Jiani Lin

1 Warm up: Support Vector Machines

1. Denote the first column as x_1 and the second column as x_2. The hyperplane with the maximum margin is the plane $x_2 = 2$. $[1, 0] \in \mathcal{D}_-$ and $[3, 1]$, $[3, -1] \in \mathcal{D}_+$ are support vectors.

2. Yes, my SVM can classify this point correctly.

3. No, the Perceptron algorithm cannot guarantee classify that point correctly, because this algorithm will update only when making mistake and if we add point $[1.999, 1]$ in to $\mathcal{D}_+$ would not make training set become linearly inseparable, which means it is possible that the algorithm make no mistake on training set but still cannot classify this point correctly. For instance , in the hypothesis space $x_2 = a$, $a \in R$, when $a \in (1, 3)$, the algorithm will make no mistake on the training set, but the point can be classified correctly only when $a \in (1.999, 2)$.

2 Kernels and the Perceptron algorithm

1. After transforming $\phi(\mathbf{x})$ that maps examples to a new space of k-conjunctions, the target concept become disjunctions which is linearly separable. Assume there are n terms in a k-DNF, the examples can be separated by $\sum_{i=1}^{n} \phi_i(\mathbf{x}) \geq 1$.

2. since the conjunction takes exactly k literals,we have

$$K(\mathbf{x}_1, \mathbf{x}_2) = \phi(\mathbf{x}_1)^T \phi(\mathbf{x}_2) = \sum_{c \in C} c(\mathbf{x}_1) c(\mathbf{x}_2) = \binom{same(\mathbf{x}_1, \mathbf{x}_2)}{k}$$

   which means choose k from the number of the same bits in $\mathbf{x}_1$ and $\mathbf{x}_2$.

3. In percetron algorithm, w update only when the algorithm make mistake. According to the algorithm, after learning make mistakes M, we have $\mathbf{w} = \mathbf{w}_0 + r \sum_{(x_i, y_i) \in M} y_i \phi(x_i)$. Considering $w_0 = 0$ and the prediction is based on the sign of $w\phi(x)$, which means the magnitude of the w is irrelevant. Hence, we have $\mathbf{w} = \sum_{(\mathbf{x}_i, y_i) \in M} y_i \, \phi(x_i)$

4. Since $\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \phi(\mathbf{x}_i)$, we have:

$$\begin{aligned} w^T \phi(\mathbf{x}) &= \sum_{i=1}^{m} \alpha_i y_i \phi^T(\mathbf{x}_i) \phi(\mathbf{x}) \\ &= \sum_{i=1}^{m} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) \\ &= \sum_{i=1}^{m} \alpha_i y_i \binom{same(\mathbf{x}_1, \mathbf{x}_2)}{k} \end{aligned}$$

where the last step getting from question 2. The prediction:

$$\begin{aligned} y &= sgn\left(\sum_{i=1}^{m} \alpha_i y_i \binom{same(\mathbf{x}_1, \mathbf{x}_2)}{k}\right) \\ &= sgn\left(\sum_{(x_i, y_i \in M)} y_i \binom{same(\mathbf{x}_1, \mathbf{x}_2)}{k}\right) \end{aligned}$$

where, M is the set of examples on which the learning algorithm made mistakes. In the last step, $\alpha_i = C$ if $(xi, yi) \in M$, and we only care about the sign of the equation.

5. the kernel Perceptron is as follows, where α is the mistake place storing.

Algorithm 1 Construct linear function

$\alpha = 0$, where $\alpha \in \mathbb{R}^m$
$m =$ the number of examples in the training set
for $i \in [2, m]$ **do**
  if $y_i \sum_{j=1}^{i-1} \alpha_j y_j K(\mathbf{x}_j, \mathbf{x}_i) \leq 0$ **then**
    $\alpha_i = \alpha_i + 1$
  end if
end for
return α

3 Experiment: Training an SVM classifier

1. Running svm on original dataset with $\rho_0 = 0.01$ and $C = 0.1$, we can predict test examples with the accuracy of 78.10%.

   Running svm on scaled dataset with $\rho_0 = 0.01$ and $C = 10$, we can predict test examples with the accuracy of 77.58%.

   Running svm on data0 with $\rho_0 = 0.01$ and $C = 50$, we can predict test examples with the accuracy of 100%.

