Pattern Recognition Assignment 1

- Daniel Tell F120181
- Linus Oleander F120180

Model usage

The model can be used to classify unknown letters.

Data description

Each of the 26 letters of the English alphabet were included in 20 different fonts. The letters were then randomly distorted in order to generate 20,000 different cases. 16 numerical attributes were generated from this data and scaled from 0 to 15.

1. lettr - The class
2. x-box - The position along the x-axis of the centre of the smallest box that contains all activated pixels.
3. y-box - The position of the aforementioned box, along the y-axis
4. width - The width of the box
5. height - Height of box
6. onpix - The total number of activated pixels
7. x-bar - Mean position, along x-axis and measured from the center, of activated pixels
8. y-bar - Mean position along y-axis
9. x2-bar - Variance of pixel positions along x-axis
10. y2-bar - Variance of pixel positions along y-axis
11. xybar - Mean value of x and y values for each pixel, as measured from the centre
12. x2ybr - Mean product of x^2 * y
13. xy2br - Mean product of x * y^2
14. x-edge - The mean number of edges - deactivated pixels or image boundary followed by an activated pixel - when searching from left to right
15. xegvy - The sum of the y-values for all edges
16. y-edge - The mean number of edges when searching from bottom to top
17. yegvx - The sum of the x-values for all edges

Class distribution

- [letter] - [amount] - [percent of total]
- A - 789 - 3.95
- B - 766 - 3.83
- C - 736 - 3.68
- D - 805 - 4.03
- E - 768 - 3.84
- F - 775 - 3.88
- G - 773 - 3.87
- H - 734 - 3.67
- I - 755 - 3.78
- J - 747 - 3.74
- K - 739 - 3.70
- L - 761 - 3.81
- M - 792 - 3.96
- N - 783 - 3.91
- O - 753 - 3.77
- P - 803 - 4.01
- Q - 783 - 3.91
- R - 758 - 3.79
- S - 748 - 3.74
- T - 796 - 3.98
- U - 813 - 4.06
- V - 764 - 3.82
- W - 752 - 3.76
- X - 787 - 3.94
- Y - 786 - 3.93
- Z - 734 - 3.67

`20000` letters in total. Most common letter is `D` and least common `Z`.

Data selection

We picked 75% of the data at random for building our model and the rest for testing the model itself. This has been done for all six implementations.

Methods

knn

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
A	186	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	177	0	1	1	0	0	5	0	0	0	0	1	0	0	0	0	5	2	0	0	4	0
C	0	0	175	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
D	0	0	0	188	0	0	1	3	0	0	0	0	0	1	2	1	0	0	1	1	0	0	0
E	0	0	1	0	194	0	5	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0
F	0	0	0	1	0	177	0	1	1	0	0	0	0	0	0	6	0	1	0	2	0	1	0
G	0	1	0	1	2	0	198	0	0	0	1	0	0	0	3	0	0	0	0	0	0	0	1
H	0	1	0	6	0	0	0	166	0	1	9	0	0	0	3	1	0	5	0	0	0	0	0
I	0	0	0	0	0	1	0	0	187	4	0	0	0	0	0	0	0	0	0	0	0	0	0
J	0	1	0	0	0	2	0	0	2	185	0	0	0	0	0	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	4	0	0	155	0	0	0	0	0	0	3	0	0	0	0	0
L	0	0	1	0	1	0	0	0	0	0	2	165	0	0	0	0	0	1	0	0	0	0	0
M	0	0	0	0	0	0	1	1	0	0	0	0	208	2	0	0	0	0	0	0	0	3	1
N	0	0	0	1	0	0	0	1	0	0	0	0	0	189	1	0	0	3	0	0	0	0	0
O	0	0	0	1	0	0	0	0	0	0	0	0	0	0	186	0	2	0	0	0	0	0	1
P	0	1	0	0	0	7	0	1	0	0	0	0	0	0	0	186	0	0	0	1	0	0	0
Q	0	0	0	1	0	0	3	0	0	0	0	0	0	0	8	2	184	1	1	0	0	0	0
R	0	7	0	1	0	0	0	0	0	0	1	0	0	2	0	0	0	171	0	0	0	1	0
S	0	1	0	1	2	0	0	1	0	0	0	0	0	0	0	0	0	1	161	0	0	0	0
T	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	196	0	0	0
U	0	0	0	0	0	0	0	2	0	0	1	0	0	0	0	0	0	0	0	0	226	0	0
V	0	2	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	168	1
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	192
X	0	0	0	1	1	0	0	1	0	0	7	0	0	0	0	0	0	0	0	1	0	0	0
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0
Z	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0

Hardest letter to classify H (86.5%)
Easiest letter to classify W (99.5%)

Error rate 4.4%

Complexity params:

- `k=1`

How where the complexity params chosen?

