Machine Learning CS 726, Fall 2011 *Christoph Schulze, Michael Whidby, and Kotaro Hara*

Project 3: Basic classification

WU1

Two eigenvectors both pass through the origin, perpendicular to each other. However, depending on your random data, slopes will vary.

WU2

Figure 1: wu2: Normalized eigenvalues plotted on the horizontal axis.

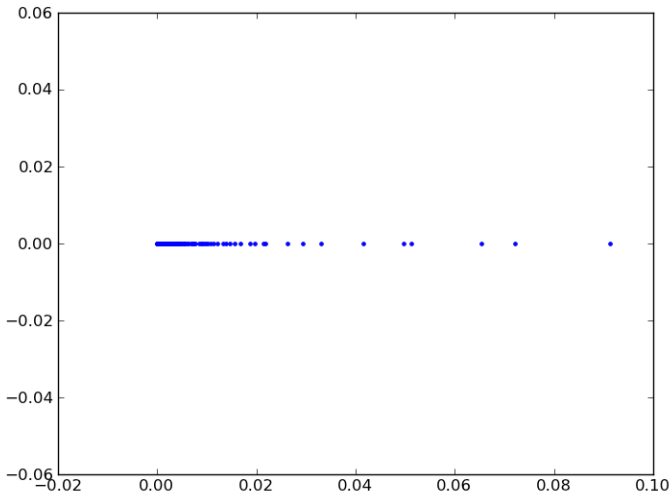


Figure 1 shows the normalized eigenvalues. To account for 90% variance, we have to include 82 eigenvectors. To account for 95% variance, we have to include 136 eigenvectors.

WU3

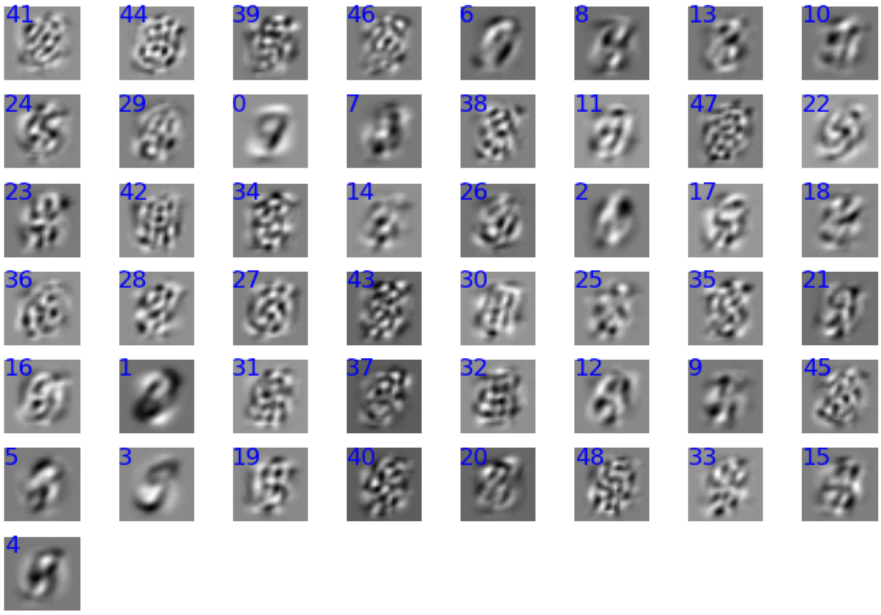
Figure 2 shows the plot of the top 50 eigenvectors. The majority of the results bear no resemblance to digits.

WU4

Figure 3 shows the plot of the data. Vanilla PCA will find this data difficult because the variance in all directions is similar. This means that eigenvectors can point in any direction.

The large eigenvalues have no significance. For example, with a poly3 kernel, the eigenvalues are ridiculously large due to cubing the dot product. What does hold significance, however, is the normalized eigenvalues.

Figure 2: wu3: Images of digits transformed by top 50 eigenvectors.



WU5

Figure 4 shows the result of PCA. PCA did not do what we want it to do :'(In addition to the reasons listed in WU4, vanilla PCA did not do what we wanted to do because the resulting data is not linearly separable.

