COMP0078 Assignment 2

Student Numbers: 21168615 & 19004608

Dec 14, 2022

1 PART I

1.1 Kernel Perceptron

1.1.1 Experimental Results

1. Basic Results:

	Train	Test
degree=1.0e+00	10.46%±1.28%	11.84%±1.72%
degree=2.0e+00	3.07%±0.94%	6.02%±1.15%
degree=3.0e+00	1.41%±0.40%	4.41%±0.55%
degree=4.0e+00	0.73%±0.15%	3.80%±0.37%
degree=5.0e+00	0.46%±0.13%	3.45%±0.44%
degree=6.0e+00	0.34%±0.08%	3.37%±0.48%
degree=7.0e+00	0.27%±0.10%	3.37%±0.33%

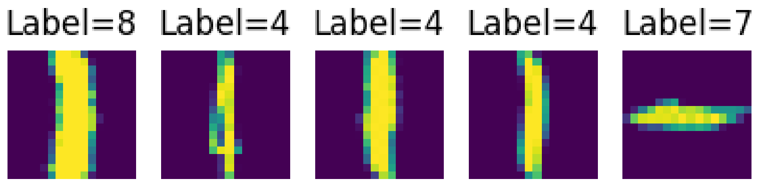
2. Cross Validation Results:

	Test Error Rate
Mean Optimal degree=6.0e+00±8.9e-01	3.38%±0.53%

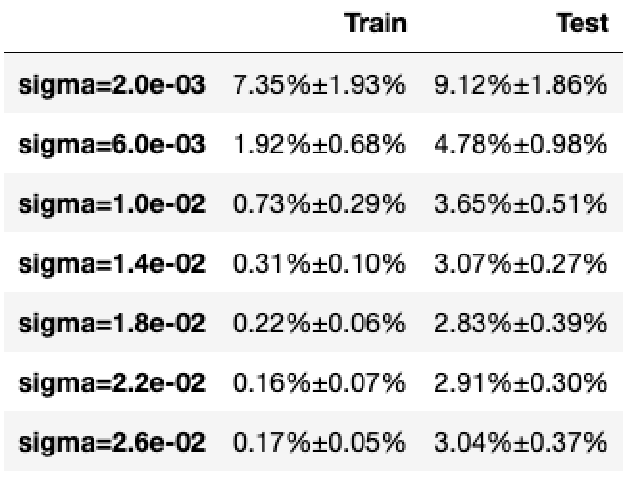
3. Confusion Matrix:

	0	1	2	3	4	5	6	
0	0.0%±0.0%	0.11%±0.0%	0.11%±0.0%	0.18%±0.0%	0.08%±0.0%	0.23%±0.0%	0.22%±0.0%	0.
1	0.0%±0.0%	0.0%±0.0%	0.06%±0.0%	0.04%±0.0%	0.19%±0.0%	0.0%±0.0%	0.08%±0.0%	0.11
2	0.43%±0.0%	0.37%±0.0%	0.0%±0.0%	0.68%±1.0%	0.57%±1.0%	0.13%±0.0%	0.16%±0.0%	0.80
3	0.35%±1.0%	0.34%±0.0%	0.69%±1.0%	0.0%±0.0%	0.06%±0.0%	1.8%±1.0%	0.1%±0.0%	0.45
4	0.12%±0.0%	1.34%±1.0%	1.1%±1.0%	0.26%±0.0%	0.0%±0.0%	0.36%±0.0%	0.6%±0.0%	0.36
5	0.75%±1.0%	0.07%±0.0%	0.35%±0.0%	1.12%±1.0%	0.17%±0.0%	0.0%±0.0%	0.65%±1.0%	0.
6	0.85%±1.0%	0.81%±1.0%	0.11%±0.0%	0.0%±0.0%	0.38%±0.0%	0.39%±1.0%	0.0%±0.0%	0.0
7	0.1%±0.0%	0.39%±1.0%	0.51%±1.0%	0.0%±0.0%	0.62%±1.0%	0.15%±0.0%	0.0%±0.0%	0.0
8	0.9%±1.0%	0.69%±1.0%	0.66%±1.0%	0.96%±1.0%	0.46%±1.0%	1.41%±1.0%	0.31%±1.0%	0.54
9	0.21%±0.0%	0.21%±0.0%	0.19%±0.0%	0.16%±0.0%	1.43%±1.0%	0.33%±0.0%	0.1%±0.0%	1.23