The k nearest neighbour algorithm looks at nearest data points from the training set in order to determine the class of new cases. While running the function with different values, we noticed a negative correlation between the k value and the hit rate. This could be because of the *curse of dimensionality* . We have 16 different numerical attributes, so the data space might be so sparse that distant data points have to be considered when trying to predict with high k values.

Benchmark results

- `k=1`

Time to run and use: `2.467` seconds.
Error rate: 4.4%

- `k=2`

Time to run and use: `2.313` seconds.
Error rate: 4.8%

- k=5

Time to run and use: 2.218 seconds.
Error rate: 5.5%

- k=60

Time to run and use: 2.616 seconds.
Error rate: 13.7%

Conclusion

A lower k value, in our case k=1, results in a better and more precis model. The execution time does not relay on the value k, which can be seen in the resuls above.

multinom

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
A	163	0	0	0	0	0	0	1	0	2	4	0	1	0	4	0	0	3	8	1	2	0	0
B	0	138	0	4	2	5	1	4	3	0	1	0	0	0	5	3	0	10	6	0	0	3	0
C	0	0	133	1	11	0	8	5	0	0	12	3	1	0	12	0	0	0	0	4	7	0	1
D	0	9	0	162	1	0	0	6	1	1	0	0	1	3	5	2	0	3	0	3	1	0	0
E	0	6	0	1	136	2	10	0	1	0	0	6	0	0	0	0	1	3	7	0	0	0	0
F	0	4	0	3	3	148	3	2	3	0	0	0	0	0	0	4	0	2	8	15	0	0	1
G	6	0	21	2	0	0	105	1	0	0	5	12	0	0	3	1	21	4	7	0	0	6	1
H	0	3	1	8	0	2	3	79	0	0	6	0	1	12	20	4	4	14	0	0	5	4	0
I	0	2	0	3	0	0	0	0	158	6	0	3	0	0	3	2	0	0	2	0	0	0	0
J	0	0	0	3	0	4	0	3	10	139	0	0	0	1	1	2	1	0	6	0	0	0	0
K	1	2	1	1	5	0	1	3	0	0	113	1	2	3	0	0	0	15	0	0	11	1	0
L	0	1	1	3	3	0	8	0	0	0	2	179	0	0	0	0	2	0	2	0	2	0	0
M	2	1	0	3	0	0	0	2	0	0	0	0	175	5	2	0	0	3	0	0	2	0	5
N	0	0	0	5	0	1	0	8	0	0	4	0	2	185	2	0	0	0	0	1	2	2	4
O	3	1	2	7	0	0	3	19	0	1	0	0	1	0	124	0	0	3	1	0	1	1	16
P	0	1	0	3	0	18	7	1	1	0	1	0	0	1	3	161	0	0	0	0	0	6	0
Q	3	6	0	3	6	0	9	1	0	2	2	4	0	0	9	4	135	0	3	0	0	0	1
R	0	11	1	1	2	2	2	6	0	0	7	1	0	0	0	1	0	136	0	0	0	0	0
S	4	16	0	3	1	2	1	0	2	5	0	8	0	0	3	1	6	3	99	1	0	0	0
T	0	2	0	1	5	1	6	2	0	1	1	0	0	0	0	2	0	1	2	169	4	1	0
U	1	0	2	1	0	0	0	7	0	0	1	0	3	3	8	0	0	0	0	0	179	2	1
V	0	0	0	0	0	0	0	4	0	0	0	0	0	0	1	1	0	0	0	1	0	167	5
W	0	1	0	0	0	0	0	4	0	0	1	0	4	2	1	0	0	2	0	0	1	5	173
X	0	0	0	6	9	0	4	0	2	0	0	1	0	0	0	0	4	3	2	2	0	0	0
Y	0	0	0	2	0	4	0	1	0	1	0	0	0	0	1	1	8	0	0	14	1	11	0
Z	0	1	0	1	13	0	0	0	0	6	0	0	0	0	0	0	0	0	12	1	0	0	0

Hardest letter to classify H (46.7%)
Easiest letter to classify V (92.3%)

Error rate 22.9%

Complexity params:

- decay=1

Benchmark results

- maxit=500
- decay=0

Time to run: 103.752 seconds.
Time to use: 0.045 seconds.
Error rate: 23.2%

- `maxit=500`
- `decay=1`

Time to run: `106.653` seconds.
Time to use: `0.046` seconds.
Error rate: 22.9%

- `maxit=500`
- `decay=2`

Time to run: `109.231` seconds.
Time to use: `0.044` seconds.
Error rate: 23.0%

- `maxit=500`
- `decay=3`

Time to run: `125.131` seconds.
Time to use: `0.046` seconds.
Error rate: 23.2%

How where the complexity params chosen?

