EE5904/ME5404
Neural Network

Kernel SVM Report

Wang Jiangyi, A0236307J, Mathematics Department
National University of Singapore
E0732857@U.NUS.EDU
April 6, 2022

1 Details of Implementation (Framework)

1.1 Pre-processing

In this experiment, we propose 2 Methods of Pre-processing:

1. For each sample, re-scale to $||x_i|| = 1 \quad i = 1, 2, ..., N$;
2. For each feature, normalize to mean 0 and variance 0.01.

1.2 Hard-margin SVM

To approximate Hard-margin SVM, we choose a Soft-margin SVM with $C = 1e4$. Through experiments, we find that, with some bigger C, i.e., $1e5$, the computation cost will increase dramatically while solving the dual-form Quadratic Programming. Therefore, we choose to use $C = 1e4$ to achieve the approximation.

1.3 Soft-margin SVM

We choose different value of C and implement the Soft-margin SVM algorithm. The result is:

1. For standardization dataset, a moderate order p of kernel and relatively big choice of C will lead to better generalization ability;

2. For rescale dataset, a moderate order p of kernel and extremely big (hard-margin) choice of C will lead to better generalization ability.

In the following discussion (Section 3), we will illustrate the reason from the effective dimension for dataset after pre-processing.

1.4 RBF Kernel

To design our own Kernel-SVM, we choose RBF Kernel to test its performance. Firstly, we analyze the influence of hyper-parameter γ theoretically. Then, based on our theoretical observation, we design experiments for different choice of γ and C in order to find the optimal ones. Moreover, through experiments, we also show that our previous theoretical observation is reasonable.

1.5 Evaluation

Here, based on all previous experiments, we choose the Kernel-SVM which gives us the best testing accuracy, that is:

1. Soft margin SVM with $C = 2.1$;

2. Polynomial Kernel that $p = 2$;

3. Standardization Data Pre-processing.

2 Task 1: Solve for discriminant function

Here, we apply two methods of **pre-processing** the data:

1. For each sample, re-scale to $||x_i|| = 1 \quad i = 1, 2, ..., N$;
2. For each feature, normalize to mean 0 and variance 0.01 (to avoid numerical problems).

Motivation for 2: We want our optimzation problem to be isotropic, which will be benifical for convergence of algorithm. This can be achieved by normalizing by each feature. Notice, here we actually choose variance 0.01 instead of 1 for the sake of **high-ordered polynomial kernel**, which we will discuss later.

Discussion Part:

1. Data-preprocessing: (this part has great connection with Admissibilty of Kernel)

Re-scale: For this method, all samples will have the same length 1 and each component will be proportional to the original samples. Notice that our original data samples have the property that, in very few features, it has very large value. Therefore, this method **naturally achieves the feature extraction**. Also, since each sample has same length 1, **kernel matrix tends to be well-conditioned**, which is a good property (our dual optimization problem is highly related to Kernel Matrix).

However, with such method, **effective dimension of each sample greatly shrinks**. Therefore, we may encounter **under-fitting with simple model**.

Normalize: For this method, if we choose **variance 1**, something bad will happen for high-ordered polynomial kernel. That is, $K(x_i, x_j) = (1 + \langle x_i, x_j \rangle)^p$ might be very very big, i.e., **1e20** approximately for $p = 4$. For this case, **all our algorithms for optimization will break down (including eigenvalue solver)**.

Therefore, I choose **mean 0 and variance 0.01** to pre-process the data. For this case, when $p = 4$, $K(x_i, x_j)$ is **around 1**, which is very acceptable for Kernel Matrix. Moreover, all features are isotropic, which is great for optimization algorithms.

2. Admissibility of kernels:

Statement: In principle, these 5 kernels are all admissible (**for arbitrary dataset**).