4. Hardest to predict images:

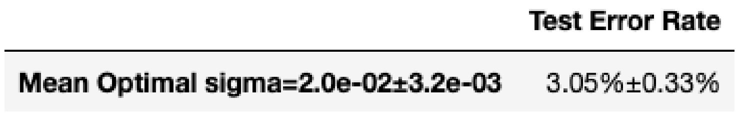


5. Basic Results:



	Train	Test
sigma=2.0e-03	7.35%±1.93%	9.12%±1.86%
sigma=6.0e-03	1.92%±0.68%	4.78%±0.98%
sigma=1.0e-02	0.73%±0.29%	3.65%±0.51%
sigma=1.4e-02	0.31%±0.10%	3.07%±0.27%
sigma=1.8e-02	0.22%±0.06%	2.83%±0.39%
sigma=2.2e-02	0.16%±0.07%	2.91%±0.30%
sigma=2.6e-02	0.17%±0.05%	3.04%±0.37%

Cross Validation Results:



	Test Error Rate
Mean Optimal sigma=2.0e-02±3.2e-03	3.05%±0.33%

1.1.2 Discussions

Choice of Parameters that weren't cross validated

Generalisation to k-classifiers

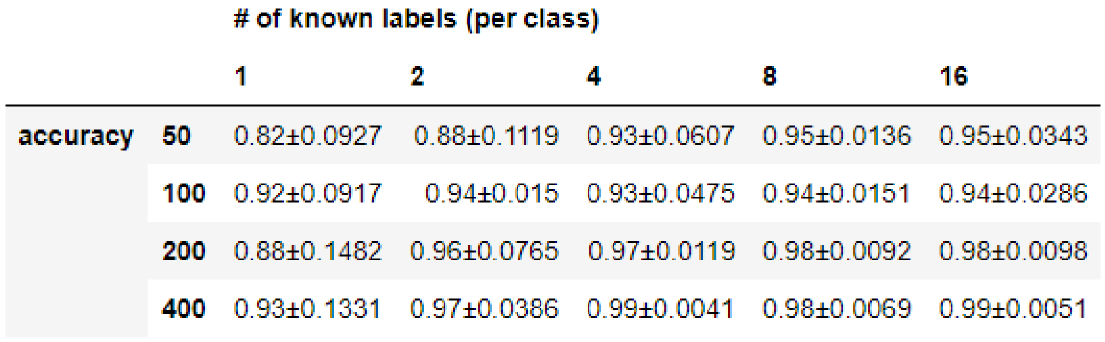
Kernel Comparison

Kernel Perceptron Implementation

2 PART II

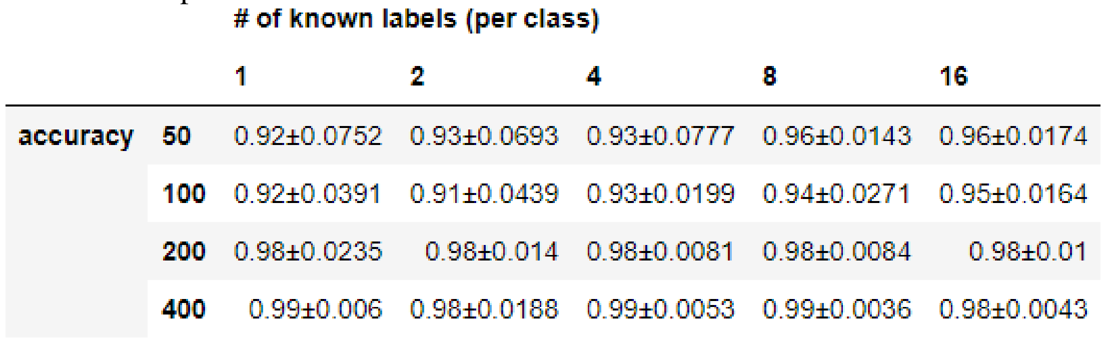
2.1 Semi-supervised Learning via Laplacian Interpolation

