(trainLab);    # m, the training example count

margins<-rep(2,15);
margins<-1/cumprod(margins);
margins<-margins*Xnorm;    # generates the decreasing margin sequence

for (marg in margins) {
  cat('For margin ', marg, '...\n');

  # loop the perceptron training until no mistakes are made

  wm<-rep(0,dim(X)[2]); # initial hypothesis

  doIter<-1;totalMistakes<-0;
  while(doIter && doIter<maxIter) {
    mistakesNow<-0;
    cat(' Starting iter ', doIter, '...\n');
    for(i in 1:NROW(trainX)) {
      pred<-trainLab[i]*sum(wm*trainX[i,]);
      if(pred<=marg) { # update if margin wasn't met
        wm<-wm + trainLab[i]*eta*trainX[i,];
        mistakesNow<-mistakesNow+1;
        wnorm=sqrt(sum(wm^2));
        if(wnorm>B) {
          wm<-wm/wnorm;
        }
        # visualize the current model
        if(runif(1)<0.01) {
          plot(pixmapGrey(matrix(data=wm,nrow=dims[1],ncol=dims[2])));
          Sys.sleep(0.01);
        }
      }
    }
  }
}

```

```

totalMistakes<-totalMistakes+mistakesNow;
normNow<-sqrt(sum(wm^2));
cat(' norm', normNow, ', made', mistakesNow, ' mistakes now, ',
    totalMistakes, 'total.\n');
doIter<-(mistakesNow>0)*(doIter+1);
}
# append current hypothesis
wstack<-rbind(wstack,matrix(data=wm,nrow=1));

# show the final model
plot(pixmapGrey(matrix(data=wm,nrow=dims[1],ncol=dims[2])));

# calculate final training and test set errors
trainErr<-sum(as.numeric(trainX%*%wm>=0)*2-1 != trainLab)
    /length(trainLab) ;
testErr<-sum(as.numeric(testX%*%wm>=0)*2-1 != testLab)
    /length(testLab) ;

cat(' train: ', trainErr, 'test: ', testErr,'\n');

# what does the test set bound say (function from prev. exer.)?
delta<-0.05;
tSB<-testSetBound(testErr, length(testLab), delta);
cat(' TSB claims P( true_error >= ', tSB, ') < ',delta,'\n');

# calculate theorem 3.27 bound
mu <- 2 * marg;
delta <- 0.01/length(margins);

hingeLosses<-apply(mu-trainLab*(trainX%*%wm),1,max,0);
bound<- ( 1/(mu*exCount) * sum(hingeLosses)
    + (4*B)/(mu*exCount) * XnormSum
    + 3 * sqrt(1/(2*exCount) * log(4/delta)) );
cat(' theorem 3.27 bound says ', bound, '\n');

a<-readline('Next... ?');
}

#####

```

□