Supporting argument: Given dataset $\mathbf{D} = \{x^i\}_{i=1}^N, x^i \in \mathbb{R}^d$. Let's consider $p = 3$ case for simplicity.

$$\begin{aligned}
K(x, z) &= (1 + x_1 z_1 + ... + x_d z_d)^3 \\
&= 1 + x_1^3 z_1^3 + ... + x_d^3 z_d^3 + 3\sum_i x_i z_i + 3\sum_i x_i^2 z_i^2 + 3\sum_{i \neq j} x_i^2 x_j z_i^2 z_j + 3\sum_{i \neq j \neq k} x_i x_j x_k z_i z_j z_k \\
&= (1, x_1^3, ..., x_d^3, \sqrt{3}x_1, ..., \sqrt{3}x_d, \sqrt{3}x_1^2, ..., \sqrt{3}x_d^2, \sqrt{3}x_1^2 x_2, ..., \sqrt{3}x_d^2 x_{d-1}, \sqrt{3}x_1 x_2 x_3, ..., \sqrt{3}x_{d-2}x_{d-1}x_d) \\
&\times (1, z_1^3, ..., z_d^3, \sqrt{3}z_1, ..., \sqrt{3}z_d, \sqrt{3}z_1^2, ..., \sqrt{3}z_d^2, \sqrt{3}z_1^2 z_2, ..., \sqrt{3}z_d^2 z_{d-1}, \sqrt{3}z_1 z_2 z_3, ..., \sqrt{3}z_{d-2}z_{d-1}z_d)^T \\
&:= \langle \phi(x), \phi(z) \rangle
\end{aligned}$$

Since $K(\star, \star)$ is induced by $\langle \phi(\star), \phi(\star) \rangle$, Kernel Matrix $\mathbf{K}$ must be PSD, **satisfying Mercer's condition**.

However, **in experiment**, if in our dataset, $||x^i||$ is very big, then Kernel Matrix (especially polynomial kernel) will be very ill-conditioned. It is difficult to check its eigenvalue because all eigenvalue numerical algorithms will break down. Therefore, we should **choose appropriate methods for data pre-processing**.

Also, in experiment, with both 2 methods of pre-processing, it shows that **all kernels are admissibile**.

3. Existence of optimal hyperplanes:

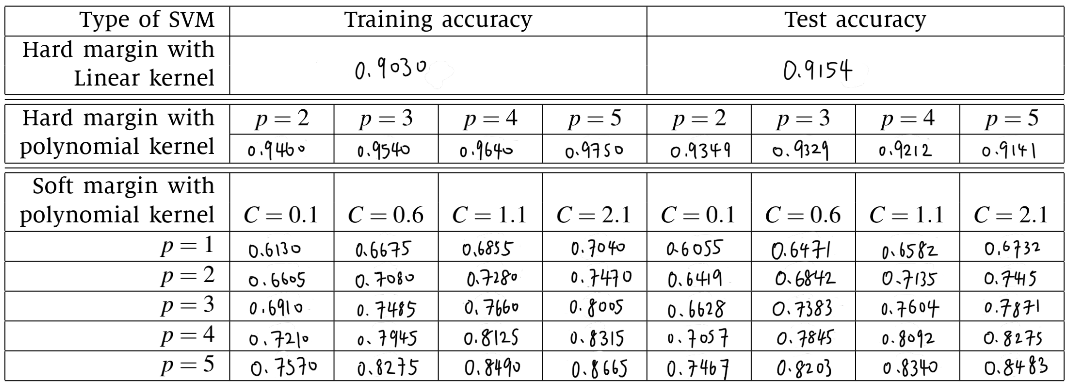
Statement: In principle, for **hard-margin SVM**, the existence of optimal hyperplane is equivalent to linear-separability in feature space; for **soft-margin SVM**, optimal hyperplane always exists.

Supporting argument: For hard-margin SVM case, if dataset is nonlinearly-separable, then the feasible region of Primal Form of SVM is empty set. Therefore, we cannot attain the optimal (w_o, b_o), implying that optimal hyperplane does not exist; However, for soft-margin SVM case, the feasible region of Primal Form of SVM is always non-empty (because the existence of slack variable ξ_i). Therefore, the optimal (w_o, b_o) always exists, implying that optimal hyperplane always exists.

However, **numerically**, we can **approximate** hard-margin SVM by soft-margin SVM **with large C**, i.e., 1e5. By this method, we can also find an optimal hyperplane for approximated hard-margin SVM. This is what we do in the following experiments.

