Programming Assignment 2

1. Logistic Regression

Regularization (with linear combination)

As shown in figure 1 that, under $\lambda = 0$ condition (without regularization), where loss functions are under maximum likelihood assumption, not only perform undoubtedly well on all training sets (possibly over-fit on complicated sets, not obvious, not illustrated here), also performed well on linearly separable case.

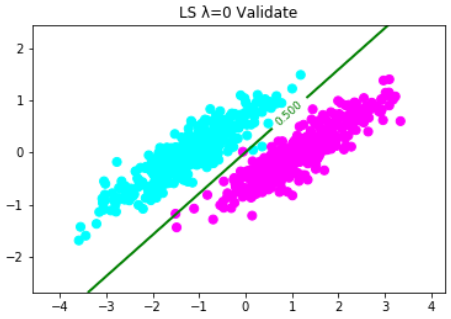

Figure 1: no regularization (validation) under MLE assumption (which is in fact true in this case)

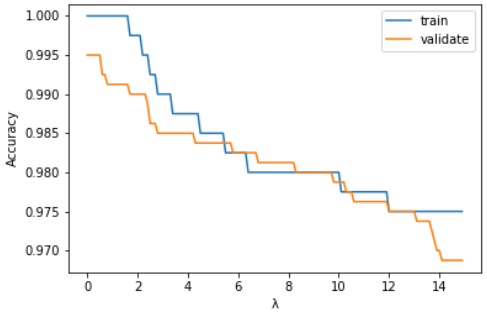

Figure 2: tuning λ for linearly separable dataset (linear function)

When λ increases dramatically, we see an decrease in accuracy, because large λ hinders descriptiveness of the function, hence creating under-fitting scenarios.

As shown in Figure 9 and Figure 10 (in Appendix) that non-linear dataset is not as prone to increased λ up front but could not benefit from it either. Because, for linear functions, regularization does not make intuitive sense as suppose to their basis function expansion counterpart. In particular, linear datasets should never benefit from increased λ at all as expect since, by definition, a straight line is representative enough for linear datasets.

Regularization (with polynomial expansion)

Although second order basis function expansion does not help with linear data so much (shown in Figure 11 and 12), which is expected as discussed in last section; we see that, for non-linear dataset, our accuracy first increases (then decreases of course) as we increase λ (shown in Figure 3). The result illustrates two important aspect of our new method (second order polynomial regularization).

1. For non-linear dataset, projecting feature space into higher dimension increases the power of the model to better fit given data, in other words, increases the descriptiveness of the model. However, by increasing descriptiveness, we are also adding risk of over-fitting.

2. Regularization helps with over-fitting, under the assumption that weights are to be regularized, higher the weights, higher the penalization. Although this assumption is false in some cases, in general (or at least in our case), it represents the hope that the true function we need to learn does not have crazy weights to fit everything in the training set, including the noise.

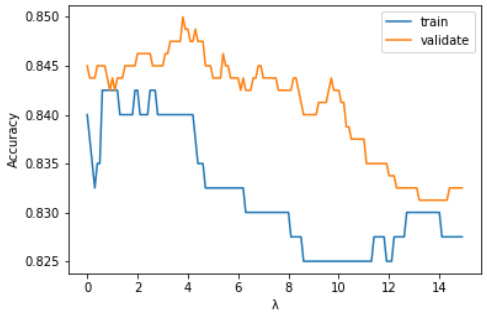

Figure 3: tuning λ for non-linear dataset (second order polynomial)

2. SVM

Linear datasets do not pose much of a challenge, so we won't discuss it here. However, for non-linear dataset, simple linear combinations (without any kernel) does not give us the flexibility or descriptiveness against non-linear dataset, as shown in Figure 13 in Appendix.

The Primal and Dual form of the SVM problem was a challenge, see notes for mathematical derivation in Figure 7 and 8 in the beginning of the Appendix.

For tuning over slack penalty C, see next section where we compare the kernels.

3. SVM Kernel

Simplicity of applying kernel is the most interesting aspect of the SVM implementation. See *svm_dual* class in the code for how simple the implementation is.

Comparing Kernels

As shown in Figure 4 that Gaussian kernel performs the best as we expected on linear non-separable dataset obviously because of much higher dimension give us the ability of finding margin.

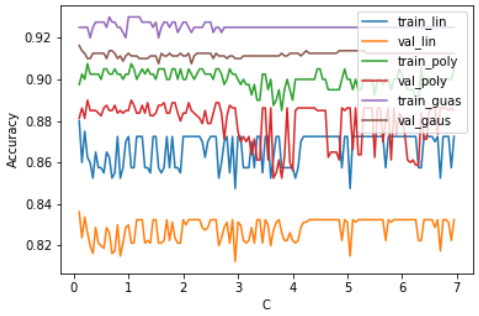

Figure 4: Linear non-Separable dataset (3 Kernels) σ=3

Before discussing how kernels differ, let's discusses the effect of slack penalty C first. As we see from Figure 5 that, although in different on other kernels, Gaussian kernel performance drops on non-linear validation set as we increase C. The reason is that high C overshadows the maximum margin term (considering the primal form), and complicated kernel and dataset strengths the effect of overshadowing.