Weight decay is a way of penalizing high or low values for the weights in the model. We iterated with values between 1 and 35 and concluded that there seemed to be a global maximum at a weight value of 3.

Conclusion

A higher `decay` value results in a slower build time, but doesn't effect the time it takes to *use the model.

- use

```
predict(model, testData, ...)
```

qda

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
A	184	0	0	2	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	2	1	0
B	0	168	0	3	2	1	1	2	1	0	0	0	0	0	1	1	0	10	1	0	0	1	0
C	0	0	170	0	8	0	4	0	0	0	0	2	2	0	4	0	0	0	0	0	0	0	0
D	0	2	0	197	0	0	0	2	0	0	0	0	4	0	5	0	0	3	2	1	2	0	0
E	0	0	0	0	149	1	9	0	0	0	0	2	0	0	0	0	3	0	1	0	0	0	0
F	0	1	0	1	4	181	2	3	1	3	0	0	0	14	0	3	0	0	2	3	0	0	0
G	0	3	11	3	0	2	154	1	0	0	0	0	1	0	3	0	4	0	5	0	0	2	0
H	0	3	1	9	0	0	4	133	0	0	15	0	1	0	9	0	2	7	0	0	2	0	0
I	0	1	1	6	0	1	0	1	146	7	1	0	0	1	0	0	0	0	9	0	0	0	0
J	0	0	0	2	0	2	0	1	5	171	0	0	0	0	1	0	2	1	4	0	0	0	0
K	0	1	5	2	1	0	1	8	0	0	159	0	0	0	0	0	3	16	0	0	0	0	0
L	0	0	2	0	4	0	3	2	0	0	0	162	0	0	0	0	4	1	8	0	0	0	0
M	1	4	0	0	0	0	3	1	0	0	0	0	181	0	1	0	0	0	0	0	0	0	1
N	0	0	0	3	0	1	0	2	0	0	1	0	0	178	6	0	0	2	0	0	0	0	1
O	1	0	0	0	0	1	0	3	0	0	0	0	0	0	158	0	2	0	0	0	0	0	1
P	0	1	0	0	0	14	7	1	0	0	0	0	0	0	0	170	2	2	0	0	0	0	0
Q	2	1	0	0	0	0	2	2	0	0	0	2	0	0	11	0	181	0	2	0	0	0	0
R	0	4	0	2	0	0	2	1	0	0	1	1	1	1	0	0	0	163	0	0	1	1	0
S	0	5	2	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	160	3	0	0	0
T	1	0	0	1	5	3	4	2	0	0	4	0	0	0	0	0	2	1	1	178	3	0	0
U	0	0	0	0	0	0	0	1	0	0	2	0	2	0	5	0	0	0	0	0	198	0	6
V	0	4	0	0	0	0	3	1	0	0	0	0	0	0	1	1	0	0	0	0	1	155	3
W	0	0	0	0	0	0	1	2	0	0	0	0	4	0	2	0	0	0	0	0	5	0	174
X	0	1	1	0	0	1	0	4	1	0	7	0	0	0	0	0	1	1	13	0	1	0	0
Y	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	3	0	0	0	12	0	18	0
Z	2	0	0	0	1	0	0	0	0	2	1	0	0	0	0	0	2	0	6	1	0	0	0

Hardest letter to classify H (70.4%)
Easiest letter to classify O (94.6%)

Error rate 12.8%

No complexity params where used.

