

Exercise 3.1

i) Again, lets continue exploiting Maple to handle tedious differentiations.

Lemma 2.18 gives the formula for calculating cL, so just get to it.

First, specify Hellinger loss,

```
> L := 1/2*(sqrt(1-y) - sqrt(1-x))^2 + (sqrt(1+y) - sqrt(1+x))^2;
```

$$L := 1/2 (\sqrt{1-y} - \sqrt{1-x})^2 + 1/2 (\sqrt{1+y} - \sqrt{1+x})^2$$

We are interested only in $y \in \{-1, 1\}$,

```
> Lminus:=eval(L,y=-1);
```

$$Lminus := 1/2 (\sqrt{2} - \sqrt{1-x})^2 + 1/2 + 1/2 x$$

```
> Lplus:=eval(L,y=1);
```

$$Lplus := 1/2 - 1/2 x + 1/2 (\sqrt{2} - \sqrt{1+x})^2$$

Need resp. 1st and 2nd order derivatives,

```
> Lminsd1:=diff(Lminus,x);
```

$$Lminsd1 := 1/2 \frac{\sqrt{2}-\sqrt{1-x}}{\sqrt{1-x}} + 1/2$$

```
> Lminsd2:=diff(Lminsd1,x);
```

$$Lminsd2 := 1/4 (1-x)^{-1} + 1/4 \frac{\sqrt{2}-\sqrt{1-x}}{(1-x)^{3/2}}$$

```
> Lplusd1:=diff(Lplus,x);
```

$$Lplusd1 := -1/2 - 1/2 \frac{\sqrt{2}-\sqrt{1+x}}{\sqrt{1+x}}$$

```
> Lplusd2:=diff(Lplusd1,x);
```

$$Lplusd2 := 1/4 (1+x)^{-1} + 1/4 \frac{\sqrt{2}-\sqrt{1+x}}{(1+x)^{3/2}}$$

Construct the ratio of the lemma,

```
> upper:= Lminsd1*(Lplusd1)^2 - Lplusd1*(Lminsd1)^2;
```

$$\begin{aligned} upper := & \left(1/2 \frac{\sqrt{2}-\sqrt{1-x}}{\sqrt{1-x}} + 1/2\right) \left(-1/2 - 1/2 \frac{\sqrt{2}-\sqrt{1+x}}{\sqrt{1+x}}\right)^2 \\ & - \left(-1/2 - 1/2 \frac{\sqrt{2}-\sqrt{1+x}}{\sqrt{1+x}}\right) \left(1/2 \frac{\sqrt{2}-\sqrt{1-x}}{\sqrt{1-x}} + 1/2\right)^2 \end{aligned}$$

```
> lower:= Lminsd1*Lplusd2 - Lplusd1*Lminsd2;
```

$$\begin{aligned} lower := & \left(1/2 \frac{\sqrt{2}-\sqrt{1-x}}{\sqrt{1-x}} + 1/2\right) \left(1/4 (1+x)^{-1} + 1/4 \frac{\sqrt{2}-\sqrt{1+x}}{(1+x)^{3/2}}\right) \\ & - \left(-1/2 - 1/2 \frac{\sqrt{2}-\sqrt{1+x}}{\sqrt{1+x}}\right) \left(1/4 (1-x)^{-1} + 1/4 \frac{\sqrt{2}-\sqrt{1-x}}{(1-x)^{3/2}}\right) \end{aligned}$$

```
> ratio := simplify(upper/lower);
```

$$ratio := 1/2 \sqrt{1+x} \sqrt{1-x} (\sqrt{1-x} + \sqrt{1+x}) \sqrt{2}$$

```
> maximize(ratio,x=0...1);
```

$$\sqrt{2}$$

The maximization can of course be done by hand, by setting the first derivative to zero, etc.

