Machine Learning 2017

HW4: Supporting Vector Machine

Group 22

A052153 洪章瑋 A051010 林奕汝 0550205 許祖瑞

As one of kernel-based algorithms that have *sparse* solutions, SVM can not only provide predictions for new inputs depends only on the kernel function with the subset of the training data points, but it can also determinate the model parameters corresponding to a convex optimization problem so that any local solution is a global optimum. SVM is a decision machine, so it doesn't provide posterior probabilities.

The structure of this report would be the theory explanation first, the implementation details next, and the performance evaluation finally.

Support vector machines (SVM)

Better than many other kernel-based algorithm, SVM doesn't need to evaluate for all possible pairs x_n and x_m of the training points in the kernel function. And it also can determinate the model parameters corresponding to a convex optimization problem.

SVM maximizes margin classifiers at the beginning, that is, maximizing the margin from the data points having minimum distance to the decision surface. To comply such circumstance, the Lagrange multipliers is a good solution. Giving dual representation eliminates **w** and *b* from $L(\mathbf{w},b,\mathbf{a})$. Furthermore defines the kernel function into $k(x,x') = ø(x)^T ø(x)$. Finally maximize such Lagrange equation:

$$\widetilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m) \qquad (7.10)$$

Instead of giving the probability of that the point belongs to such class, SVM do the classification for any new input data **x**:

$$y(\mathbf{x}) = \sum_{n=1}^{N} a_n t_n k(\mathbf{x}, \mathbf{x}_n) + b. \qquad (7.13)$$

For equation (7.10) in Karush-Kuhn-Tucker (KKT) conditions, every data point, either $a_n = 0$ or $t_n y(x_n) = 1$. Combining to the equation (7.13), for a_n does not equal to zero, the corresponding data points influence the result of y(x) and thus are called *support vectors*.

In the following sections we introduce basic concepts of dual representations and kernel methods. Then introduces the details of how the a_n be estimated allowing class overlapping with **ν**-SVM.

Dual Representations

A dual representation of a regularized sum-of-squares error function with the parameter vector **a** is reformulated from the parameter vector **w**.

A regularized sum-of-squares error function with **w** is

$$J(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left\{\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) - t_n\right\}^2 + \frac{\lambda}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} \tag{6.2}$$

Define **a** and *Gram matrix* **K** to reformulate the error function *J*(w)

$$a_n = -\frac{1}{\lambda}\left\{\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) - t_n\right\}. \tag{6.4}$$

$$K_{nm} = \boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m) \tag{6.6}$$

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^{\mathrm{T}}\mathbf{K}\mathbf{K}\mathbf{a} - \mathbf{a}^{\mathrm{T}}\mathbf{K}\mathbf{t} + \frac{1}{2}\mathbf{t}^{\mathrm{T}}\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^{\mathrm{T}}\mathbf{K}\mathbf{a}. \tag{6.7}$$

At the minimum of *J*(a), the gradient should be zero, and the solution of **a** can be

$$\mathbf{a} = (\mathbf{K} + \lambda\mathbf{I}_N)^{-1}\mathbf{t}. \tag{6.8}$$

Even though in the dual formulation, the parameter vector **a**, a N x N matrix, is larger than **w**, a M x M matrix in original parameter space formulation. The advantage is that the dual representation is entirely in terms of the kernel function. As a result, it avoids the explicit introduction of the feature vector *ø(x)*. That is, it allows us implicitly to use feature spaces of infinite dimensionality.