Ida

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	180	0	0	0	0	0	0	3	0	3	3	0	2	0	5	0	0	0	3	0	2	0	2			
B	0	128	0	5	0	1	2	5	0	0	2	0	0	0	5	0	0	11	13	0	0	0	0			
C	0	4	137	0	8	2	10	2	0	0	14	0	1	0	4	0	0	0	5	0	5	0	0			
D	0	11	0	144	0	0	2	5	2	4	0	0	3	2	4	0	0	4	5	0	0	0	0			
E	0	14	22	0	85	3	22	0	0	0	1	0	0	0	0	0	0	4	4	0	2	0	0	2		
F	0	9	0	3	0	118	2	0	2	0	0	0	0	1	0	21	4	1	3	2	0	0	3			
G	3	8	25	0	1	0	86	3	0	0	7	0	1	0	3	0	7	7	8	0	0	0	1			
H	0	1	0	14	0	0	2	84	0	0	15	0	0	11	19	1	8	9	0	0	5	1	0	1		
I	0	9	0	2	1	1	0	0	153	3	0	0	0	0	0	3	5	1	9	0	0	0	0			
J	1	2	0	1	0	7	0	3	17	134	0	0	0	0	2	1	4	0	6	0	0	0	0			
K	0	6	1	2	5	0	7	4	0	0	138	0	1	2	3	0	1	20	0	0	9	1	1	1		
L	3	6	1	0	8	0	10	0	2	6	1	128	1	0	1	0	3	0	6	0	0	0	0			
M	1	2	0	0	0	0	0	5	0	0	2	0	178	4	3	0	0	1	0	0	0	0	5			
N	0	0	0	9	0	0	0	4	0	0	2	0	5	158	3	0	0	1	0	2	0	1	13			
O	6	1	2	17	0	1	5	23	0	0	2	0	0	1	127	0	8	1	0	0	0	0	4			
P	0	4	0	2	0	11	1	0	0	0	1	0	1	0	5	157	8	0	0	1	0	0	1			
Q	3	9	0	1	6	0	19	1	0	1	7	3	1	0	17	0	125	1	9	0	0	1	2			
R	0	18	0	10	0	0	0	5	0	0	5	0	0	3	2	0	0	143	0	0	0	0	0			
S	6	23	0	1	1	6	9	1	1	5	0	3	0	0	0	0	0	9	92	0	0	1	0	1		
T	0	4	0	0	7	19	8	4	1	0	5	0	0	0	2	1	0	0	5	116	1	1	0			
U	0	0	1	2	0	0	0	12	0	0	4	0	7	7	7	0	0	0	0	0	174	0	1			
V	0	0	0	0	0	7	2	6	0	0	2	0	0	0	6	2	0	1	0	1	1	146	5			
W	0	0	0	0	0	0	0	11	0	0	2	0	7	4	0	0	0	2	0	0	0	1	159			
X	0	6	0	3	3	0	2	0	1	0	1	0	0	0	1	0	12	2	16	2	9	0	0	14		
Y	0	0	0	2	0	17	0	0	0	0	0	0	0	0	0	0	16	0	5	13	0	45	0			
Z	0	0	0	0	13	0	3	0	0	4	0	0	0	0	0	0	11	0	19	0	0	0	0			

Hardest letter to classify S (46.2%)
Easiest letter to classify M (88.6%)

Error rate 30.8%

No complexity params where used.