Certainly we know that for non-linear data we are bound to use non-linear functions (in this case, kernelize the SVM), and we clearly see that second order polynomial performs way better than fitting linear separator.

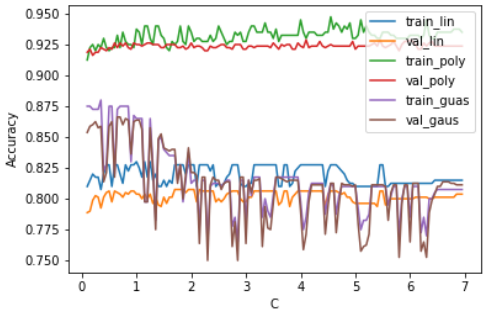

Figure 5: non-linear dataset (3 Kernels) σ=3

The effect of σ is quite clear through Figure 6 although we might be expecting the curve to go up a little and then go done. However, due to the simplicity of our dataset, although not linear, is considerably easy as shown in Figure 14. We would conclude that Gaussian kernel is very prone to over-fitting.

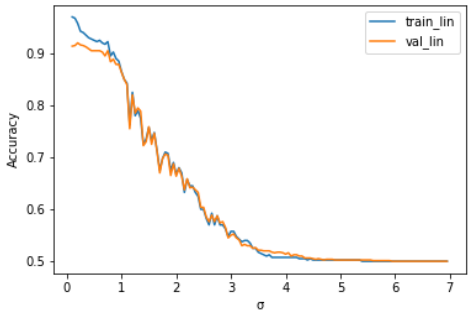

Figure 6: Tuning σ on non-linear dataset

Appendix

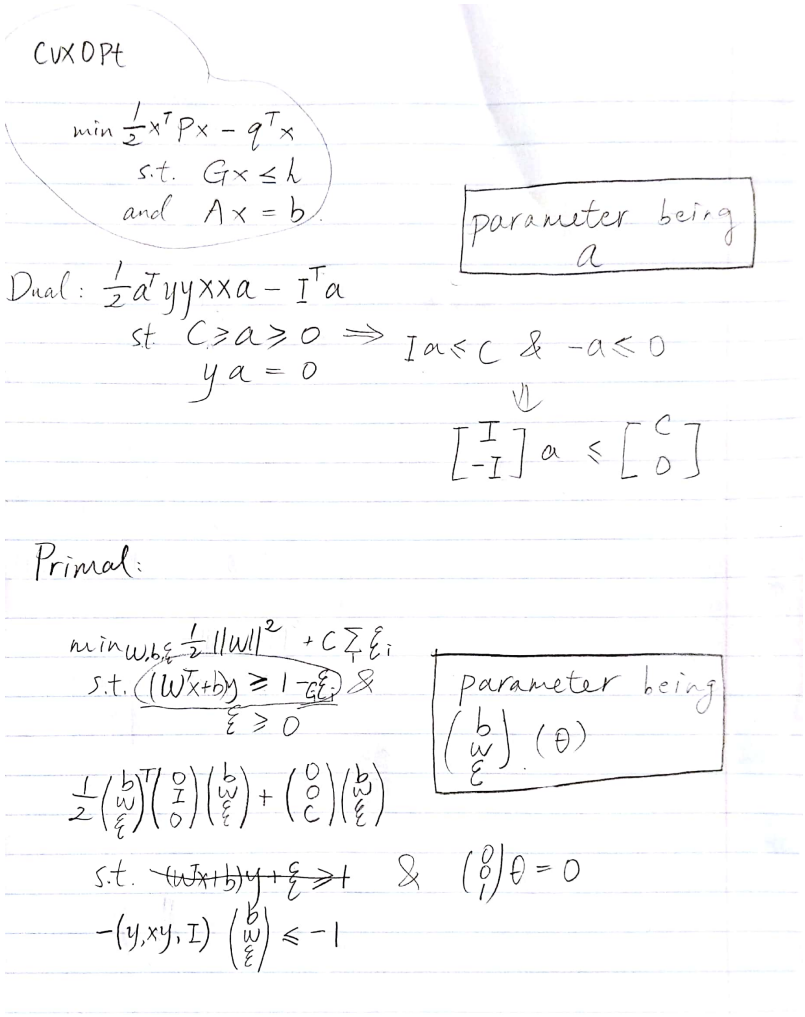