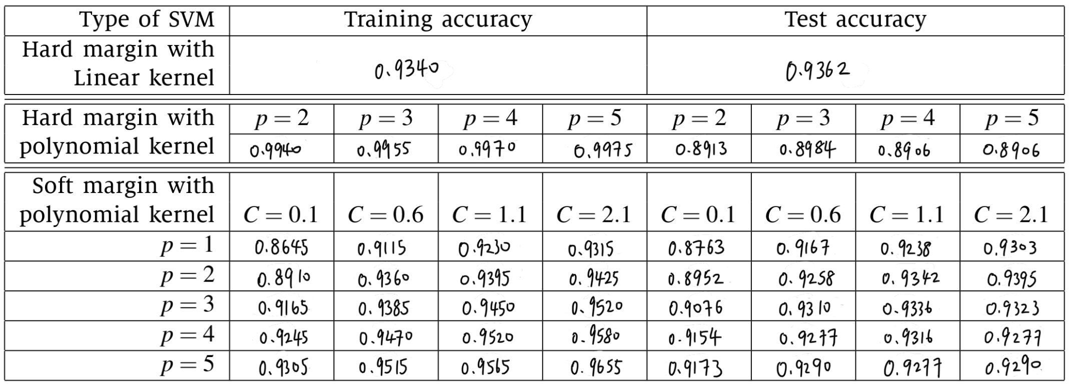
3 Task 2: Classification Result and Discussion

Here, we implement our SVM on both two pre-processed data. To approximate hard-margin SVM, we use soft-margin SVM with C**=1e4.** The results table is shown as follows:



Type of SVM	Training accuracy				Test accuracy			
Hard margin with Linear kernel	0.9030				0.9154			
Hard margin with polynomial kernel	$p=2$	$p=3$	$p=4$	$p=5$	$p=2$	$p=3$	$p=4$	$p=5$
	0.9460	0.9540	0.9640	0.9750	0.9349	0.9329	0.9212	0.9141
Soft margin with polynomial kernel	$C=0.1$	$C=0.6$	$C=1.1$	$C=2.1$	$C=0.1$	$C=0.6$	$C=1.1$	$C=2.1$
$p=1$	0.6130	0.6675	0.6855	0.7040	0.6055	0.6471	0.6582	0.6732
$p=2$	0.6605	0.7080	0.7280	0.7470	0.6419	0.6842	0.7135	0.7415
$p=3$	0.6910	0.7485	0.7660	0.8005	0.6628	0.7383	0.7604	0.7871
$p=4$	0.7210	0.7945	0.8125	0.8315	0.7057	0.7845	0.8092	0.8275
$p=5$	0.7570	0.8275	0.8490	0.8665	0.7467	0.8203	0.8340	0.8483

Figure 1: Rescale Table



Type of SVM	Training accuracy				Test accuracy			
Hard margin with Linear kernel	0.9340				0.9362			
Hard margin with polynomial kernel	$p=2$	$p=3$	$p=4$	$p=5$	$p=2$	$p=3$	$p=4$	$p=5$
	0.9940	0.9955	0.9970	0.9975	0.8913	0.8984	0.8906	0.8906
Soft margin with polynomial kernel	$C=0.1$	$C=0.6$	$C=1.1$	$C=2.1$	$C=0.1$	$C=0.6$	$C=1.1$	$C=2.1$
$p=1$	0.8645	0.9115	0.9230	0.9315	0.8763	0.9167	0.9238	0.9303
$p=2$	0.8910	0.9360	0.9395	0.9425	0.8952	0.9258	0.9342	0.9395
$p=3$	0.9165	0.9385	0.9450	0.9520	0.9076	0.9310	0.9336	0.9323
$p=4$	0.9245	0.9470	0.9520	0.9580	0.9154	0.9277	0.9316	0.9277
$p=5$	0.9305	0.9515	0.9565	0.9655	0.9173	0.9290	0.9277	0.9290

Figure 2: Normalize Table

Dicussion Part:

1. From Figure 1 and 2, we can observe that, Method 2 can fit our training data better than Method 1 because **all training accuracy is higher**. This **corresponds to our previous discussion of pre-processing**:

For Method 1, effective dimension of each training sample shrinks greatly. Therefore, it will be more difficult to make correct classification for training set.

However, for Method 2, after normalization, each feature is isotropic. Therefore, after feature map, the training data is more likely to be linearly-separable (or the separability is better) in feature space because **training accuracy is almost 1 for hard-margin SVM** when $p \geq 2$.

2. Let's focus on hard-margin SVM for 2 Methods:

For Method 1, the training accuracy and test accuracy is relatively close. This can be one evidence that, for this case, the SVM is not over-fitting. **Therefore, actually we should choose some big value of** C instead of small one, because our model needs more penalty to fit well.