Experimental report for the laplacian interpolation approach:



		# of known labels (per class)				
		1	2	4	8	16
accuracy	50	0.82±0.0927	0.88±0.1119	0.93±0.0607	0.95±0.0136	0.95±0.0343
	100	0.92±0.0917	0.94±0.015	0.93±0.0475	0.94±0.0151	0.94±0.0286
	200	0.88±0.1482	0.96±0.0765	0.97±0.0119	0.98±0.0092	0.98±0.0098
	400	0.93±0.1331	0.97±0.0386	0.99±0.0041	0.98±0.0069	0.99±0.0051

And for the laplacian kernel method:



		# of known labels (per class)				
		1	2	4	8	16
accuracy	50	0.92±0.0752	0.93±0.0693	0.93±0.0777	0.96±0.0143	0.96±0.0174
	100	0.92±0.0391	0.91±0.0439	0.93±0.0199	0.94±0.0271	0.95±0.0164
	200	0.98±0.0235	0.98±0.014	0.98±0.0081	0.98±0.0084	0.98±0.01
	400	0.99±0.006	0.98±0.0188	0.99±0.0053	0.99±0.0036	0.98±0.0043

Some observations to make here are:

a) Both models seem to perform fairly well, even in the low-data setting.

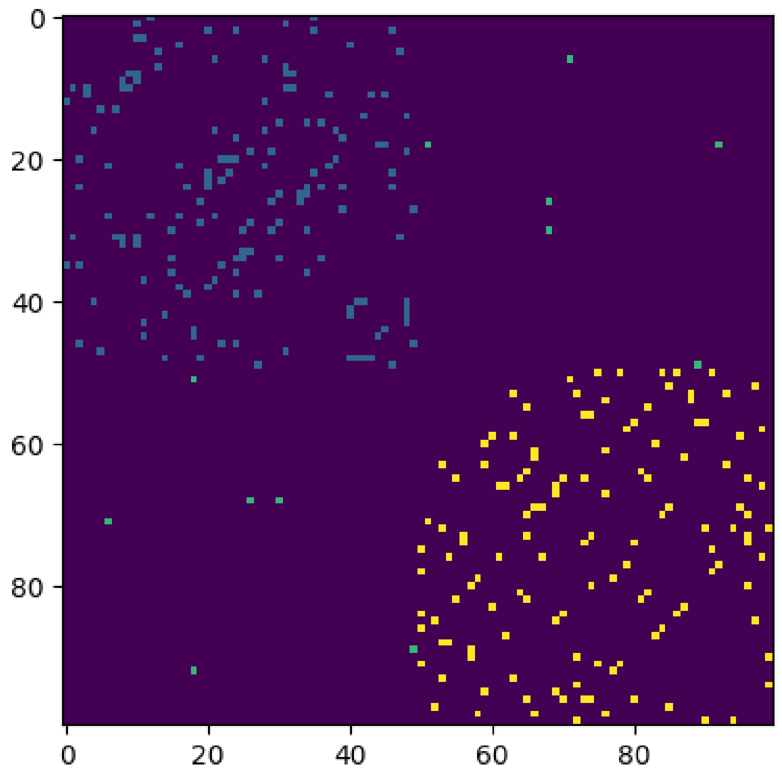
b) Increating the number of known labels increases the accuracy and decreases the variance of the predictors.

c) Increasing the total number of datapoints generally increases both model accuracies, and decreases variance.

d) The laplacian kernel method outperforms the vanilla inerpolation approach considerably, on both accuracy and variance.

The main reason for the success of this algorithm is the high degree of cluster separation observed in the dataset.

For illustration, here is a diagram representing the graph adjacency matrix, with colours showing the labelling of an edge. Here blue edges are where both labels are -1, teal when both are different, and yellow when both are +1.



We see through this that the vast majority of datapoints are connected only to points of the same labelling, and that the graph is highly separated into 2 clusters. Hence with minimal training information, both methods are able to predict with a high degree of accuracy.

