cs584 Assignment 4: Report

Xin Su
Department of Computer Science
Illinois Institute of Technology

April 24, 2016

Abstract

This report states the analysis of the Assignment 4's results. It resolves 3 major problems one by one. First I implement a linear SVM algorithm with hard margins to fit the datasets of linearly separable and non-linearly separable 2-class. Second I implement a linear SVM algorithm with soft margins model and fit it to the same datasets. Third, I implement an SVM with different kernel functions (polynomial and gaussian kernel functions). I evaluate the results for each model using confusion matrix, and accuracy. I also fit the models to two external datasets (one has missing features) and run the models with different parameters to see the impact of the parameters.

1. Problem statement

I explored the following problems:

(1) Generate a small dataset of 2D feature vectors of two classes such that the classes are linearly separable. Similarly generate an additional set with examples that are not linearly separable.
(2) Implement a linear SVM model with hard margins. Apply it to the datasets generated in step 1.
(3) Derive the expression of the dual problem that has to be maximized for linear SVM algorithm with soft margins.
(4) Implement a linear SVM model with soft margins. Apply it to the datasets generated in step 1.
(5) Implement a kernel-based SVM algorithm using polynomial and gaussian kernel functions. Apply the model to the datasets generated in step 1 and two external datasets as well. Test the effect of modifying different parameters of the algorithm.
(6) Test the model to datasets that one of the classes has more examples.

2. Proposed solution

- SVM with hard margins:
  - Dual problem objective function:

$$\max L_D = -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y^{(i)}y^{(j)}x^{(i)T}x^{(j)} + \sum_{i=1}^{m}\alpha_i$$

$$\sum_{i=1}^{m}\alpha_i y^{(i)} = 0$$

- Support vectors set:

$$sv = \{x^{(i)} \setminus \alpha^{(i)} > 0\}$$

- Parameters:

$$W = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}$$

$$W_0 = \frac{1}{\#\ of\ sv} \sum_{x \in sv}^{\|} \left(y^{(i)} - W^T x^{(i)}\right)$$

- Predict function:

$$\dot{y} = \begin{cases} 1\ if\ W^T x^{(i)} + W_0 > 0 \\ 0 \qquad otherwise \end{cases}$$

- SVM with soft margins:
  - Dual problem objective function:

$$\max L_D = -\frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y^{(i)} y^{(j)} x^{(i)T} x^{(j)} + \sum_{i=1}^{m} \alpha_i$$

$$\sum_{i=1}^{m} \alpha_i y^{(i)} = 0$$

$$0 \le \alpha_i \le c$$

  - Support vectors set:

$$sv = \{x^{(i)} \setminus \alpha^{(i)} > \varepsilon\}$$

  - Parameters:

$$W = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}$$

$$w_0 = \frac{1}{\#\ of\ sv} \sum_{x \in sv}^{\|} \left(y^{(i)} - W^T x^{(i)}\right)$$

  - Predict function:

$$\dot{y} = \begin{cases} 1\ if\ W^T x^{(i)} + w_0 > 0 \\ 0 \qquad otherwise \end{cases}$$

- Kernel based SVM
  - Dual problem objective function:

$$\max L_D = -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y^{(i)}y^{(j)}k\left(x^{(i)}, x^{(j)}\right) + \sum_{i=1}^{m}\alpha_i$$

$$\sum_{i=1}^{m}\alpha_i y^{(i)} = 0$$

$$0 \le \alpha_i \le c$$

  - Polynomial kernel function:

$$k\left(x^{(i)}, x^{(j)}\right) = \left(x^{(i)T}x^{(j)} + 1\right)^q$$

  - Gaussian kernel function

$$k\left(x^{(i)}, x^{(j)}\right) = \exp\left(-\frac{\left\|x^{(i)} - x^{(j)}\right\|^2}{2\sigma^2}\right)$$

- Evaluation function
  - Confusion Matrix:

		Label Classifier		
		C1	**…**	**Ck**
Predict Classifier	**C1**	C11	…	C1k
	…	…	…	…
	Ck	Ck1	…	Ckk

