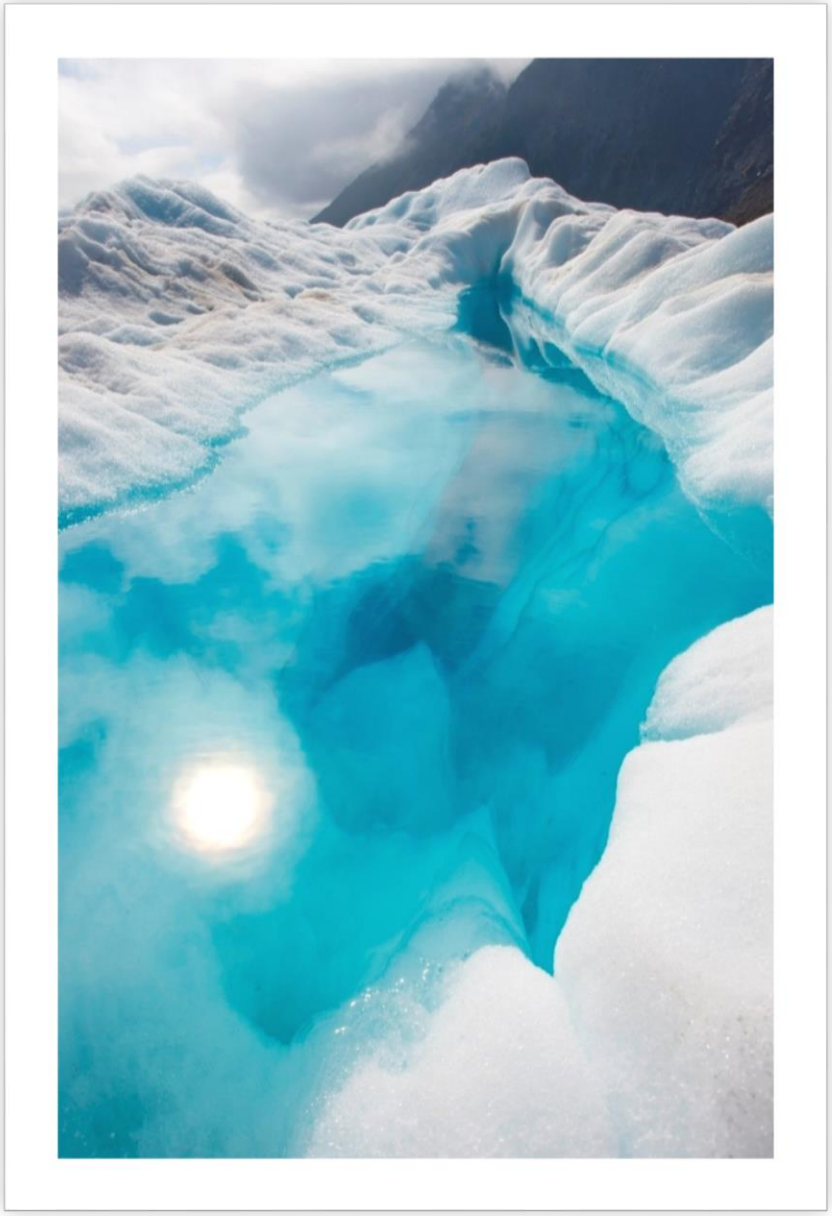


Exploring kNN, MLE and SVM Classification Techniques

ON LETTER RECOGNITION DATA SET

Matt Hall | CS559: Machine Learning: Fundamentals and Applications | December 16, 2015

Introduction

This project will explore 3 classification methods to classify a set of randomly distorted English letters. Primary Component Analysis (PCA) will be performed on the data, prior to classification to investigate potential dimensionality reductions. The classification techniques incorporated into this project are Maximum Likelihood Estimation (MLE), k-Nearest Neighbor (kNN) and the Support Vector Machine (SVM). The outcome of this study will be a graph comparing the 3 classifiers as well as some variations within the various classifier parameters.

Data

The data set used for this project came from the UCI Machine Learning Repository - https://archive.ics.uci.edu/ml/datasets/Letter+Recognition. The data was donated by David J. Slate (dave@math.nwu.edu) and phone (708)491.3867.