2. The farthest distances for each four cases are shown in Table **??**.

	original	scaled	original transformed	scaled transformed
training	596.4621	1.9427	3.6096×10^5	3.4724
test	575.8904	1.9490	3.3279×10^5	3.4683
farthest	596.4621	1.9490	3.6096×10^5	3.4724

Table 1: The distance of the farthest data

3. Running 10-fold cross validation on original, scaled, original.transformed, scaled.transformed and data0 set, and using the best hyperparameters to predict test data in each case, we can get the accuracy of prediction as Table **??**.

	original	scaled	original transformed	scaled transformed	data0
accuracy	79.43%	78.05%	90.87%	94.00%	100%
margin	13.5134	0.5399	4.1751	0.2465	0.2132

Table 2: The accuracy of predicting data in test set with best hyperparameters

When doing cross validation, on data0, the combination of $\rho_0 \in \{0.001, 0.01, 0.1, 1\}$ and $C \in \{0.001, 0.01, 0.1, 1, 10, 20, 50, 100\}$ have been tried. The result is as Table **??**. As we can see, the best hyperparameters are in seventh, $\rho_0 = 0.001$ and $C = 50$. The accuracy in cross validation and the test set are both 1, which means data in data0 can be separated linearly.

In original set, we have tried the combination of $\rho_0 \in \{0.001, 0.01, 0.1, 1\}$ and $C \in \{0.001, 0.01, 0.1, 0.5, 0.8, 1, 10\}$. The result is as Table **??**. The the twelfth hyperparameters are the best, which is $\rho_0 = 0.01$ and $C = 0.8$. The accuracy in cross validation is 84.14% and in the test set is 79.43%.

In the original.transformed set, we have tried the combination of $\rho_0 \in \{0.001, 0.01, 0.1, 1\}$ and $C \in \{0.001, 0.01, 0.1, 1, 10\}$. The result is as Table **??**. The the forth hyperparameters are the best, which is $\rho_0 = 0.001$ and $C = 1$. The accuracy in cross validation is 84.98% and in the test set is 90.87%.

In scaled set, we have tried the combination of $\rho_0 \in \{0.001, 0.01, 0.1, 1\}$ and $C \in \{0.001, 0.01, 0.1, 1, 10, 12, 15\}$. The result is as Table **??**. The the thirteenth hyperparameters are the best, which is $\rho_0 = 0.01$ and $C = 12$. The accuracy in cross validation is 83.49% and in the test set is 78.05%.

In scaled.transformed set, we have tried the combination of $\rho_0 \in \{0.001, 0.01, 0.1, 1\}$ and $C \in \{0.001, 0.01, 0.1, 1, 10, 70, 100\}$. The result is as Table **??**. The the sixth hyperparameters are the best, which is $\rho_0 = 0.001$ and $C = 70$. The accuracy in cross validation is 91.74% and in the test set is 94.00%.

Discussed with Yi Ou.

ordering	ρ_0	C	accuracy	ordering	ρ_0	C	accuracy
1	0.0010	0.0010	0.5250	17	0.1000	0.0010	0.4680
2	0.0010	0.0100	0.4920	18	0.1000	0.0100	0.5240
3	0.0010	0.1000	0.4920	19	0.1000	0.1000	0.4920
4	0.0010	1.0000	0.4920	20	0.1000	1.0000	0.4920
5	0.0010	10.0000	0.8550	21	0.1000	10.0000	0.7560
6	0.0010	20.0000	0.9990	22	0.1000	20.0000	0.9690
7	0.0010	50.0000	1.0000	23	0.1000	50.0000	0.9350
8	0.0010	100.0000	0.9950	24	0.1000	100.0000	0.7500
9	0.0100	0.0010	0.4840	25	1.0000	0.0010	0.5200
10	0.0100	0.0100	0.5120	26	1.0000	0.0100	0.5080
11	0.0100	0.1000	0.4920	27	1.0000	0.1000	0.4800
12	0.0100	1.0000	0.4920	28	1.0000	1.0000	0.4920
13	0.0100	10.0000	0.7720	29	1.0000	10.0000	0.8970
14	0.0100	20.0000	0.9800	30	1.0000	20.0000	0.9910
15	0.0100	50.0000	0.9840	31	1.0000	50.0000	0.9310
16	0.0100	100.0000	0.8500	32	1.0000	100.0000	0.6740