  - accuracy

$$accuracy = \frac{\Sigma C_{ii}}{\Sigma C_{ij}}$$

3. Implementation details

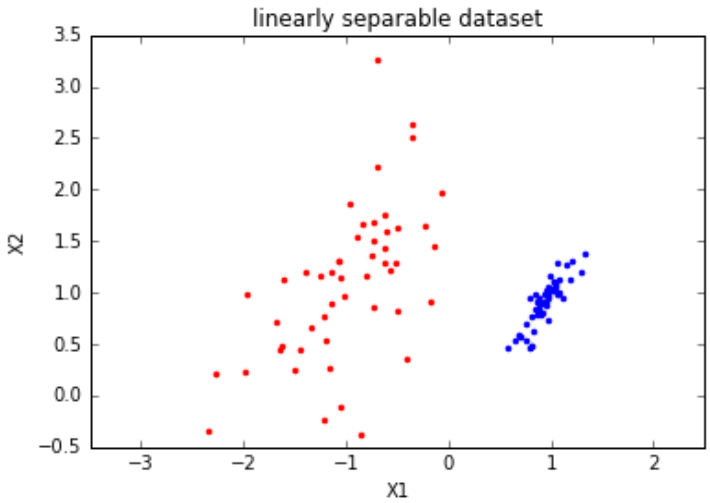
I use Python 2.7 as development language, and use IPython Notebook to write the code and run. It is very convenient for use. A user can view it simply using a browser, and the content will contain all output, including text, data and graphs.

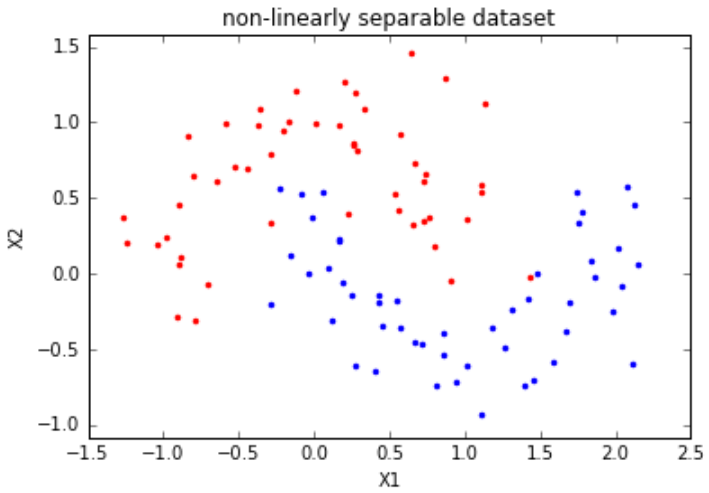
When you want to run the code, you should first install the IPython Notebook (Please refer to the Jupyter official website: http://jupyter.readthedocs.org/en/latest/install.html). After this step, you can open the .ipynb file and run. Just run from the top cell to the bottom. This top-down order is required.

4. Results and discussion

Problem 1

I use make_classification function to generate linearly separable dataset and make_moon generate non-linearly separable dataset.





Problem 2

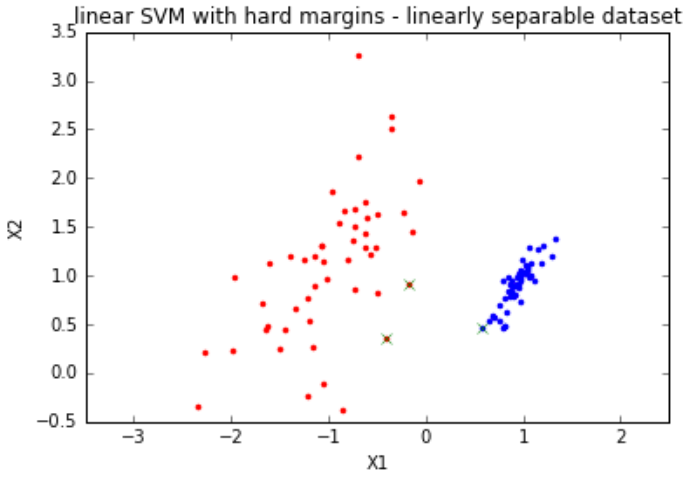
I apply the linear SVM model with hard margins to the generated datasets and use 5-fold cross validation to validate the model. The result when fitting to the linearly separable dataset is good, but it is not when fitting to the non-linearly separable dataset. The reason is that when applying to the non-linearly separable dataset, the model cannot find any acceptable solution to separate the data using a clear line with hard margins. Therefore it marks all data points as support vectors. (The point with 'x' in the figures are the support vectors)