ii) This can be handled similarly to ex. 2.17. of the lecture notes. We are interested in cases $y \in \{-1, 1\}$. Simplify the Hellinger loss for these two cases, leading to

$$L(-1, x) = 2 - \sqrt{2}\sqrt{1-x}$$

$$L(1, x) = 2 - \sqrt{2}\sqrt{1+x}.$$

According to the prediction rule of the Aggregating Algorithm (AA), we need

$$L(y, \tilde{y}) \leq \Delta(y),$$

where Δ is as specified in the lecture notes. Plugging in the simplified losses,

$$2 - \sqrt{2}\sqrt{1-x} \leq \Delta(-1)$$

$$2 - \sqrt{2}\sqrt{1+x} \leq \Delta(1)$$

and solving for x , we get

$$-\frac{(2 - \Delta(1))^2}{2} + 1 \geq x \geq \frac{(2 - \Delta(-1))^2}{2} - 1$$

We can suppose that a reasonable choice for prediction lies in the middle of these two bounds (i.e. $[a, b] \rightarrow a + \frac{1}{2}(b - a)$), leading to the selection of

$$\tilde{y} = \Delta(1) - 1/4\Delta(1)^2 - \Delta(-1) + 1/4\Delta(-1)^2.$$

□

Exercise 3.2

Following the hint, we can proceed by first transforming the given constraint to an easier form and then showing that the related function is convex. As we know that the constraint holds for the range endpoints $\{-1, 1\}$, the convexity of the function will show us that the resp. constraint must also hold for every $y \in [-1, 1]$ (intuition e.g. through visualizing the idea for convex $f(x) = x^2$). Now that we have this plan, tweak the constraint,

$$\begin{aligned} (y - \hat{y})^2 &\leq \Delta(y) \\ \eta(y - \hat{y})^2 - \eta\Delta(y) &\leq 0 \\ f(y) := \exp(\eta(y - \hat{y})^2 - \eta\Delta(y)) &\leq \exp(0) = 1. \end{aligned}$$

Separating the $\exp()$ for $f(y)$, we get

$$f(y) = \exp(\eta(y - \hat{y})^2) \exp(-\eta\Delta(y)),$$

and plugging in the definition of

$$\Delta(y) = c \ln\left(\frac{W_t}{W_{t+1}}(y)\right) = -c \ln\left(\sum_{i=1}^N \frac{W_{t,i}}{W_t} \exp(-\eta(y - x_{t,i})^2)\right),$$

we can arrive at

$$f(y) = \sum_{i=1}^N \frac{W_{t,i}}{W_t} \exp(\eta(y - \hat{y})^2 - \eta(y - x_{t,i})^2).$$

Since $\frac{W_{t,i}}{W_t}$ is positive, it suffices to examine the $\exp()$ part on the right. Denote it by $g(y)$. Now

$$g'(y) = (2\eta(y - \hat{y}) - 2\eta(y - x_i)) \exp(\eta(y - \hat{y})^2 - \eta(y - x_i)^2)$$

$$g''(y) = (2\eta(y - \hat{y}) - 2\eta(y - x_i))^2 \exp(\eta(y - \hat{y})^2 - \eta(y - x_i)^2) \geq 0,$$

clearly. From the form of $f(y)$ we can conclude that $f''(y) \geq 0$ as well. According to a well-known result from analysis, $f(y)$ is then convex (providing it is continuous and twice differentiable, as our f is).

Exercise 3.3

Use the same potential technique as in the lecture notes.

Write out the one-step potential difference as

$$P_t - P_{t+1} = \frac{1}{2}\|u - w\|_2^2 - \frac{1}{2}\|u - w'\|_2^2 \quad (33)$$

$$= (w' - w)(u - w) - \frac{1}{2}\|w - w'\|_2^2 \quad (34)$$

$$= (w_{t+1} - w_t)(u - w_t) - \frac{1}{2}\|w_t - w_{t+1}\|_2^2, \quad (35)$$

where on the last line we just substituted $w = w_t$ and $w' = w_{t+1}$.