WU6

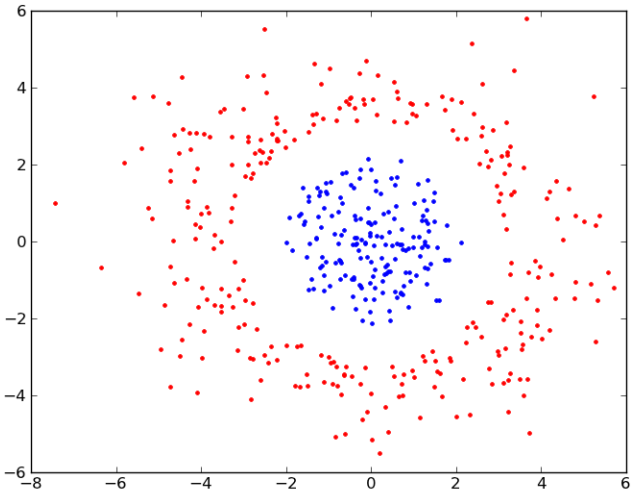
The eigenvalues for KPCA with rbf1 kernel is [0.08270371 0.07927822], which is significantly smaller than the eigenvalues for KPCA with linear kernel: [5.68035099 1.51738947]. This means that the rbf1 kernel does a much better job at making the data linearly separable than the linear kernel. See the Figure 11 for a plot of the rbf1 kernel and Figure 6 for the plot of the linear kernel for a visual comparison.

WU7

Table 1 shows that the linear kernel does the best job in terms of getting most variance in the first two principle component. The poly2 kernel and the poly3 kernel also do good job in getting much of variance with the first two principle component. However, if you look at plotted data points, none of three kernels make data linearly separable.

On the other hand, RBF kernels make the data linearly separable. Among all RBF kernels, RBF 0.2 does

Figure 3: wu4: Plot of circular data that are generated with shared center.



Kernel	Top eigen values	Variance	λ_0	$\lambda_0 + \lambda_1$
Linear	(6.163, 5.757)	(3.032E+03, 2.833E+03)	0.517	1.
Poly2	(6.410E+01, 6.277E+01)	(3.154E+04, 3.088E+04)	0.312	0.618
Poly3	(2.906E+03, 2.477E+03)	(1.429E+06, 1.219E+06)	0.379	0.701
RBF 0.2	(1.454E-01, 1.008E-01)	(7.153.E+01, 4.959E+01)	0.176	0.297
RBF 0.5	(1.122E-01, 6.507E-02)	(5.522E+01, 3.201E+01)	0.123	0.194
RBF 1	(7.797E-02, 5.560E-02)	(3.836E+01, 2.735E+01)	0.082	0.140
RBF 2	(5.140E-02, 4.200E-02)	(2.529E+01, 2.067E+01)	0.053	0.096
RBF 5	(2.830E-02, 2.551E-02)	(1.392E+01 1.255E+01)	0.029	0.055

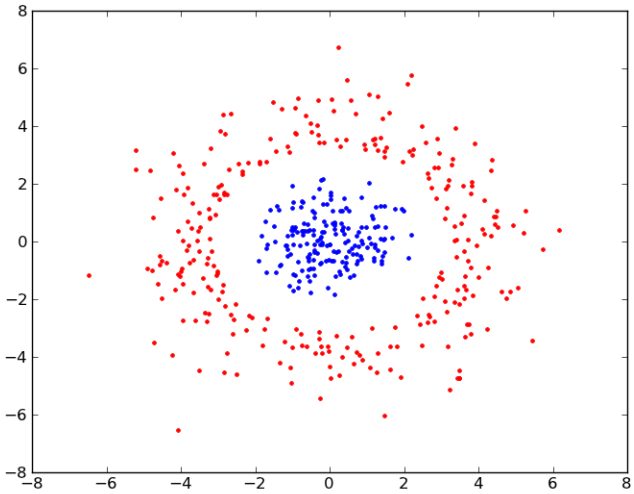
Table 1: Examples of convergence

the best job in getting as much of the variance on the first two principle component.

Plots for the various kernels:

- linear: Figure 6
- poly2: Figure 7
- poly3: Figure 8
- rbf0.2: Figure 9
- rbf0.5: Figure 10
- rbf2: Figure 12
- rbf5: Figure 13

Figure 4: wu5: Plot of data points transformed by vanilla PCA



WU8

The Observation vectors [1,1,1,2] and [2,1,1,2] are different in only one place and lead to the outputs [0,0,0,1] and [1,1,1,1] that differ in more than one place.

```
(a,b,pi) = datasets.getHMMData()
print hmm.viterbi(array([1,1,1,2]), a, b, pi)    -> [0 0 0 1]
print hmm.viterbi(array([2,1,1,2]), a, b, pi)    -> [1 1 1 1]
```

WU9

It seems to have hit a local maximum at the log probability -2.77259 since there was no increase in the log probabilities over 20 iterations)