This is a reasonable conclusion because, in Figure 1, when we decrease C, the performance of our model becomes worse. Therefore, **the experience is, when training accuracy is relatively close to test accuracy, we should not decrease** C **too much because this is likely to lead to under-fitting.**

For Method 2, when $p = 1$, the training accuracy is very close to test accuracy. Therefore, we can deduce that this SVM is not over-fitting. However, when $p = 2, 3, 4, 5$, the training accuracy is around 1 and the test accuracy is only 0.9. Therefore, our model may be **over-fitting**. We need to decrease C so that we can have a large margin and better generalization ability.

Also, from Figure 2, we can notice that this method works. When we **decrease C to around 2**, training accuracy and test accuracy for each p becomes very close, aprrroximately 0.94. This shows that the **over-fitting problem has been released**. Actually this is an extremely good result because we can **attain 0.9395 testing accuracy by normalizing dataset and setting $p = 2, C = 2.1$.**

The **experience is, when training accuracy is high but test accuracy is low, we can try to decrease C to attain better generalization ability.**

From above discussion, as for this SPAM classification problem, we **prefer to use Method 2** to pre-process the data.

3. From Figure 1 and 2, it can be observed that with large value of p, the **training accuracy must increase**. However, the conclusion **does not hold** for test accuracy.

Supporting argument: Recall the feature map for $p = 3$ in **Task 1 Disccusion**:

$$\phi_3(x) = (1, x_1^3, ..., x_d^3, \sqrt{3}x_1, ..., \sqrt{3}x_d, \sqrt{3}x_1^2, ..., \sqrt{3}x_d^2, \sqrt{3}x_1^2x_2, ..., \sqrt{3}x_d^2x_{d-1}, \sqrt{3}x_1x_2x_3, ..., \sqrt{3}x_{d-2}x_{d-1}x_d)$$

where $x \in \mathbb{R}^d$

Notice that, this feature mapping $\phi_3(\star)$ actually includes all 1-order and 2-order term, i.e., $\{1, x_i, x_ix_j, x_i^2\}$. Therefore, if we set those weights of 3-order term to 0, this feature map is reduced to a 2-order polynomial feature map $\phi_2(\star)$.

That is to say, **from the perspective of discriminant function**, low-order polynomial feature map is included in high-order polynomial feature map. Or we can say, low-order polynomial feature map can be viewed as one special case of high-order polynomial feature map. Then, it is obvious that, with large value of p, the training accuracy will definitely increase since **our hypothesis of discriminant fucntion (candidates) is larger**. The result is shown as follows:

Hard margin with polynomial kernel	$p = 2$	$p = 3$	$p = 4$	$p = 5$
	0.9460	0.9540	0.9640	0.9750

Hard margin with polynomial kernel	$p = 2$	$p = 3$	$p = 4$	$p = 5$
	0.9940	0.9955	0.9970	0.9975

Figure 3: Training Accuracy for different p

However, when we consider the test accuracy, it is not always the case. The result is shown in Figure 4:

Hard margin with polynomial kernel	$p = 2$	$p = 3$	$p = 4$	$p = 5$
	0.9349	0.9329	0.9212	0.9141

Hard margin with polynomial kernel	$p = 2$	$p = 3$	$p = 4$	$p = 5$
	0.8913	0.8984	0.8906	0.8906

Figure 4: Test Accuracy for different p

In Figure 4, it can be observed that, with large value of p, the test accuracy still may decrease. The story is, our ultimate goal is to achieve better generalization ability, not to classify all training samples correctly. For large p, we will easily over-fit, which leads to high training accuracy and low test accuracy.

4. Admissibility of kernel and existence of optimal hyperplane.

As we stated in Task 1 Discussion, all these kernels are admissible because we can construct an feature map $\phi(\star)$ for each kernel. Therefore, Kernel Matrix $\mathbf{K}$ must be PSD, satisfying Mercer's condition.

Also, in principle, optimal hyperplanes do not exist for hard-margin SVM (at least for Method 1) because with $C = 1e4$, its training accuracy is still at most 0.95. Therefore, the data samples are very likely to be nonliearly-separable in feature space.

