Introduction to machine learning Exercise set 2 report

Elias Annila 014328901

Pen and paper:

1.

Expected costs for predicting:

+ : $0\alpha+10(1-\alpha)=10(1-\alpha)$

- : $3\alpha+0(1-\alpha)=3\alpha$

? : $1\alpha+2(1-\alpha)=\alpha-2\alpha+2=2-\alpha$

Choose + wehn $10(1-\alpha)<3\alpha \wedge 10(1-\alpha)<2-\alpha$

$$\begin{aligned} 10(1-\alpha)&<3\alpha \\ 10-10\alpha&<3\alpha \\ 10&<13\alpha \\ \alpha&>\frac{10}{13} \end{aligned}$$ and $$\begin{aligned} 10(1-\alpha)&<2-\alpha \\ 10-10\alpha&<2-\alpha \\ 10-9\alpha&<2 \\ -9\alpha&<-8 \\ \alpha&>\frac{8}{9} \end{aligned}$$ because $\frac{8}{9}>\frac{10}{13}$ choose + when $\alpha>\frac{8}{9}$

Choose – when $3\alpha<10(1-\alpha) \wedge 3\alpha<2-\alpha$

as above $$\begin{aligned} 3\alpha&<10(1-\alpha) \\ \alpha&<\frac{10}{13} \end{aligned}$$ and $$\begin{aligned} 3\alpha&<2-\alpha \\ 4\alpha&<2 \\ \alpha&<\frac{1}{2} \end{aligned}$$ because $\frac{10}{13}>\frac{1}{2}$ choose – when $\alpha<\frac{1}{2}$

Choose ? when $2-\alpha<10(1-\alpha) \wedge 2-\alpha<3\alpha$

as above: $$\begin{aligned} 2-\alpha&<3\alpha \\ \alpha&>\frac{1}{2} \end{aligned}$$ and $$\begin{aligned} 2-\alpha&<10(1-\alpha) \\ \alpha&<\frac{8}{9} \end{aligned}$$ because $\frac{8}{9}>\frac{1}{2}$ choose ? when $\frac{1}{2}<\alpha<\frac{8}{9}$

Result: $\alpha<\frac{1}{2} \rightarrow$ -, $\frac{1}{2}<\alpha<\frac{8}{9} \rightarrow$?, $\alpha>\frac{8}{9} \rightarrow$ +. If $\frac{1}{2}=\alpha$ it doesn't matter if we choose – or ?

and if $\frac{8}{9}=\alpha$ it doesn't matter if we choose ? or +.

2.

True probabilities P: $P(y=1)=a$, $P(y=0)=1-a$ Cost function *C*: $C(y=1)=(1-b)^2$, $C(y=0)=b^2$ Expected cost:

$$E_C(b)=a(1-b)^2+(1-a)b^2$$
$$E_C(b)=a(1-2b+b^2)+b^2-ab^2$$
$$E_C(b)=a-2ab+ab^2+b^2-ab^2$$
$$E_C(b)=b^2-2ab+a$$

find where cost rises $E_C'(b)>0$, $2b-2a>0$, $b-a>0$, $b>a$ gets lower: $E_C'(b)<0$, $a>b$ zero points of E_C': $E_C'(b)=0$, $a=b$ This means that while b<a the expected cost gets lower and while b> a expected cost rises. As only zero point of E_C' is b=a we know that expected cost is lowest at b=a so our cost function C is proper.

3.