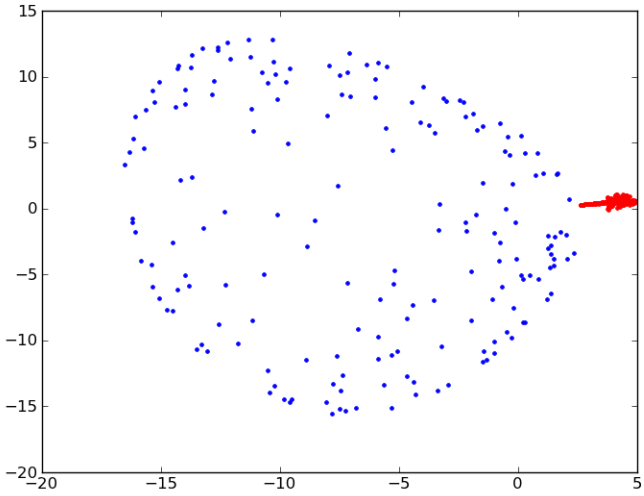
With three states the log error decreases and the model is much more likely to produce the observed emissions.

With four states it does something akin to overfitting where it creates a transition matrix that perfectly fits the observed emissions. The log probability that the observed sequence is generated by this model is 4.16334e-17 which basically means that it is impossible to create any other sequence. It now perfectly emits the observation sequence but gets stuck in state 4 after the third transition and will continue to emit Wet with a very high probability. (95%) So it perfectly fits the observation but looses the ability to generalize.

Results for 2 States:

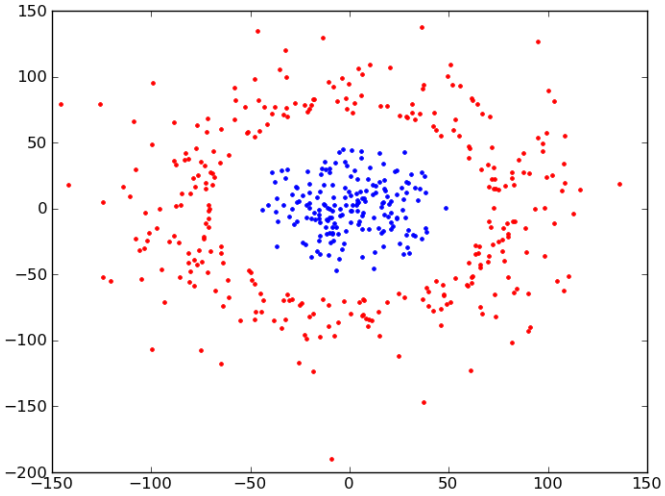
```
iteration 50    ... log probability -2.77259
a_em [[  1.20100122e-15   1.00000000e+00]
 [  1.00000000e+00   0.00000000e+00]]

b_em [[ 0.   0.5  0.5]
 [ 0.5  0.5  0. ]]
```

Figure 5: wu6: Plot of data points transformed by a rbf1 kernel




```
pi_em [ 0.  1.]


Results for 3 States:

iteration 50    ... log probability -1.21641
a_em [[  0.00000000e+000   1.00000000e+000   7.50925376e-038]
 [  7.62540047e-298   3.33436869e-001   6.66563131e-001]
 [  2.66946976e-008   2.58112825e-005   9.99974162e-001]]

b_em [[  1.00000000e+000   0.00000000e+000   9.10012371e-227]
 [  0.00000000e+000   1.00000000e+000   5.21870817e-018]
 [  0.00000000e+000   3.33229765e-001   6.66770235e-001]]

pi_em [ 1.  0.  0.]


Results for 4 States:

iteration 50    ... log probability 4.16334e-17
a_em [[ 0.          0.          0.          1.        ]
 [ 1.          0.          0.          0.        ]
 [ 0.          1.          0.          0.        ]
 [ 0.00970598  0.02388482  0.00708391  0.95932529]]

b_em [[ 0.  1.  0.]
 [ 0.  1.  0.]
 [ 1.  0.  0.]
 [ 0.  0.  1.]]

pi_em [ 0.  0.  1.  0.]
```

Figure 6: wu7: KPCA with a linear kernel



WU10

It learned how likely certain letters appear and in what pairs of combinations they are likely to appear.

- It learned that _,e,a,o,i and t,r,n,s are the most frequent letters. (This can be calculated by combining the emission probabilities with the likelyhood of beeing in state 0 andor 1.)