nnet

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
A	166	0	0	0	0	2	0	3	0	4	1	1	4	2	4	0	1	1	2	0	0	0	1
B	0	146	0	8	2	0	1	3	0	0	0	0	0	0	0	1	0	23	3	0	0	1	0
C	1	0	134	2	11	0	6	0	0	0	25	5	0	0	6	0	1	1	3	1	1	0	0
D	2	5	0	152	2	0	0	2	0	0	0	0	0	1	6	3	0	12	7	0	1	0	0
E	0	4	0	1	126	1	6	0	0	0	1	0	0	0	0	0	5	5	7	2	0	0	0
F	0	6	2	2	2	131	0	1	1	0	0	0	0	1	0	10	0	2	6	5	0	0	1
G	0	2	28	0	2	0	114	1	0	0	7	0	3	0	3	1	12	7	5	0	1	0	1
H	0	6	3	13	2	0	0	118	0	0	3	0	2	0	9	4	5	16	1	0	4	1	0
I	0	2	0	2	0	3	0	0	150	4	0	0	0	0	0	2	1	2	15	0	0	0	0
J	0	4	0	0	0	6	0	2	5	138	0	0	0	0	1	4	0	0	13	0	0	0	0
K	1	1	3	0	8	0	0	3	0	0	140	0	1	4	0	0	0	18	0	0	3	0	0
L	1	3	6	1	8	0	6	0	0	0	7	145	0	1	0	0	4	3	1	0	3	0	0
M	0	1	0	1	0	0	0	3	0	0	1	0	170	7	0	0	0	7	0	0	0	0	2
N	1	2	0	2	0	2	0	4	0	0	1	0	7	164	11	0	0	1	0	0	4	0	3
O	3	0	0	12	0	0	2	7	0	0	0	0	0	1	133	2	8	3	0	0	2	0	9
P	0	1	0	5	2	10	4	0	2	1	1	0	0	0	1	163	0	1	0	0	0	1	1
Q	3	1	0	1	14	0	10	1	0	0	0	1	0	0	16	0	148	2	11	0	0	2	2
R	0	10	0	1	0	1	1	4	0	1	7	0	0	2	2	0	1	148	0	0	0	0	1
S	3	20	1	1	4	1	1	0	5	1	0	3	0	0	3	1	1	4	117	5	0	1	0
T	0	5	0	3	3	6	6	1	0	0	4	0	0	0	0	0	0	1	2	155	2	0	0
U	1	0	0	1	0	0	1	4	0	0	8	0	14	8	9	0	0	0	0	0	164	0	3
V	0	2	0	1	1	0	0	8	0	0	0	0	0	1	0	1	2	0	0	0	0	153	10
W	0	0	0	0	0	0	0	2	0	0	0	0	9	1	5	0	0	3	0	0	0	1	181
X	0	3	0	4	9	4	0	0	1	1	1	1	0	0	1	0	5	0	4	2	1	0	0	14
Y	0	0	0	0	0	3	0	0	0	0	0	0	2	0	0	8	5	0	5	14	0	14	0
Z	0	0	0	0	12	1	0	0	1	0	0	0	0	0	0	0	1	0	28	0	0	0	0

Hardest letter to classify G (60.0%)
Easiest letter to classify W (89.6%)

Error rate 19.5%

Complexity params:

- `size=22`
- `decay=11`

How where the complexity params chosen?

We used brute force to find the best combination. We tried to find the global maximum by iterating over a large span for each param.

Benchmark results

- `size=22`
- `decay=8`

Time to run: `207.906` seconds.
Time to use: `0.091` seconds.
Error rate: 23.3%

- `size=22`
- `decay=9`

Time to run: `178.885` seconds.
Time to use: `0.048` seconds.
Error rate: 21.1%

- `size=22`

- decay=10

Time to run: 142.829 seconds.
Time to use: 0.056 seconds.
Error rate: 20.4%

- size=22
- decay=11

Time to run: 153.440 seconds.
Time to use: 0.049 seconds.
Error rate: 19.5%

- size=22
- decay=12

Time to run: 150.814 seconds.
Time to use: 0.052 seconds.
Error rate: 19.9%

- size=22
- decay=13

Time to run: 165.992 seconds.
Time to use: 0.052 seconds.
Error rate: 22.8%

- size=22
- decay=14

Time to run: 143.739 seconds.
Time to use: 0.048 seconds.
Error rate: 23.6%

- size=22
- decay=15

Time to run: 144.457 seconds.
Time to use: 0.050 seconds.
Error rate: 26.6%

- size=22
- decay=16

Time to run: 143.988 seconds.
Time to use: 0.048 seconds.
Error rate: 27.0%

- size=15
- decay=9

Time to run: 109.825 seconds.
Time to use: 0.044 seconds.
Error rate: 24.0%

- size=15
- decay=10

Time to run: 116.584 seconds.
Time to use: 0.043 seconds.
Error rate: 24.7%

- `size=15`
- `decay=12`

Time to run: `124.040` seconds.
Time to use: `0.043` seconds.
Error rate: 26.2%

- `size=15`
- `decay=11`

Time to run: `134.863` seconds.
Time to use: `0.052` seconds.
Error rate: 26.1%

Conclusion

A lower `decay` value results in faster build time, the same goes for the `size` param.