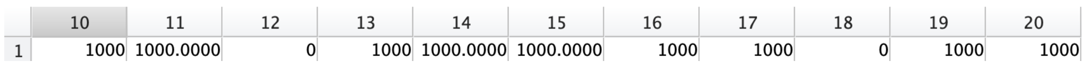
Moreover, optimal hyperplanes must exist for soft-margin SVM.

4 Task 3: Design own Kernel SVM

Here, I choose 1 more kernel: RBF Kernel: $K(x_i, x_j) = exp(-\gamma||x_i - x_j||^2)$, where γ is the hyperparameter.

Discussion Part:

1. Observation: with extremely small value of γ, i.e., 1e-6, most entries of the optimal dual solution α_o will be C (penalty coefficient). Part of the results can be seen as follows:



	10	11	12	13	14	15	16	17	18	19	20
1	1000	1000.0000	0	1000	1000.0000	1000.0000	1000	1000	0	1000	1000

Figure 5: 10-20 entries of α_o when C=1000

Supporting argument: It is obvious to see that, now $K(x_i, x_j) \equiv 1$. Let's consider one **extreme** case: **number of postive class = number of negative class**. Now, the dual problem is:

$$min: \frac{1}{2}\sum_{s,t=1}^{N} \alpha_t\alpha_s d_t d_s - \sum_{t=1}^{N}\alpha_t \equiv -\sum_{t=1}^{N}\alpha_t$$

$$s.t. \quad \alpha_t \in [0, C], \quad \sum_{t=1}^{N}\alpha_t d_t = 0$$

Based on our extreme case assupmtion, it is easy to check that, $\alpha_t^o \equiv C, \forall t$ is the optimal dual solution.

Using KKT Complementary Slackness Condition, we can interpret this as, all samples are Non-margin Support Vectors (classified wrongly).

For general case, using simliar technique, we can show that, **most** of samples will be Non-margin SVs (classified wrongly). This can be one theoretical evidence for: **small γ leads to under-fitting.**

2. Observation: with extremly large value of γ, i.e., 1e6, we will definitely achieve 100% training accuracy.

Supporting argument: Now, $K(x_i, x_j) = \delta_{ij}$. We can **construct a feature map**, $\phi(x_i) = e_i$ (e_i represents the i-th column of Identity Matrix $\mathbf{I}_N$). We can choose the weight vector $\hat{w} = (d_1, ..., d_N)^T$ in the feature space to separate these two classes because:

$$d_i \langle \hat{w}, e_i \rangle = d_i^2 \equiv 1 > 0$$

This shows that, in feature space, these two classes are **linearly separable**. Therefore, we will **definitely achieve 100% training accuracy** as our observation says.

Note that, the above conclusion holds for arbitrary dataset. That is, **no matter what task we encounter, using this kernel will always achieves 100% training accuracy**.

Therefore, this can be one theoretical evidence for: **large γ leads to over-fitting**.

To conclude, this two observation gives us a **guideline** for choosing γ:

1. Large γ means our feature space tends to be linearly-separable (and our feature map is actually **sharp** Gaussian). Therefore, it should be more appropriate to select small C to resist over-fitting.

2. Small γ will lead to large number of Non-margin SVs. Therefore, we should guarantee there are not so many Non-margin SVs by gradually increasing γ. It should be more appropriate to select large C to resist under-fitting in this case.

In the following experiments, we aim to find the best RBF Kernel SVM and check these 2 observation.

Experiments Part:

In our experiment, the candidate of C is $\{0.1, 1, 2, 4, 6, 8, 10, 20, 40, 60, 100, 200, 500, 1e4\}$ and candidate of γ is $\{0.1, 1, 10\}$. The result is given as follows:

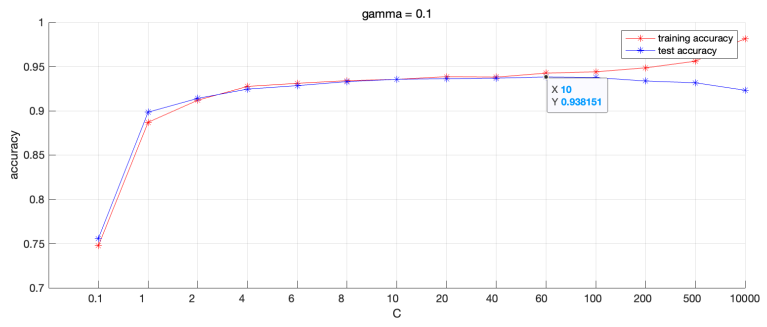


Figure 6: training and test accuracy curve for $\gamma = 0.1$

In Figure 6, the best C for $\gamma = 0.1$ is 60. The corresponding test accuracy is 0.9382.

