Voice data

Large C => few (no) violations

Small C => many violations

Run and tune parameters on validations set, pick the best classifier and run on test data to get the final error rate

SVM

Linear

|  |  |
| --- | --- |
| C | Error on val set |
| 0.01 | 0.0197 |
| 0.05 | 0.0237 |
| 0.1 | 0.0375 |
| **0.5** | **0.0217** |
| 1 | 0.0237 |
| 10 | no convergence |
| 100 | no convergence |

Polynomial order 2

|  |  |
| --- | --- |
| C | Error on val set |
| 0.01 | 0.0276 |
| 0.05 | 0.0237 |
| **0.1** | **0.0276** |
| 0.5 | No convergence |
| 1 | No convergence |
| 10 | no convergence |
| 100 | no convergence |

Polynomial order 3

|  |  |
| --- | --- |
| C | Error on val set |
| 0.01 | 0.0355 |
| 0.05 | 0.0217 |
| 0.1 | 0.0256 |
| 0.5 | 0.0217 |
| 1 | 0.0375 |
| 10 | 0.0217 |
| **100** | **0.0178** |

RBF

|  |  |
| --- | --- |
| C | Error on val set |
| 0.01 | 0.2742 |
| 0.05 | 0.1144 |
| 0.1 | 0.0868 |
| 0.5 | 0.0276 |
| 1 | 0.0217 |
| 10 | 0.0256 |
| **100** | **0.0237** |

Multi-Layer Perceptron

|  |  |
| --- | --- |
| C | Error on val set |
| 0.01 | 0.5108 |
| 0.05 | 0.4162 |
| 0.1 | 0.4931 |
| 0.5 | 0.3688 |
| 1 | 0.3886 |
| 10 | 0.3964 |
| **100** | **0.3412** |

KNN

Distance - ‘seuclidean’ - Standardized Euclidean distance. Each coordinate difference between rows in X and the query matrix is scaled by dividing by the corresponding element of the standard deviation computed from X

There are about 10 variations of ways to calculate distance between the data points, running all of them, we found that using standardized euclidean distance versus any other variants gives up much smaller error, for example, comparing to euclidean distance:

|  |  |  |
| --- | --- | --- |
| K | ‘seuclidean’ - Error on validation set | ‘euclidean’ - Error on validation set |
| 1 | 0.8268 | 0.2840 |
| 3 | 0.1732 | 0.3748 |
| **5** | **0.0454** | 0.3905 |
| 7 | 0.0533 | 0.3688 |
| 9 | 0.0631 | 0.3826 |

Naive Bayes

Using the naive bayes we implemented, with assumption that the features are independent from each other. No parameters to tune so ran on entire training data set and then on test data, test error = 0.5118

Kernel Perceptron

C =1, using kernel perceptron we implemented, takes longer time to run comparing to previous methods

Linear, error rate = 0.3096 (0.4028)

Polynomial, order 2, error rate= 0.4850 (0.5213)

Polynomial, order 3, error rate = 0.5166 (0.5087)

**RBF, error rate = 0.4771 (0.2385)**

**MUSHROOM DATA**

SVM

Linear

|  |  |
| --- | --- |
| C | Error on val set |
| 0.01 | 0.0878 |
| 0.05 | 0.0947 |
| 0.1 | 0.1085 |
| 0.5 | 0.1001 |
| **1** | **0.1016** |
| 10 | no convergence |
| 100 | no convergence |

Polynomial order 2

|  |  |
| --- | --- |
| C | Error on val set |
| 0.01 | 0.0223 |
| 0.05 | 0.0239 |
| 0.1 | 0.0239 |
| 0.5 | 0.0239 |
| 1 | 0.0239 |
| 10 | 0.0239 |
| **100** | **0.0239** |

Polynomial order 3

|  |  |
| --- | --- |
| C | Error on val set |
| 0.01 | 0.0139 |
| 0.05 | 0.0139 |
| 0.1 | 0.0139 |
| 0.5 | 0.0139 |
| 1 | 0.0139 |
| 10 | 0.0139 |
| **100** | 0.0139 |

RBF

|  |  |
| --- | --- |
| C | Error on val set |
| 0.01 | 0.1732 |
| 0.05 | 0.0139 |
| 0.1 | 0.0139 |
| 0.5 | 0.0139 |
| 1 | 0.0139 |
| 10 | 0.0139 |
| **100** | 0.0139 |

**KNN**

Knn matlab gives only bad dummy results because the standardized Euclidean distance can not be computed properly in the presence of flat dimensions, and that’s what we get in mushroom data using standardized euclidean distance - ‘seuclidean’

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| K | ‘City block’ - Error on validation set | ‘euclidean’ - Error on validation set | ‘chebychev’ | ‘cosine’ |
| **1** | **0.0031** | **0.0139** | **0.0262** | **0.0139** |
| 3 | 0.0054 | 0.0400 | 0.0947 | 0.0439 |
| 5 | 0.0123 | 0.0600 | 0.0839 | 0.0631 |
| 7 | 0.0139 | 0.0801 | 0.0893 | 0.0793 |
| 9 | 0.0300 | 0.1024 | 0.0908 | 0.1001 |
| 51 | 0.3911 | 0.5166 | 0.6867 | 0.4942 |
| 101 | 0.3426 | 0.3172 | 0.7167 | 0.3580 |

**Naive bayes, error 0.3122**

**Kernel perceptron**

Linear, error rate = 0.0499

Polynomial, order 2, error rate= 0.0203

Polynomial, order 3, error rate = 0.0203

**RBF, error rate =** 0.0123