Considering that if a point x_i is outside rectangle $y=n$ there is 0 probability that $x_i \in y=n$. Because of that and the fact that posterior probabilities need to sum up to 1 and given marginal probabilities are uniformly distributed, we can give probabilities for x_1, x_2 as follows

$$P(x_1 \in y=1 | (x_1 \notin y=0 \wedge x_1 \notin y=2))=1$$
$$P(x_5 \in y=2 | (x_5 \notin y=0 \wedge x_5 \notin y=1))=1$$

as for x_2 we know nothing about it so we must keep to the marginal probabilities $P(x_2 \in y=0)=0.2$, $P(x_2 \in y=1)=0.7$, $P(x_2 \in y=2)=0.1$ for remaining points $x_{3,} x_4$ we can say that

$$P(x_3 \in y=0)=0$$
$$P(x_4 \in y=1)=0$$

so we have the following probabilities to find:

$$P(x_3 \in y=1 | x_3 \notin y=0)=\frac{P(x_3 \notin y=0 | x_3 \in y=1) P(x_3 \in y=1)}{P(x_3 \notin y=0)}=\frac{1 \cdot 0.7}{1-0.2}=\frac{7}{8}$$

$$P(x_3 \in y=2 | x_3 \notin y=0)=\frac{P(x_3 \notin y=0 | x_3 \in y=2) P(x_3 \in y=2)}{P(x_3 \notin y=0)}=\frac{1 \cdot 0.1}{1-0.2}=\frac{1}{8}$$

the results add up to 1 as expected.

$$P(x_4 \in y=0 | x_4 \notin y=1)=\frac{P(x_4 \notin y=1 | x_4 \in y=0) P(x_4 \in y=0)}{P(x_4 \notin y=1)}=\frac{1 \cdot 0.2}{1-0.7}=\frac{2}{3}$$

$$P(x_4 \in y=2 | x_4 \notin y=1)=\frac{P(x_4 \notin y=1 | x_4 \in y=2) P(x_4 \in y=2)}{P(x_4 \notin y=1)}=\frac{1 \cdot 0.1}{1-0.7}=\frac{1}{3}$$

the results add up to 1 as expected.

Results:

	P(Y=0)	P(Y=1)	P(Y=2)
x_1	0	1	0
x_2	1/ 5	7/10	1/10
x_3	0	7/8	1/8
x_4	2/3	0	1/3
x_5	0	0	1

Programming:

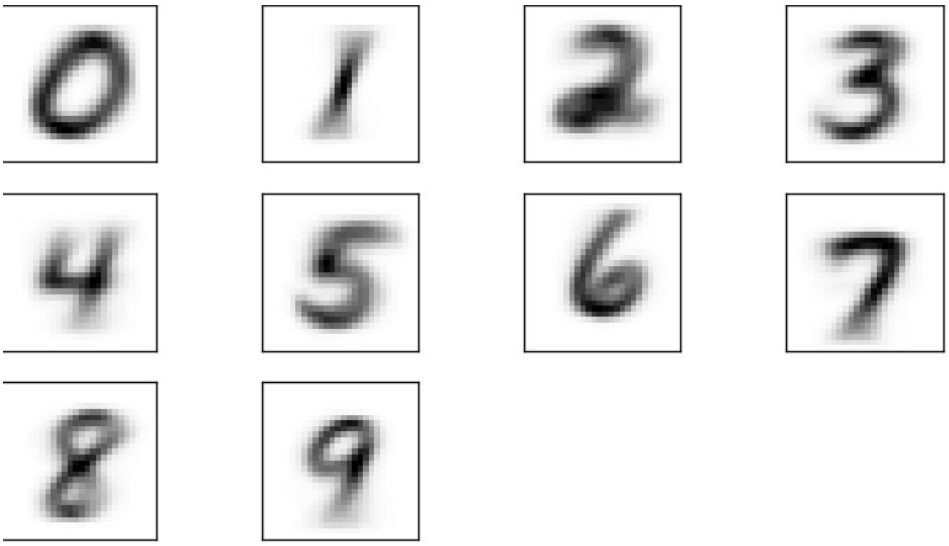

Image 1: Average digits

As seen above average digits look pretty much as expected. They are what a typical digit would look like and it seems clear that any funkier instance of a digit is not going to be very close to any average digit. It is interesting to note how areas where there is a change in the direction of stroke when writing such as the sharp corner in five, the digits seem to have been more similar (darker and sharper) whereas towards the ends of the strokes there has been more variation in images (fuzzier)