Figure 7: Dual and Primal problem formalization into QP

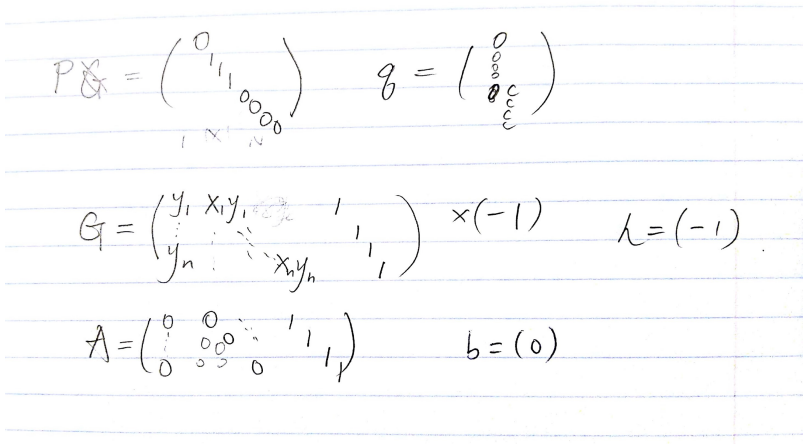

Figure 8: Dual and Primal problem formalization into QP

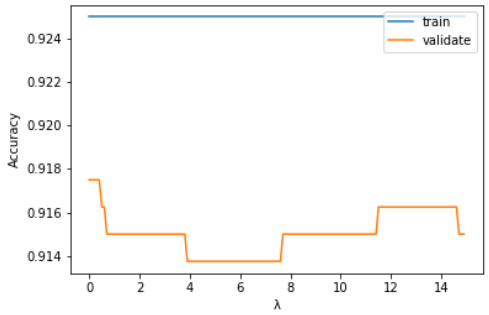

Figure 9: tuning λ for non-linearly separable dataset (linear function)

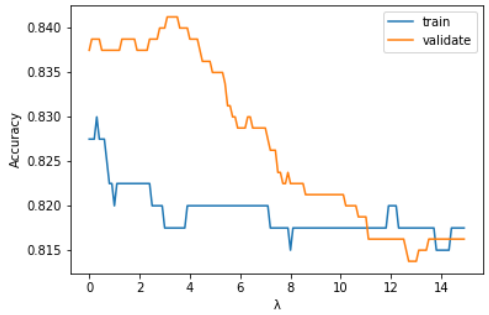

Figure 10: tuning λ for non-linear dataset (linear function)

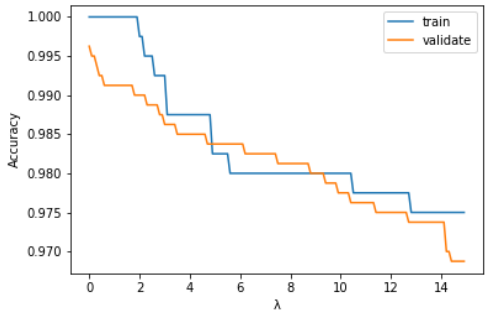

Figure 11: tuning λ for linearly separable dataset (second order polynomial)

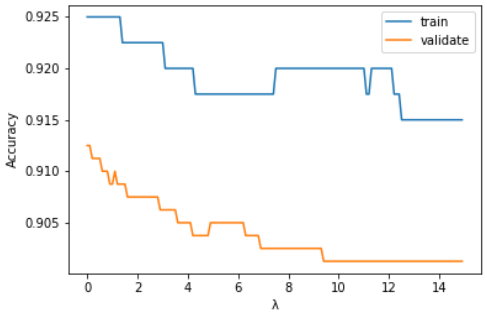

Figure 12: tuning λ for non-linearly separable dataset (second order polynomial)

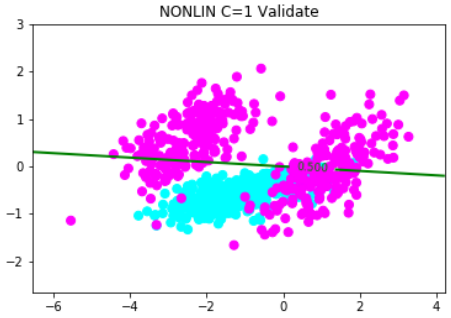

Figure 13: non-linear dataset (no Kernel)

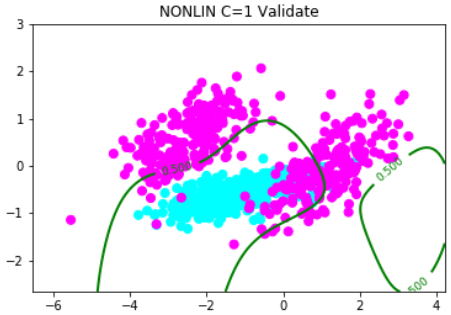

Figure 14: Gaussian Kernel on non-linear dataset