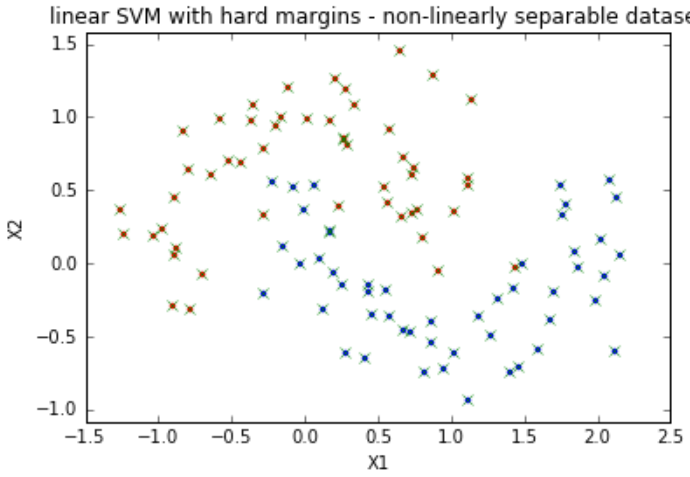
Problem 2 Result - apply to linearly separable dataset

	cross validation training avg	cross validation test avg	apply to all data
Confusion matrix	[[40. 0.] [0. 40.]] [[40. 0.] [0. 40.]] [[43. 0.] [0. 37.]] [[39. 0.] [0. 41.]] [[38. 0.] [0. 42.]]	[[10. 0.] [0. 10.]] [[10. 0.] [0. 10.]] [[7. 0.] [0. 13.]] [[11. 0.] [0. 9.]] [[12. 0.] [0. 8.]]	[[50. 0.] [0. 50.]]
accuracy	1.0	1.0	1.0
duration (s)	0.0257840156555		

Problem 2 Result - apply to non-linearly separable dataset

	cross validation training avg	cross validation test avg	apply to all data
Confusion matrix	[[11. 27.] [30. 12.]] [[20. 20.] [17. 23.]] [[7. 28.] [34. 11.]] [[10. 28.] [29. 13.]] [[9. 30.] [33. 8.]]	[[4. 7.] [5. 4.]] [[6. 3.] [7. 4.]] [[4. 9.] [5. 2.]] [[5. 6.] [6. 3.]] [[2. 7.] [6. 5.]]	[[11. 37.] [39. 13.]]
accuracy	0.31	0.39	0.24
duration (s)	0.0311119556427		





Problem 3

This is the procedure to derive the expression of the dual problem that has to be maximized for linear SVM algorithm with soft margins. The primal objective function is:

$$L_p = \frac{1}{2}W^T W + c\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\alpha_i\left(y^{(i)}\left(W^T x_i + w_0\right) - 1 + \xi_i\right) - \sum_{i=1}^{m}\beta_i\xi_i$$

we have derivative functions:

$$\frac{\partial L_p}{\partial w_i} = W - \sum_{i=1}^{m}\alpha_i y^{(i)} x_i = 0 \quad \Rightarrow \quad W = \sum_{i=1}^{m}\alpha_i y^{(i)} x^{(i)}$$

$$\frac{\partial L_p}{\partial w_0} = -\sum_{i=1}^{m}\alpha_i y^{(i)} = 0 \quad \Rightarrow \quad \sum_{i=1}^{m}\alpha_i y^{(i)} = 0$$

$$\frac{\partial L_p}{\partial \xi_i} = c - \alpha_i - \beta_i = 0 \quad => \quad c - \alpha_i - \beta_i = 0$$

Then use the first equation to replace all W, we have:

$$L_p = \frac{1}{2}\left(\sum_{i=1}^{m} \alpha_i y^{(i)} x_i^T\right)\left(\sum_{i=1}^{m} \alpha_i y^{(i)} x_i\right) + c\sum_{i=1}^{m} \xi_i - \sum_{i=1}^{m} \alpha_i y^{(i)}\left(\sum_{j=1}^{m} \alpha_j y^{(j)} x_j^T\right) x^{(i)} - \sum_{i=1}^{m} \alpha_i y^{(i)} w_0 + \sum_{i=1}^{m} \alpha_i - \sum_{i=1}^{m} \alpha_i \xi_i - \sum_{i=1}^{m} \beta_i \xi_i$$