		Predicted									
	228	0	1	2	0	4	6	2	2	0
	0	281	1	0	0	3	0	0	1	0
Actual	2	19	192	5	5	2	2	2	11	1
	1	9	4	193	2	20	0	5	11	8
	1	4	1	0	202	0	5	2	0	40
	9	16	1	14	11	136	2	6	1	11
	5	12	15	0	9	8	198	0	0	0
	1	14	1	0	15	0	0	232	1	11
	3	13	9	25	4	19	1	1	151	14
	3	4	6	2	38	2	1	14	2	179

Table 1: Confusion matrix of prototype classifier. Error rate = 0.203

As seen from above confusion matrix the prototype classifier is doing pretty well in general when the digits are well formed with error rate of 0.203. However in cases were digits are not close to the average there exists some confusion especially between fours and nines, which seems reasonable as the general shape is similar and the separating factor is only missing or having a line in the upper part of the digit. Also fives were in general difficult to classify for the prototype based classifier. They got confused with eights and threes. This also seemed to be due to the three digits having similar parts. Ones got confused with several digits. Which is probably due to the very simple shape of one, which lead to the average image being very fuzzy in every other area except in the center of

the image. As the center is also dark in several other prototype digits that may be the reason why ones got confused so much despite having pretty unique shape. Easiest digits seemed to be zero and seven. This is probably due to the same fact that most digits have dark areas in the middle of the image, but seven and zero do not.

		Predicted									
	243	0	0	1	0	0	1	0	0	0
	0	284	1	0	1	0	0	0	0	0
Actual	1	10	215	2	0	0	3	4	5	1
	1	3	3	226	1	9	1	4	3	2
	0	2	0	0	239	1	0	0	0	13
	3	3	0	2	0	184	6	1	3	5
	4	2	0	0	1	2	238	0	0	0
	0	4	1	0	4	0	0	259	0	7
	4	8	5	7	4	7	2	3	192	8
	2	2	2	3	11	1	1	8	0	221

Table 2: Confusion matrix of kNN, k = 3 classifier. Error rate = 0.080

		Predicted									
	241	0	0	2	0	1	1	0	0	0
	0	283	0	0	1	1	0	1	0	0
Actual	2	6	216	5	0	1	1	5	5	0
	0	0	3	221	2	11	2	4	5	5
	0	3	0	0	231	0	0	2	0	19
	5	3	0	6	0	183	5	1	1	3
	3	1	0	0	1	1	240	0	0	1
	0	3	2	0	6	0	0	257	0	7
	4	5	9	10	1	8	0	2	196	5
	2	2	0	2	12	1	2	11	0	219

Table 3: Confusion matrix of kNN, k=1 classifier. Error rate = 0.085

The kNN classifier did considerably better than prototype classifier with error rate of 0.080 (k=3). It is easy to see why kNN is better able to deal with funkier handwriting as there is likely to be a digit in the training data which has been made with similar handwriting and thus the general shape of digits doesn't matter. Especially as the training data and test data are two halves of the same data it is likely that training data and test data both include signatures of same people.

Still the kNN method had some difficulties in recognizing fours and nines as well as threes, fives and eights. This is probably due to the same reasons as with the prototype classifier, the confused images have similar distributions of dark and light areas. Further improvement on separating these digits could perhaps made by trying to detect defining features of each digit and comparing these, rather than trying to compare all the pixels.

Differences between kNN k=1 and k=3 were not large, but k=3 did a little better. This was expected as for larger than k=1 values the program only uses majority of nearest neighbors if the majority was clear. In case of ties the class of nearest neighbor (k=1) was chosen. With k=3 this meant that result could only be different from k= 1 if both second and third nearest neighbors were of the same class and in disagreement with the nearest neighbor. Better results were not obtained by further increasing k.