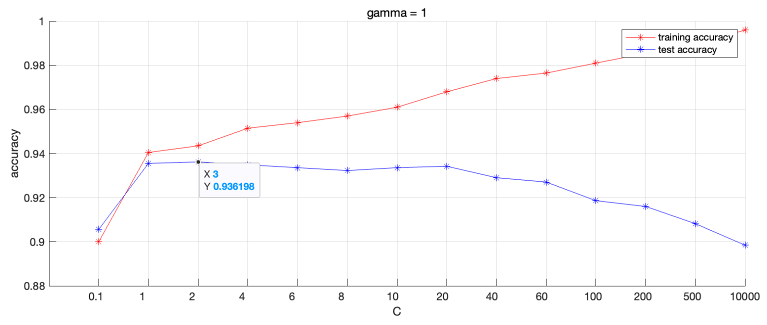


Figure 7: training and test accuracy curve for $\gamma = 1$

In Figure 7, the best C for $\gamma = 1$ is 2. The corresponding test accuracy is 0.9361.

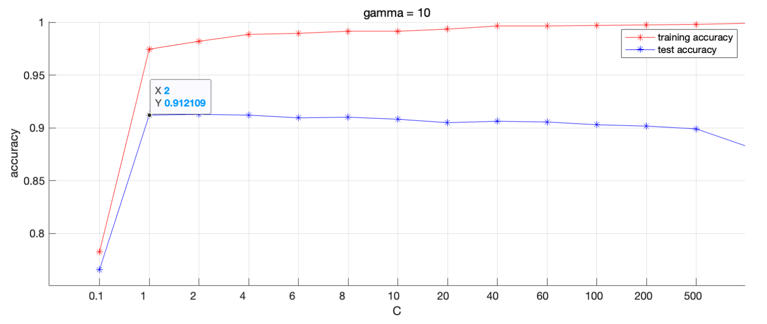


Figure 8: training and test accuracy curve for $\gamma = 10$

In Figure 8, the best C for $\gamma = 10$ is 1. The corresponding test accuracy is 0.912.

From these 3 figures, we can be convinced that our previous conclusion is true. Also, actually the first 2 RBF SVMs give us rather good results, approximately 0.938 test accuracy. This test accuracy is slightly lower than the SVM with $C = 2.1, p = 2$, **0.9395**.

Therefore, we choose **Polynomial Kernel SVM with** $p = 2, C = 2.1$ to predict *eval.mat*. The Pre-processing Procedure is **Normalization**.

5 Conclusion

5.1 Linear Kernel

1. It is very safe, in the sense that it is less likely to be over-fitting.

2. We can solve primal problems for Linear kernel since the dimension of feature space is not so big and sometimes can be more efficient.

3. The result is interpretab. We can make interpretation on the coefficient of each feature through our optimal weight.

4. Disadvantage: It may suffer from under-fitting when the dataset in the original space is highly non-linearly separable.

5.2 Polynomial Kernel

1. When we have some prior knowledge about the order of polynomial kernel (i.e., geometric information about our dataset), this method will be good.

2. Powerful than Linear Kernel, but not so powerful that will definitely over-fitting.

3. Disadvantage: When the order of polynomial kernel is big, we will suffer from great numerical problems about Kernel Matrix $\mathbf{K}$. The entry of Kernel Matrix will exponentially explode as the order increases (so we should not choose the order too big).

4. Disadvantage: Less interpretable than Linear Kernel.

5.3 Gaussian Kernel (RBF)

1. One of the most powerful kernel. in the sense that with special choice of parameter, we will definitely achieve 100% training accuracy (also extremely over-fitting).

2. We will not suffer from numerical problems like Polynomial Kernel.

3. Disadvantage: No good ways to determine the hyper-parameter σ.

4. Disadvantage: Less interpretable than Polynomial Kernel. Actually we can not attain the weights for the primal problems since the feature space is infinite-dimension.

5. Disadvantage: Large computational cost for the exponential computation.

6. Disadvantage: Easily over-fitting.

6 Appendix

All matlab codes (*.m) will be attached in .zip.