Supposing a mistake made by the algorithm and applying the perceptron update rule, we know that $w_{t+1} - w_t = \eta y_t x_t$ and $w_t - w_{t+1} = -\eta y_t x_t$. Plugging these in, we get

$$P_t - P_{t+1} = (\eta y_t x_t)(u - w_t) - \frac{1}{2}\|-\eta y_t x_t\|_2^2 \quad (36)$$

$$= (\eta y_t x_t)(u - w_t) - \frac{1}{2}\eta^2\|x_t\|_2^2 \quad (37)$$

$$= \eta y_t x_t u - \eta y_t x_t w_t - \frac{1}{2}\eta^2\|x_t\|_2^2 \quad (38)$$

$$\geq \eta y_t x_t u - \frac{1}{2}\eta^2 X^2 \quad (39)$$

$$\geq \eta - \frac{1}{2}\eta^2 X^2 \quad (40)$$

$$= \eta\left(1 - \frac{\eta X^2}{2}\right), \quad (41)$$

where we applied the knowledge that w_t made a mistake ($-\eta y_t x_t w_t$ is positive), that u had a margin ≥ 1 , and that the norm of each x was bounded by X .

Summing the potential differences over the trials gives

$$\sum_{t=1}^T (P_t - P_{t+1}) \geq \eta\left(1 - \frac{1}{2}\eta X^2\right) \sum_{t=1}^T \sigma_t \quad (42)$$

$$P_1 - P_{T+1} \geq \eta\left(1 - \frac{1}{2}\eta X^2\right) \sum_{t=1}^T \sigma_t \quad (43)$$

$$P_1 \geq \eta\left(1 - \frac{1}{2}\eta X^2\right) \sum_{t=1}^T \sigma_t \quad (44)$$

$$\frac{1}{2}\|u - w_{init}\|_2^2 \geq \eta\left(1 - \frac{1}{2}\eta X^2\right) \sum_{t=1}^T \sigma_t \quad (45)$$

where we knew that P_{T+1} is positive and the value of the starting potential P_1 . From this and selecting $\eta = \frac{1}{X^2}$, we can get the mistake bound

$$\sum_{t=1}^T \sigma \leq \|u - w_{init}\|_2^2 X^2.$$

Exercise 3.4

Although this exercise could be solved by optimization techniques (i.e. minimize a target function w.r.t. a constraint), it is rather tedious. An easier way for now is to visualize the feasible region, the old w_t , and the new x we made a mistake on. As we made a mistake, old w_t is not in the feasible region $\{w|y_t w \cdot x > 1\}$, whereas x is. Now clearly the point w' that both fulfills the margin criteria and is closest to the old w_t , relies on the hyperplane $\{w|w \cdot x = 1\}$. (You can think of w' as a sum of two vectors, perpendicular to each other and the other collinear to the hyperplane. Set the length of the collinear vector to 0 to minimize the total norm. In 2D, think of what kind of triangle gives the smallest hypotenuse length).

Thus, the solution is just the *orthogonal projection* of w_t to the hyperplane $\{w|w \cdot x = 1\}$. From the orthogonal projection formula we get the update rule

$$w_{t+1} = w_t + \left(\frac{1 - y_t x \cdot w_t}{\|w_t\|_2^2}\right)x.$$

As the Euclidean distance to the hyperplane is $\|w_{t+1} - w_t\|_2$, the distance is also the norm of the vector (perpendicular to the hyperplane!) we are incrementing our hypothesis w_t with. To put it another way, we are adding a scaled version of the hyperplane normal to our previous hypothesis.

Exercise 3.5

Here you were supposed to learn how to do basic things on high-level languages such as R or Matlab. In empirical machine learning, data-analysis and other similar areas, it is often the best to start with some interactive, high-level language to experiment with prototype algorithms². Possibility to use matrix operations, plots and all kinds of helpful add-on toolboxes can make exploratory work much smoother.