The objective is to identify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet (i.e. 26 classes). The character images were based on 20 different fonts and each letter within these 20 fonts was randomly distorted to produce a file of 20,000 unique stimuli. Each stimulus was converted into 16 primitive numerical attributes (statistical moments and edge counts) which were then scaled to fit into a range of integer values from 0 through 15.

The data set contains 20,000 samples, with the following class distribution:

789 A	766 B	736 C	805 D	768 E	775 F	773 G
734 H	755 I	747 J	739 K	761 L	792 M	783 N
753 O	803 P	783 Q	758 R	748 S	796 T	813 U
764 V	752 W	787 X	786 Y	734 Z		

Each record consists of 17 attributes, consisting of 1 letter category and 16 numeric features. The attributes are as follows:

1.	lettr	capital letter	(26 values from A to Z)
2.	x-box	horizontal position of box	(integer)
3.	y-box	vertical position of box	(integer)
4.	width	width of box	(integer)
5.	high	height of box	(integer)
6.	onpix	total # on pixels	(integer)
7.	x-bar	mean x of on pixels in box	(integer)
8.	y-bar	mean y of on pixels in box	(integer)
9.	x2bar	mean x variance	(integer)
10.	y2bar	mean y variance	(integer)
11.	xybar	mean x y correlation	(integer)
12.	x2ybr	mean of x * x * y	(integer)
13.	xy2br	mean of x * y * y	(integer)
14.	x-ege	mean edge count left to right	(integer)
15.	xegvy	correlation of x-ege with y	(integer)
16.	y-ege	mean edge count bottom to top	(integer)
17.	yegvx	correlation of y-ege with x	(integer)

Methods

The classifiers were trained on the first 10,000 items and the resulting model was used to predict the category for the remaining 10,000 items. Each classifier with its specific parameter set was run 10 times, randomizing the order of the 20,000 items for each run. The tabulated accuracy rates are an average of the 10 runs. Standard deviations have not been recorded b/c they were all < 0.5, with most below 0.3.

PCA was performed on the resulting randomized training data for each of the 10 runs, then sorted by most important to least. An example set of the 16 principal components is in the Results section. Each classifier was then used after applying the 1st principal component (PC), then the 1st two PCs, then the 1st three PCs, and so on until all 16 PCs were included. Resulting in a data point for each classifier at each PC dimensional inclusion.

Maximum Likelihood Estimation (MLE), being a simple classifier was implemented to see how it would perform against the other two classifiers.

k-Nearest Neighbor (kNN) was also used. The odd numbers in range (1, 11), inclusive, were arbitrarily selected as the k-value parameters for this classifier.

This turned out to be a good test range as will be seen in the Results section. Again, each k-value was run against identical training and test data, with the sets only varying on each of the 10 runs. In order to calculate nearest neighbors, the KDTree function, included in python's scipy spatial library, was used.

Support Vector Machine (SVM), an advanced classification technique was performed to see if it would outperform MLE and kNN. The LIBSVM python extension was used for classification. Kernel selection was done on a few sample runs using linear, polynomial, radial basis function (RBF) and sigmoid. RBF significantly outperformed the other 3 kernels, so it was selected as the kernel for this experiment, with its degree parameter set to the default value of 3. The cost parameter was also optimized and found to perform best with C = 6. The svm_type parameter remained as the default nu-SVC. All other parameters of the LIBSVM implementation remained as their default values.

Results

PCA Results:

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1st	0.33	0.60	0.35	0.39	0.37	0.05	-0.03	0.00	0.02	0.03	-0.06	-0.01	0.25	0.00	0.23	-0.01
2nd	0.09	0.08	0.08	0.04	-0.03	-0.26	0.54	-0.16	-0.03	0.22	0.60	-0.13	-0.01	0.28	-0.22	-0.18
3rd	-0.02	0.12	-0.01	-0.02	-0.10	0.05	-0.02	-0.60	0.45	0.43	-0.22	-0.04	-0.38	-0.05	0.14	-0.04
4th	-0.02	0.00	-0.09	0.11	-0.02	-0.30	0.10	0.32	0.53	-0.32	0.21	0.24	-0.28	0.08	0.46	0.05
5th	-0.04	-0.43	0.09	-0.18	0.33	0.10	0.21	-0.38	-0.04	-0.21	0.00	-0.17	0.31	0.12	0.53	-0.05
6th	0.00	-0.03	0.10	-0.02	0.04	-0.04	-0.26	-0.43	0.06	-0.21	0.28	0.71	0.20	-0.06	-0.23	0.01
7th	0.02	-0.15	0.00	-0.10	0.04	0.33	0.05	0.29	-0.06	0.64	0.25	0.36	0.10	-0.03	0.30	0.25
8th	-0.20	0.26	-0.25	0.25	-0.07	0.31	0.20	-0.24	-0.43	-0.25	0.09	0.09	-0.35	0.15	0.16	0.38
9th	-0.05	0.00	0.06	0.04	-0.03	0.71	-0.13	0.07	0.43	-0.20	0.34	-0.26	0.03	0.09	-0.19	-0.09
10th	0.20	-0.03	0.22	-0.16	-0.12	-0.02	0.34	-0.02	0.27	-0.11	-0.23	-0.02	0.17	0.08	-0.25	0.72
11th	-0.26	-0.33	0.16	0.45	0.43	0.09	0.36	0.08	0.07	0.04	-0.19	0.17	-0.21	-0.24	-0.28	-0.08
12th	0.16	-0.33	0.26	0.19	0.18	-0.12	-0.46	0.00	-0.10	0.09	0.05	-0.10	-0.32	0.58	-0.07	0.20
13th	-0.13	-0.05	0.02	0.11	0.13	-0.24	-0.25	-0.11	0.00	0.04	0.42	-0.37	-0.02	-0.58	0.03	0.40
14th	-0.52	0.19	-0.43	0.06	0.31	-0.16	-0.11	0.01	0.22	0.17	-0.06	-0.05	0.37	0.35	-0.12	0.15
15th	0.14	-0.29	-0.20	0.66	-0.50	-0.02	-0.01	-0.09	0.06	0.07	-0.05	-0.04	0.37	0.02	0.09	-0.01
16th	0.63	-0.10	-0.64	-0.03	0.37	0.06	0.02	-0.04	0.05	-0.02	0.02	0.01	-0.09	-0.08	-0.14	0.03

Each row represents the identified primary component vector, with the column labels corresponding to the attribute numbers defined in the Data section of this report.

MLE Results:

	MLE
PCA Dimension(s)	Accuracy
1	5.5
2	16.38
3	28.28
4	42.62
5	51.1
6	62.85
7	68.42
8	72.99
9	75.53
10	79.41
11	81.81
12	83.77
13	85.57
14	86.81
15	87.61
16	88.28

MLE classification, obviously performed better with each inclusion of a PCA dimension. The reason for the low performance with the 1st PC is due to the number of available classes set at 26.

kNN Results:

	kNN (k = 1)	kNN (k = 3)	kNN (k = 5)	kNN (k = 7)	kNN (k = 9)	kNN (k = 11)
PCA Dimension(s)	Accuracy					
1	11.59	8.43	8.03	7.69	7.49	7.25
2	20.16	18.19	18.32	18.38	18.48	18.54
3	38.39	38.2	39.47	39.83	40.05	39.92
4	58.74	58.13	59.05	59.18	59.17	59.07
5	70.14	69.52	70.05	70.14	69.88	69.62
6	80.32	79.95	80.13	79.94	79.49	79.01
7	84.84	83.89	83.84	83.47	83.04	82.44
8	87.3	86.74	86.69	86.22	85.78	85.24
9	89.37	88.69	88.44	87.98	87.47	86.89
10	91.31	90.49	90.2	89.71	89.19	88.73
11	92.33					89.84
12	92.95					90.23
13	93.68					
14	93.99					
15	94.23					
16	94.27					