Therefore, using equations derived from the previous derivative functions, the primal objective function becomes the dual objective function:

$$L_D = -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y^{(i)} y^{(j)} x_i^T x_j + \sum_{i=1}^{m} \alpha_i$$

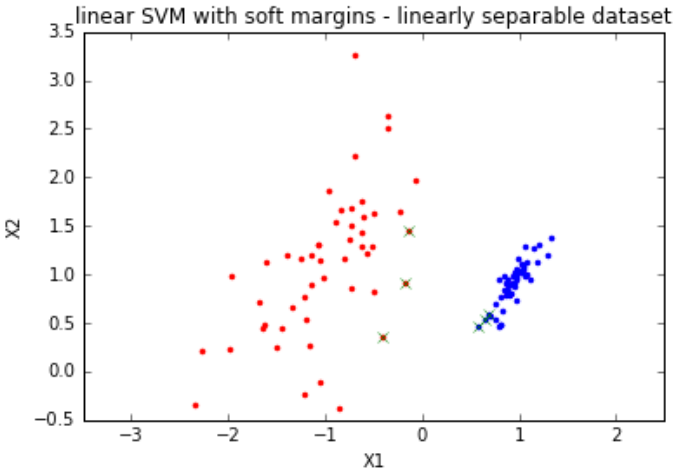
This is the dual problem that has to be maximized for linear SVM algorithm with soft margins. The following are the conditions it has to meet:

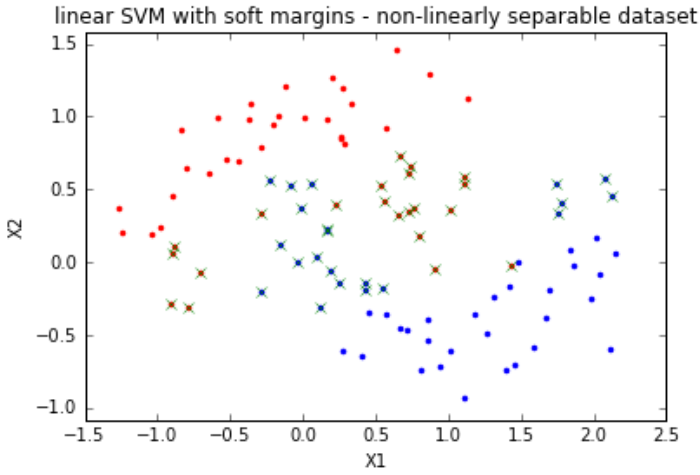
$$\sum_{i=1}^{m} \alpha_i y^{(i)} = 0$$

$$0 \le \alpha_i \le c$$

Problem 4

I apply the linear SVM model with soft margins to the generated datasets and use 5-fold cross validation to validate the model. The result when fitting to the linearly separable dataset is good, but it is not when fitting to the non-linearly separable dataset. The reason is that when applying to the non-linearly separable dataset, the model can find a acceptable solution to separate the data using a clear line with soft margins. However, it will mix up some points. Therefore it marks points as support vectors but within the margin there are some points will be discriminated by mistake. (The point with 'x' in the figures are the support vectors)





Problem 4 Result - apply to linearly separable dataset

	cross validation training avg	cross validation test avg	apply to all data
Confusion matrix	[[42. 0.] [0. 38.]] [[41. 0.] [0. 39.]] [[37. 0.] [0. 43.]] [[42. 0.] [0. 38.]] [[38. 0.] [0. 42.]]	[[8. 0.] [0. 12.]] [[9. 0.] [0. 11.]] [[13. 0.] [0. 7.]] [[8. 0.] [0. 12.]] [[12. 0.] [0. 8.]]	[[50. 0.] [0. 50.]]
accuracy	1.0	1.0	1.0
duration (s)	0.035218000412		

Problem 4 Result - apply to non-linearly separable dataset

	cross validation training avg	cross validation test avg	apply to all data
Confusion matrix	[[37. 5.] [4. 34.]] [[34. 3.] [5. 38.]] [[36. 7.] [4. 33.]] [[37. 6.] [3. 34.]] [[36. 7.] [4. 33.]]	[[8. 2.] [1. 9.]] [[11. 4.] [0. 5.]] [[9. 0.] [1. 10.]] [[8. 1.] [2. 9.]] [[9. 0.] [1. 10.]]	[[45. 7.] [5. 43.]]
accuracy	0.88	0.88	0.88
duration (s)	0.0293900966644		

Problem 5

I apply the linear kernel-based SVM model with soft margins to the generated datasets and use 5-fold cross validation to validate the model.