-It learned that it is not likely for two blanks to appear after each other, but single blanks are very common. You can see this by looking at the transition probabilities in combination with the emission probabilities. The model emits a _ with aprobability of 37% if it is in state 1 but only with 0.0014% if it is in state 0. Since it is much more likely to transition from state 1 to 0 than to stay in state 0 the combination of two _ is not very likely. This matches with the observed text where double blanks don't appear at all.

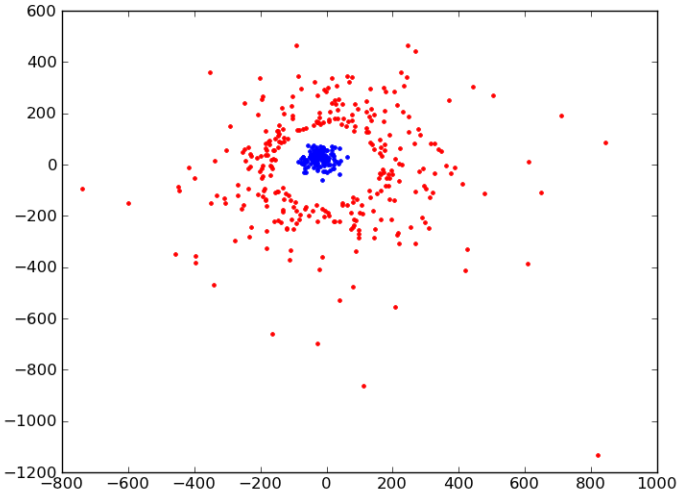
- Similar to the blanks you can see that two vowels in a row are much less likely than other combinations and that words tend to start more often without vowels.

- The start state indicates that it also learned that the text is unlikely to start with a blank.

However with only two states the model cannot really emit the observed sequence anymore since it is much too general for that.

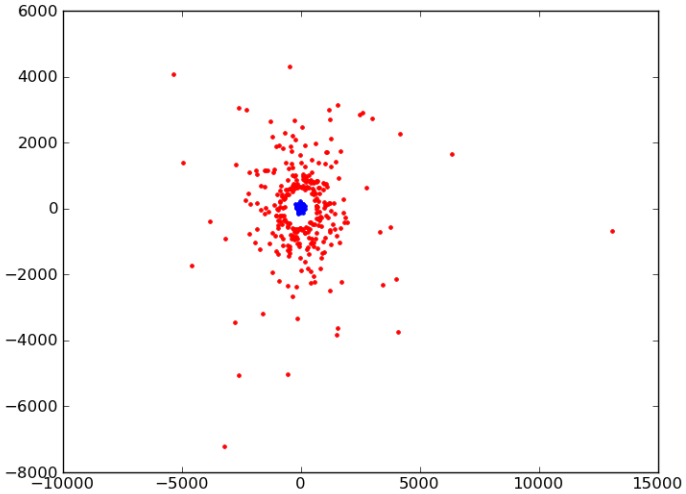
```
iteration 1     ... log probability -11135.1
...
iteration 50    ... log probability -9220.35

Initial state probabilities:
       state(0):       1
       state(1):       3.07405e-18

Transition probabilities:
FROM\TO
       0       1
0       0.270982        0.729018
1       0.73666         0.26334
```

Figure 7: wu7: KPCA with a poly2 kernel




```
Emission probabilities:
        State 0:
                t  0.146579
                r  0.10483
                n  0.0989395
                s  0.0983503
                h  0.0894233
                d  0.0683162
                l  0.0645881
                c  0.04888
                m  0.0447589
                y  0.0358725
                f  0.035336
                p  0.0335527
                w  0.0312134
                b  0.0265008
                g  0.0253234
                v  0.0206126
                k  0.0172418
                a  0.00509005
                j  0.0017668
                x  0.0017668
                z  0.000588933
                e  0.000413468
                u  4.23781e-05
                _  1.35474e-05
                q  1.17165e-08
                o  1.076e-09
                i  7.3767e-16
        State 1:
                _  0.372007
```

Figure 8: wu7: KPCA with a poly3 kernel




```
e  0.228152
a  0.12164
o  0.118451
i  0.102975
u  0.0445997
h  0.00723833
y  0.00183861
k  0.00162122
q  0.00119046
l  0.000196595
t  6.60738e-05
p  1.66102e-05
c  1.47489e-06
s  1.4587e-06
b  1.22527e-06
n  1.16083e-06
g  7.58508e-07
r  3.44083e-07
d  1.86647e-09
x  2.4489e-13
f  8.2358e-17
m  2.43184e-17
w  4.14047e-20
v  2.7307e-25
j  2.9153e-39
z  1.77391e-40
```