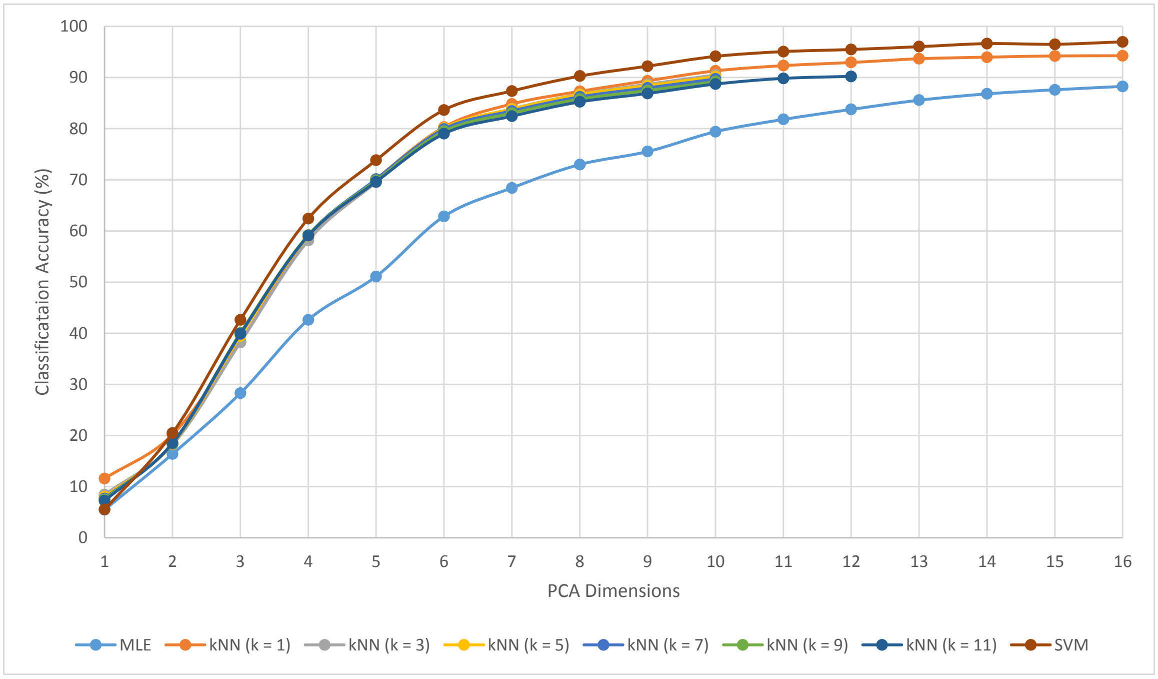
kNN results were interesting in that k = 1 turned out to be the most accurate against the other k-values. Parity amongst k-values was seen during inclusion of the 3rd through 6th PCs, with k = 1 outperforming the other k-values as dimensionality increased. The test was stopped at the 10th dimension for k-values of 3, 5, 7 and 9 because they were slow to run and were not showing signs of improvement against k = 1. Dimensions 11 and 12 were run on k = 11 just to confirm that the trend seen in the separation versus k = 1 would hold. The KDTree was constructed 10 times for each run, but all of the k-values used the same KDTree structure in each of the 10 runs.

SVM Results:

	SVM
PCA Dimension(s)	Accuracy (%)
1	5.51
2	20.47
3	42.61
4	62.42
5	73.85
6	83.66
7	87.37
8	90.28
9	92.23
10	94.14
11	95.08
12	95.49
13	96.04
14	96.64
15	96.51
16	96.99

SVM outperformed the other 2 classifiers.

Here is an overall chart giving a pictorial representation of the classification methods against one another.



Conclusions

MLE performed reasonably well, however, the results are not interesting when compared against kNN and SVM, except that MLE performed much faster than kNN. kNN was the most time consuming, especially at higher dimensionality and higher k-values, however, it performed well and only slightly worse than the SVM classifier. The kNN result with k = 1 performing seems counter-intuitive this should be investigated for the exact reasoning, perhaps running the classifier without PCA. SVM consistently performed the best at every level of dimensionality after surpassing random chance accuracy levels. All the methods began to flatten out around the 10th PC inclusion. Both kNN (k = 1) and SVM top out accuracy with inclusion of the first 13 PCs.

One thing for further exploration would be to look at running a boosting algorithm using these 3 classifiers. MLE went above the 50% accuracy level at the inclusion of the 5th PC, while kNN (k = 1) and SVM went above the 50% accuracy level at the inclusion of the 4th PC. It's conceivable that a near perfect strong classifier could be achieved using 5 PC dimensions with a boosting algorithm.