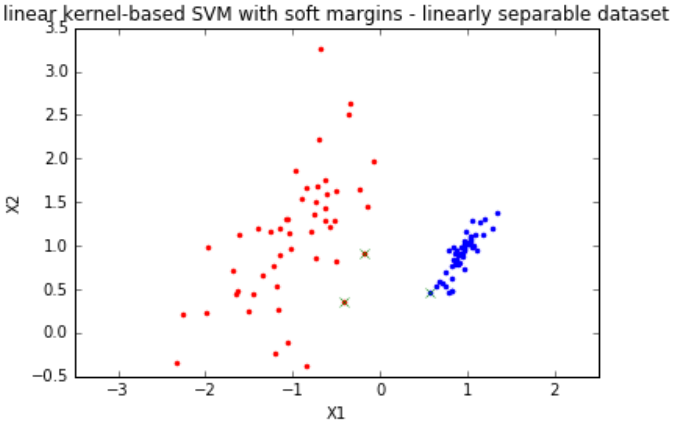
First, the result when fitting to the linearly separable dataset with polynomial kernel function is good, but it becomes lower, yet acceptable, when fitting to the non-linearly separable dataset. The reason is that when applying to the non-linearly separable dataset, the model can find a acceptable solution to separate the data using a clear line with soft margins. However, it will mix up some points. Therefore it marks points as support vectors but within the margin there are some points will be discriminated by mistake. (The point with ‘x’ in the figures are the support vectors)

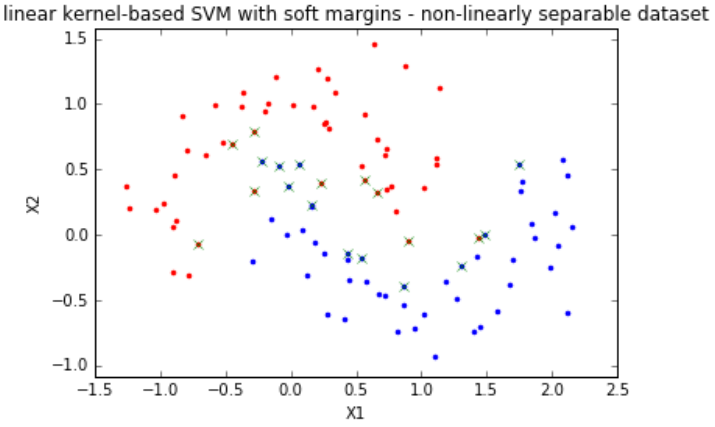
Problem 5 Result - apply to linearly separable dataset (polynomial)

	cross validation training avg	cross validation test avg	apply to all data
Confusion matrix	[[39. 0.] [0. 41.]] [[42. 0.] [0. 38.]] [[40. 0.] [0. 40.]] [[37. 0.] [0. 43.]] [[42. 0.] [0. 38.]]	[[11. 0.] [0. 9.]] [[8. 0.] [0. 12.]] [[10. 0.] [0. 10.]] [[13. 0.] [0. 7.]] [[8. 0.] [0. 12.]]	[[50. 0.] [0. 50.]]
accuracy	1.0	1.0	1.0
duration (s)	0.0247509479523		

Problem 5 Result - apply to non-linearly separable dataset (polynomial)

	cross validation training avg	cross validation test avg	apply to all data
Confusion matrix	[[36. 5.] [5. 34.]] [[38. 4.] [5. 33.]] [[35. 5.] [5. 35.]] [[34. 7.] [5. 34.]] [[33. 7.] [4. 36.]]	[[8. 2.] [1. 9.]] [[6. 3.] [1. 10.]] [[9. 2.] [1. 8.]] [[10. 0.] [1. 9.]] [[11. 0.] [2. 7.]]	[[44. 7.] [6. 43.]]
accuracy	0.87	0.87	0.87
duration (s)	0.0342810153961		