After implementing the requirements of this particular assignment, we can see that

- The algorithm always converges to zero training error as there is no noise in the data/labels.
- Repeating the experiment shows the high variance in the number of iterations required, depending on the data we happened to create (especially its margin).
- With 48 irrelevant attributes, the test error varies roughly around 10% and 30%, compared to the typical rate (around 2%) of the basic two-dimensional case. Thus, the perceptron algorithm is clearly misled by the extra attributes.

²Later on, if your algorithm requires e.g. explicit loops, it may be fruitful to either implement the loops in e.g. C, but this probably only after you already have fixed your specification.

This time I have included an implementation on R. Troubles with availability of Matlab (and licenses for its different toolboxes) should probably make R (or Octave) your first choice, unless you specifically need some features provided only by Matlab. Next time, I'll present a Matlab solution for comparison.

```
#####

EXC<-200;
DIM<-2;

# Create random data from [-1,1]^DIM
X<-matrix(data=runif(EXC*DIM),nrow=EXC,ncol=DIM)*2-1;

# Make the target concept classifier (just a vector)
w<-rep(0,DIM);
w[c(1,2)]<-0.5;

# Label (classify) the data
labels<-as.numeric(X%*w>=0)*2-1;

# split the data and labels to train and test sets
trainX<-X[1:(EXC/2),];
testX<-X[(EXC/2+1):NROW(X),];
trainLab<-labels[1:(EXC/2)];
testLab<-labels[(EXC/2+1):NROW(X)];

wm<-rep(0,DIM); # initial hypothesis
alpha<-1;      # learning rate coeff.

# loop the perceptron training until no mistakes are made
doIter<-TRUE;
while(doIter) {
  didMistake<-FALSE;
  cat('Start iter...\n');
  for(i in 1:NROW(trainX)) {
    pred<-as.numeric(sum(wm*trainX[i,])>=0)*2-1;
    if(pred!=trainLab[i]) { # update on mistake
      wm<-wm + trainLab[i] * alpha*trainX[i,];
      didMistake<-TRUE;

      # visualize the current model
      wmt<-wm/sqrt(sum(wm^2));
      cat(' ', i, wm, ' angle ', acos( (wmt%*%w)/sqrt(sum(w^2))) ,'\n');
      plot.default(c(0,wmt[1]),c(0,wmt[2]),type="l",col="red",
                   xlim=c(-1,1),ylim=c(-1,1))
      par(new=TRUE);
      plot.default(c(0,w[1]),c(0,w[2]),type="l",col="blue",
                   xlim=c(-1,1),ylim=c(-1,1))
      points(trainX,col=palette()[trainLab+2]); # use 2 colors
    }
  }
  if(didMistake) doIter=TRUE;
  else doIter=FALSE;
}
```

```

        Sys.sleep(0.5);
    }
}
doIter<-didMistake;
}

# show the final model
wmt<-wm/sqrt(sum(wm^2));
plot.default(c(0,wmt[1]),c(0,wmt[2]),type="l",col="red",
             xlim=c(-1,1),ylim=c(-1,1))
par(new=TRUE);
plot.default(c(0,w[1]),c(0,w[2]),type="l",col="blue",
             xlim=c(-1,1),ylim=c(-1,1))
points(trainX,col=palette()[trainLab+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');
cat(' final angle ', acos( (wmt%*%w)/sqrt(sum(w^2))) ,'\n');

par(new=FALSE);

#####

```

Most of the mess of the code was related to data handling and diagnostic output and plots. For comparison, if you were doing *LMS linear regression* (a standard method for numerical prediction in statistics), you could have learned the model with just one line of code (using matrix inversion).

rant: Considering domains like visual learning, bioinformatics or text classification, 50 dimensions is small beans. Next time we will be having a Winnow-exercise, and to that, I'll write a more hearty rant regarding the nature of irrelevant attributes. Stay tuned!