Note that error variance decreases as a function of the number of known labels, but has a muhc more significant impacgt for the vanilla interpolation method.

We see that the kernel approach consistently outperforms the simple laplacian interpolation method. A reasonable explanation for this is that the laplacian interpolation approach utilises local information to ’diffuse’ labels through the graph. This means that only datapoints close to labelled data receive information from the labels themselves, and accuracy is likely reduced for datapoints far from the labels. In contrast, the kernel interpolation method takes global information from all the labelled datapoints and weights them according to their proximity and connectedness to eachother to predict labels.

3 PART III

3.1 Questions

a.

b.

c.

D. We note that since for any X_t , $y_t = X_{1,t}$, we have the following expression representing the linear separability of our dataset:

$$(v \cdot x_t)y_t \geq 1$$

Where $v = (1, 0, \dots, 0)^T$. Note that $\|v\| = 1$.
Hence, in our online mistake bound for the perceptron, we have that $\gamma = 1$. Further note that $\forall t$, $x_t \in -1, 1^n \implies \|x_t\|^2 = n$. This gives us that $R = max_t\|x_t\| = \sqrt{n}$. Hence our mistake bound for the perceptron algorithm is given by:

$$M \leq n$$

.

Using the theorem on page 60 of the online learning notes, we arrive at:

$$Prob(\mathcal{A}_S(x') \neq y') \leq \frac{n}{m}$$

E. From our experimental observations we may expect the sample complexity of 1-NN to be lower bounded by some exponential function. We formalise this in the following proposition:

Proposition:

Following the data-generating distribution described above, we have that our sample complexity given by:

$m(n) = \Omega(n)$
Proof:

Suppose we sample a training set $S = \{(x_1, y_1), \dots, (x_n, y_n)\}$ uniformly from the set $\{-1, 1\}^n$, with associated labels defined by the rule $y|x = x_1$, and use this training set for inference on an arbitrary test point (x, x_1), sampled uniformly from the same set.

We note the following observation:

Obs:
if $x \in S$, then $\mathcal{A}_S(x)) = y$, where y is the true label for x.

To justify this, observe that our dataset represents a realiseable learning problem, and as such, if $(x, y), (x\prime, y\prime)$ are datapoints sampled from our distribution, $x = x\prime \implies y = y\prime$. Trivially, x is the 1-nearest neighbour to itself, so the algorithm makes a correct prediction for any datapoint present in our training set.

Hence, $\mathcal{A}_S$ makes an error on x $\implies x \notin S$.

Hence, the set of all training sets that make an error on x is contained in the set of all training sets not containing x.

Hence, for a given x, $P_S(\mathcal{A}_S(x) \neq y) \leq P_S(x \notin S)$.

Since S is a collection of points sampled iid from our data generating distribution,
$P_S(x \notin S) = \prod_{i=1}^{m} P(x_i \neq x) = \prod_i (1 - P(x_i = x)) = (1 - 2^{-n})^m$

We note that our choice of x was arbitrary, and since we sampled x uniformly, we arrive at the following generalisation error bound:

$\mathbb{E}_{S \sim \mathcal{D}^m, x \sim \mathcal{D}}[\mathcal{L}_S(x)] \leq (1 - 2^{-n})^m$
Using the identity $1 - x \leq e^{-x}$ provided frequently in the notes, we simplify this bound to give:

$\mathbb{E}_{S \sim \mathcal{D}^m, x \sim \mathcal{D}}[\mathcal{L}_S(x)] \leq exp(-2^n m)$
Suppose we seek m such that our generalisation error is less than some fixed ϵ.

Hence we require $\mathbb{E}_{S \sim \mathcal{D}^m, x \sim \mathcal{D}}[\mathcal{L}_S(x)] \leq exp(-2^n m) \leq \epsilon$

$\implies -m2^{-n} \leq log(\epsilon)$

Hence, $m \geq -log(\epsilon)2^n = log(\frac{1}{\epsilon})2^n$

$\implies m = \Omega(2^n)$
$\square$