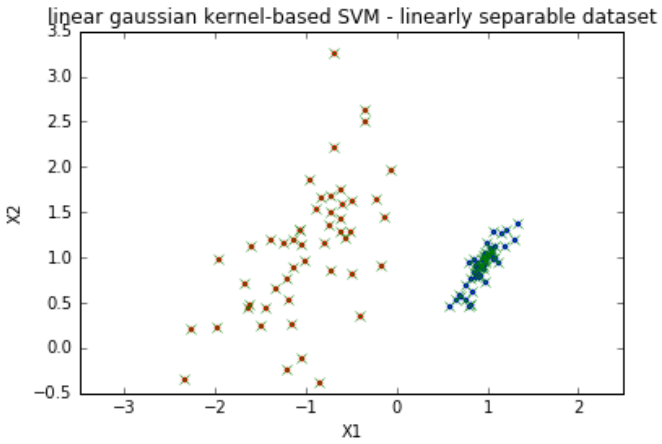
Second, the result when fitting to the linearly separable dataset with gaussian kernel function is good, but it becomes lower, yet acceptable, when fitting to the non-linearly separable dataset. (The point with 'x' in the figures are the support vectors)

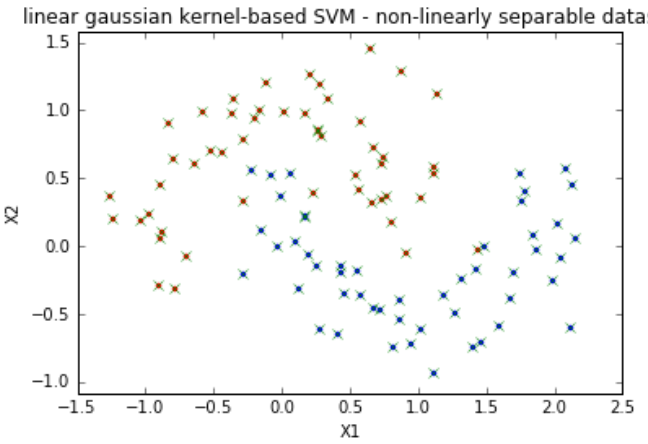
Problem 5 Result - apply to linearly separable dataset (gaussian)

	cross validation training avg	cross validation test avg	apply to all data
Confusion matrix	[[40. 0.] [0. 40.]] [[44. 0.] [0. 36.]] [[40. 0.] [0. 40.]] [[38. 0.] [0. 42.]] [[38. 0.] [0. 42.]]	[[10. 0.] [0. 10.]] [[6. 0.] [0. 14.]] [[10. 0.] [0. 10.]] [[12. 0.] [0. 8.]] [[12. 0.] [0. 8.]]	[[50. 0.] [0. 50.]]
accuracy	1.0	1.0	1.0
duration (s)	0.0303721427917		

Problem 5 Result - apply to non-linearly separable dataset (gaussian)

	cross validation training avg	cross validation test avg	apply to all data
Confusion matrix	[[35. 9.] [7. 29.]] [[35. 10.] [8. 27.]] [[31. 9.] [5. 35.]] [[32. 8.] [8. 32.]] [[31. 8.] [8. 33.]]	[[6. 2.] [2. 10.]] [[6. 1.] [1. 12.]] [[10. 2.] [4. 4.]] [[9. 3.] [1. 7.]] [[10. 3.] [1. 6.]]	[[41. 11.] [9. 39.]]
accuracy	0.8	0.8	0.8
duration (s)	0.0281429290771		





Third, the I implement the models to two external datasets, iris.data and wind.data. The iris.data has missing features. I use the average values to fill in the missing values. The result shows below:

Problem 5 Result - apply to external dataset

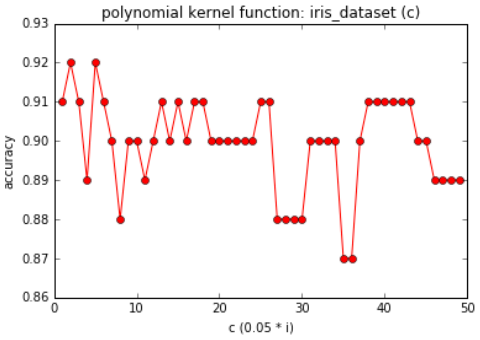
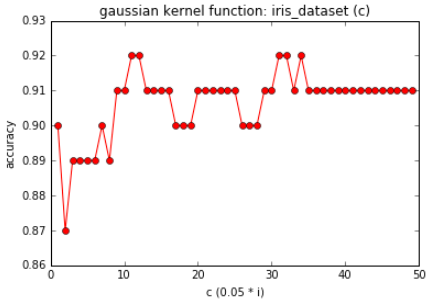
	iris.data		wine.data	
	polynomial	**gaussian**	**polynomial**	**gaussian**
Confusion matrix	[[43. 3.] [7. 47.]]	[[46. 4.] [4. 46.]]	[[59. 8.] [0. 63.]]	[[58. 0.] [1. 71.]]
accuracy	0.9	0.92	0.938461538462	0.992307692308