Table 3: Running cross validation on data0

ordering	ρ_0	C	accuracy	ordering	ρ_0	C	accuracy
1	0.0010	0.0010	0.6475	15	0.1000	0.0010	0.6193
2	0.0010	0.0100	0.6533	16	0.1000	0.0100	0.6487
3	0.0010	0.1000	0.8352	17	0.1000	0.1000	0.8291
4	0.0010	0.5000	0.8297	18	0.1000	0.5000	0.8352
5	0.0010	0.8000	0.8271	19	0.1000	0.8000	0.8223
6	0.0010	1.0000	0.8242	20	0.1000	1.0000	0.7990
7	0.0010	10.0000	0.7747	21	0.1000	10.0000	0.7527
8	0.0100	0.0010	0.6701	22	1.0000	0.0010	0.5892
9	0.0100	0.0100	0.7271	23	1.0000	0.0100	0.5371
10	0.0100	0.1000	0.8368	24	1.0000	0.1000	0.6999
11	0.0100	0.5000	0.8394	25	1.0000	0.5000	0.8304
12	0.0100	0.8000	0.8414	26	1.0000	0.8000	0.8336
13	0.0100	1.0000	0.8194	27	1.0000	1.0000	0.8239
14	0.0100	10.0000	0.7753	28	1.0000	10.0000	0.7750

Table 4: Running cross validation on original

ordering	ρ_0	C	accuracy	ordering	ρ_0	C	accuracy
1	0.0010	0.0010	0.7928	11	0.1000	0.0010	0.5918
2	0.0010	0.0100	0.8346	12	0.1000	0.0100	0.6970
3	0.0010	0.1000	0.8498	13	0.1000	0.1000	0.8304
4	0.0010	1.0000	0.8252	14	0.1000	1.0000	0.8278
5	0.0010	10.0000	0.8129	15	0.1000	10.0000	0.8051
6	0.0100	0.0010	0.6825	16	1.0000	0.0010	0.5280
7	0.0100	0.0100	0.7912	17	1.0000	0.0100	0.5507
8	0.0100	0.1000	0.8336	18	1.0000	0.1000	0.6970
9	0.0100	1.0000	0.8190	19	1.0000	1.0000	0.8161
10	0.0100	10.0000	0.8213	20	1.0000	10.0000	0.7941

Table 5: Running cross validation on original.transformed

ordering	ρ_0	C	accuracy	ordering	ρ_0	C	accuracy
1	0.0010	0.0010	0.6691	15	0.1000	0.0010	0.5918
2	0.0010	0.0100	0.6475	16	0.1000	0.0100	0.5724
3	0.0010	0.1000	0.6475	17	0.1000	0.1000	0.6568
4	0.0010	1.0000	0.6475	18	0.1000	1.0000	0.6475
5	0.0010	10.0000	0.8271	19	0.1000	10.0000	0.8258
6	0.0010	12.0000	0.8300	20	0.1000	12.0000	0.8278
7	0.0010	15.0000	0.8300	21	0.1000	15.0000	0.8297
8	0.0100	0.0010	0.4857	22	1.0000	0.0010	0.5808
9	0.0100	0.0100	0.6423	23	1.0000	0.0100	0.5571
10	0.0100	0.1000	0.6491	24	1.0000	0.1000	0.5111
11	0.0100	1.0000	0.6475	25	1.0000	1.0000	0.6475
12	0.0100	10.0000	0.8307	26	1.0000	10.0000	0.8278
13	0.0100	12.0000	0.8349	27	1.0000	12.0000	0.8226
14	0.0100	15.0000	0.8336	28	1.0000	15.0000	0.8271

Table 6: Running cross validation on scaled

ordering	ρ_0	C	accuracy	ordering	ρ_0	C	accuracy
1	0.0010	0.0010	0.7170	15	0.1000	0.0010	0.5552
2	0.0010	0.0100	0.7093	16	0.1000	0.0100	0.5585
3	0.0010	0.1000	0.7044	17	0.1000	0.1000	0.7261
4	0.0010	1.0000	0.7044	18	0.1000	1.0000	0.7057
5	0.0010	10.0000	0.8666	19	0.1000	10.0000	0.8750
6	0.0010	70.0000	0.9174	20	0.1000	70.0000	0.8867
7	0.0010	100.0000	0.9116	21	0.1000	100.0000	0.8783
8	0.0100	0.0010	0.5341	22	1.0000	0.0010	0.5370
9	0.0100	0.0100	0.7501	23	1.0000	0.0100	0.6057
10	0.0100	0.1000	0.7303	24	1.0000	0.1000	0.5983
11	0.0100	1.0000	0.7057	25	1.0000	1.0000	0.7057
12	0.0100	10.0000	0.8676	26	1.0000	10.0000	0.8699
13	0.0100	70.0000	0.8903	27	1.0000	70.0000	0.8656
14	0.0100	100.0000	0.8896	28	1.0000	100.0000	0.8822

Table 7: Running cross validation on scaled.transformed