svm

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
A	190	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0
B	0	182	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	3	0	0	0	1	0
C	0	0	163	0	2	0	6	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0
D	0	1	0	196	1	0	0	4	0	0	0	0	0	1	2	0	0	0	0	0	0	0	0
E	0	0	0	0	186	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	221	0	0	0	1	0	0	0	1	0	2	0	0	0	1	0	0	0
G	0	0	0	0	1	0	178	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	2	0	0	1	182	0	0	5	0	0	1	1	0	1	7	0	0	2	0	0
I	0	0	0	0	0	0	0	0	173	8	0	0	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	1	0	0	0	0	0	175	0	0	0	0	0	0	0	0	0	0	0	0	0
K	0	1	0	1	0	0	0	4	0	0	187	0	0	0	0	0	0	1	0	0	0	0	0
L	0	0	1	0	4	0	1	0	0	0	0	172	0	0	0	0	0	0	0	0	0	0	0
M	0	1	0	0	0	0	2	0	0	0	0	0	197	0	0	0	0	0	0	0	0	0	0
N	0	0	0	1	0	0	0	2	0	0	0	0	1	187	0	0	0	2	0	0	0	0	0
O	0	0	1	2	0	0	0	0	0	1	0	0	0	0	179	0	1	2	0	0	1	0	0
P	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	206	0	1	0	0	0	0	0
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	198	0	0	0	0	0	0
R	0	4	0	2	0	0	0	4	0	0	2	0	0	0	0	0	0	179	0	0	0	1	0
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	181	1	0	0	0
T	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	210	0	0	0
U	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	193	0	0
V	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	175	0
W	0	0	0	0	0	0	0	0	0	0	0	0	2	0	1	0	0	0	0	0	0	0	179
X	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	1	1	0
Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Hardest letter to classify H (90.1%)
Easiest letter to classify X (99.5%)

Error rate 2.9%

Complexity params:

- `cost=12`
- `epsilon=1`
- `kernel=radial`

How where the complexity params chosen?

We used the same technique as we did for the `nnet` method.

Benchmark results

- cost=10
- epsilon=0.0
- kernel=radial

Time to run: 34.963 seconds.
Time to use: 4.706 seconds.
Error rate: 2.9%

- cost=10
- epsilon=0.2
- kernel=radial

Time to run: 35.808 seconds.
Time to use: 4.818 seconds.
Error rate: 2.9%

- cost=10
- epsilon=0.4
- kernel=radial

Time to run: 34.435 seconds.
Time to use: 4.393 seconds.
Error rate: 2.9%

- cost=10
- epsilon=0.7
- kernel=radial

Time to run: 33.239 seconds.
Time to use: 4.494 seconds.
Error rate: 2.9%

- cost=10
- epsilon=1.0
- kernel=radial

Time to run: 32.253 seconds.
Time to use: 4.692 seconds.
Error rate: 2.9%

- cost=9
- epsilon=1.0
- kernel=radial

Time to run: 33.762 seconds.
Time to use: 4.562 seconds.
Error rate: 3.1%

- cost=10
- epsilon=1.0
- kernel=radial

Time to run: 32.760 seconds.
Time to use: 4.169 seconds.
Error rate: 3.1%

- cost=11
- epsilon=1.0
- kernel=radial

Time to run: 32.925 seconds.

Time to use: 4.222 seconds.
Error rate: 3.1%

- cost=12
- epsilon=1.0
- kernel=radial

Time to run: 31.883 seconds.
Time to use: 4.207 seconds.
Error rate: 2.8%

- cost=13
- epsilon=1.0
- kernel=radial

Time to run: 32.081 seconds.
Time to use: 4.510 seconds.
Error rate: 2.8%

- cost=14
- epsilon=1.0
- kernel=radial

Time to run: 32.920 seconds.
Time to use: 5.041 seconds.
Error rate: 2.8%

- cost=15
- epsilon=1.0
- kernel=radial

Time to run: 37.2968 seconds.
Time to use: 4.489 seconds.
Error rate: 2.9%

- cost=16
- epsilon=1.0
- kernel=radial

Time to run: 35.168 seconds.
Time to use: 4.248 seconds.
Error rate: 3.0%

- cost=17
- epsilon=1.0
- kernel=radial

Time to run: 32.353 seconds.
Time to use: 4.393 seconds.
Error rate: 3.0%

- cost=18
- epsilon=1.0
- kernel=radial

Time to run: 31.690 seconds.
Time to use: 4.040 seconds.
Error rate: 3.0%

- cost=19
- epsilon=1.0

- `kernel=radial`

Time to run: `32.814` seconds.
Time to use: `4.135` seconds.
Error rate: 3.0%

How were the complexity params chosen?

We used the same technique as we did for the `nnet` method.

Best classification method

The `svm` method with `cost=12`, `epsilon=1`, `kernel=radial` gives the best result. On average `svm` predicts `97.1%` of the testing data correctly. This method, however, also has a significantly longer execution time. It seems like a good choice for this data set if you only intend to run it a few times. Otherwise, k nearest neighbour might be better suited.