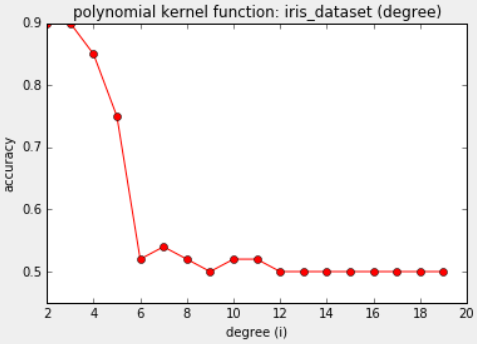
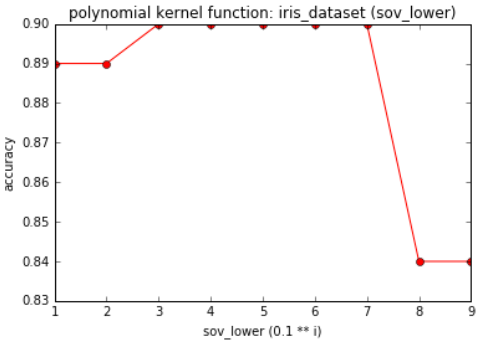
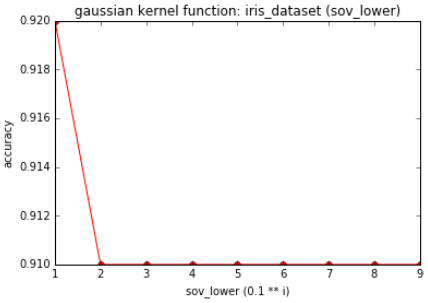
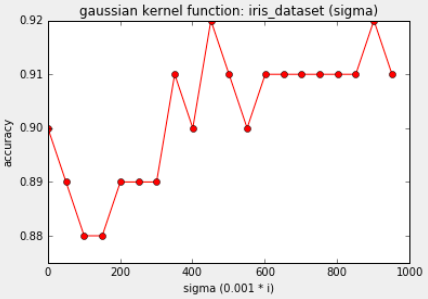
Forth, I test the models with different parameters and plot the graph using the iris.data. It shows that:

- The c parameter affects the accuracy little, with floating results. The reason may be that when using this model, the c parameter only constrains the alpha which will not quite affect the final result.
- As the degree goes high the accuracy drops for the polynomial kernel-based SVM algorithm. The reason may be the dataset is not fit to a higher dimensional features combination.
- As the sov_lower goes up, it will affect the accuracy and makes it drop somehow, but when it raises to some level the accuracy will remain the same. The reason may be that this parameter defines the margin and when it raises, the margin will become larger which will discriminate the points with more mistake, but when it raises to some level the margin will enclose all the points and will never change.
- The sigma parameter affects the accuracy little, with floating results.

The result shows below:

Problem 5 Result - apply to external dataset (different parameters)

	iris.data	
	polynomial	**gaussian**
c		

	iris.data	
	polynomial	**gaussian**
degree		0.92
sov_lower		
sigma		

Problem 6

I apply the linear kernel-based SVM model with soft margins to the iris_c.data which cuts the data of one of the classes and use 5-fold cross validation to validate the model. The result shows that it won't affect too much.

Problem 6 Result - apply to external dataset (iris_c.data)

	iris.data		iris_c.data	
	polynomial	**gaussian**	**polynomial**	**gaussian**
Confusion matrix	[[43. 3.] [7. 47.]]	[[46. 4.] [4. 46.]]	[[47. 1.] [3. 23.]]	[[49. 4.] [1. 20.]]
accuracy	0.9	0.92	0.945945945946	0.932432432432

5. References

[1] https://www.coursera.org/learn/machine-learning
[2] http://www.numpy.org/
[3] https://www.python.org/
[4] http://scikit-learn.org/