Figure 9: wu7: KPCA with a rbf0.2 kernel

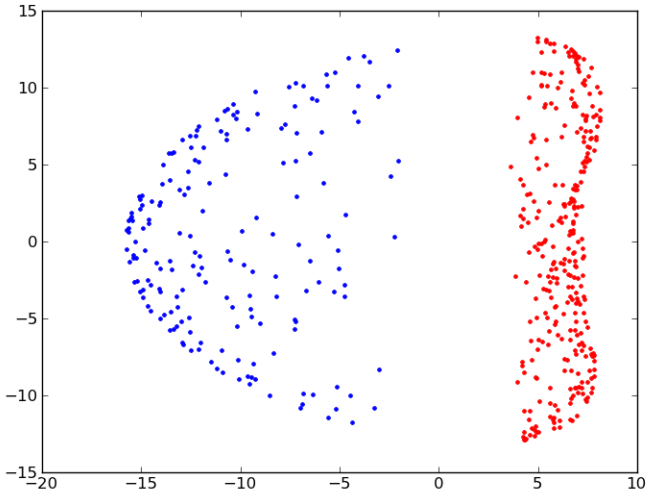


Figure 10: wu7: KPCA with a rbf0.5 kernel

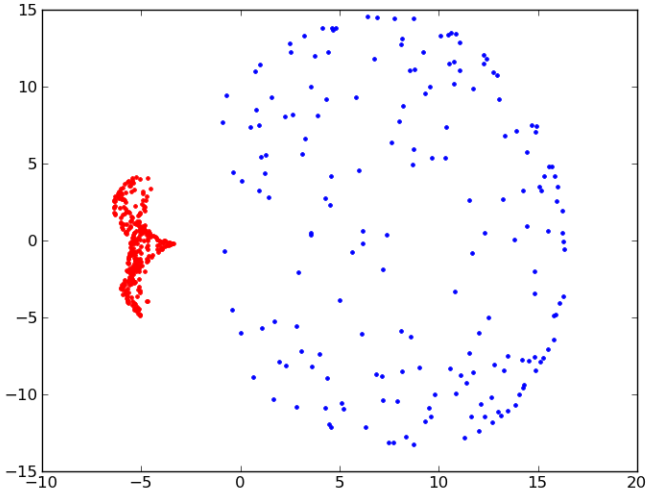


Figure 11: wu7: KPCA with a rbf1 kernel

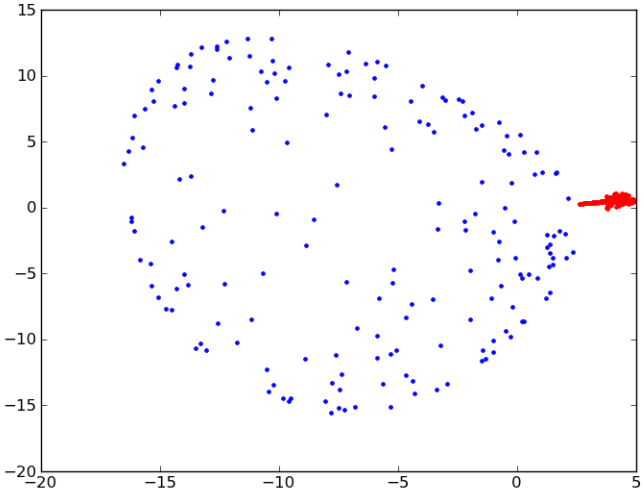


Figure 12: wu7: KPCA with a rbf2 kernel

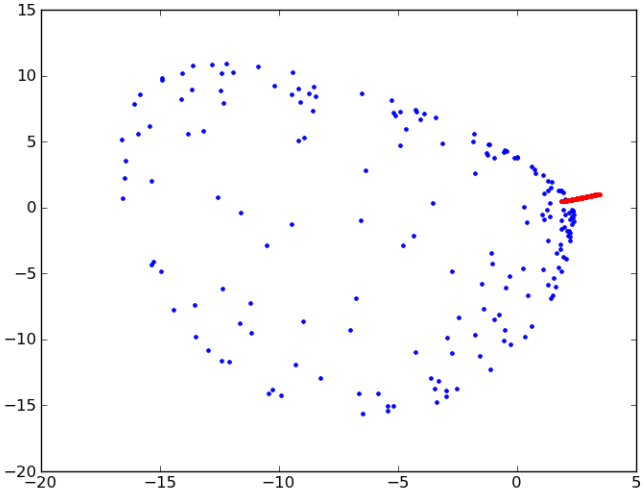


Figure 13: wu7: KPCA with a rbf5 kernel