Kernel methods

Refers to the discussion in Bayesian Linear Regression (Section 3.3). The predictive mean can be written as below. And the mean of the predictive distribution at a point x is given by a linear combination of the training set target variables t_n .

$$y(\mathbf{x}, \mathbf{m}_N) = \mathbf{m}_N^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}) = \beta \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}} \mathbf{S}_N \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{t} = \sum_{n=1}^{N} \beta \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}} \mathbf{S}_N \boldsymbol{\phi}(\mathbf{x}_n) t_n \tag{3.60}$$

Define a function $k(x,x_n)$, which is known as the *smoother matrix* or *equivalent kernel*.

$$y(\mathbf{x}, \mathbf{m}_N) = \sum_{n=1}^{N} k(\mathbf{x}, \mathbf{x}_n) t_n \tag{3.61}$$

$$k(\mathbf{x}, \mathbf{x}') = \beta \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}} \mathbf{S}_N \boldsymbol{\phi}(\mathbf{x}') \tag{3.62}$$

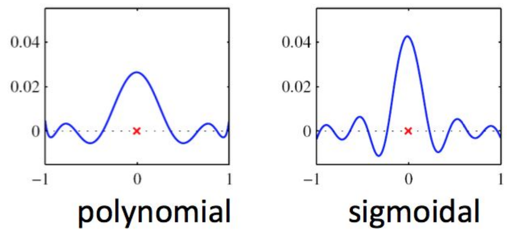
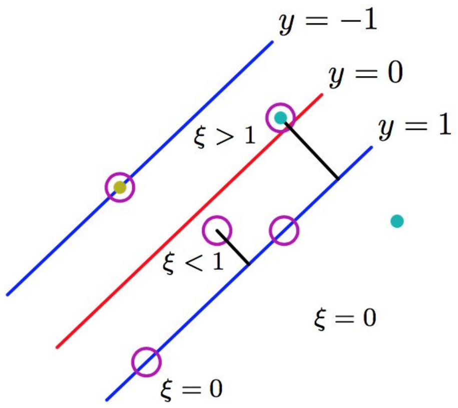


Figure1. local equivalent kernels among different basis functions

There are a few remarkable properties the equivalent kernel perform. First, even for non-local basis functions still have local equivalent kernels like Figure1. Second, instead of introducing a set of basis functions, we can define a localized kernel directly and use it to make predictions for new input vectors x. Third, It can be expressed as an inner product as below. Last, for all **x**, the summation of kernel functions of x and x_n, n=1,..., N, equals to one.

However, the kernel function can be negative as well as positive. Hence, the corresponding predictions are not necessarily convex combinations of the training set target variables.

ν-SVM and C-SVM



Because in practice, the class-conditional distributions may overlap and need to modify the SVM to allow some of the training points to be misclassified. In subsequent optimization problem, we use *slack variables* ξ_n to make penalty a linear function of its distance in the wrong side. These variables are defined by zero for data points that are on or inside the correct margin boundary and $\xi_n = |t_n - y(x_n)|$ for other points.

While softly penalizing the points misclassified, we use a parameter $C > 0$ to control the trade-off between the slack variable penalty and the margin:

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N} a_n \{t_n y(\mathbf{x}_n) - 1 + \xi_n\} - \sum_{n=1}^{N}\mu_n\xi_n \quad (7.22)$$

After the eliminating to **w**, *b*, and $\{\xi_n\}$, we have dual Lagrangian:

$$\widetilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m) \quad (7.32)$$

Maximizing it leads to *box contstraints* (7.33) and it has property (7.34):

$$0 \leqslant a_n \leqslant C \quad (7.33)$$

$$\sum_{n=1}^{N} a_n t_n = 0 \quad (7.34)$$

An alternative, equivalent formulation of the SVM

$$\widetilde{L}(\mathbf{a}) = -\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m) \quad (7.38)$$

subject to the constraints

$$0 \leqslant a_n \leqslant 1/N \quad (7.39)$$

$$\sum_{n=1}^{N} a_n t_n = 0 \quad (7.40)$$

$$\sum_{n=1}^{N} a_n \geqslant \nu. \quad (7.41)$$

This approach has the advantage that the parameter $\boldsymbol{\nu}$, which replaces *C*, can be interpreted as both an upper bound on the fraction of *margin errors* and a lower bound on the fraction of support vectors.

Experiments

Performance with different hyper parameters

Figure 2. influence of C in C-SVM

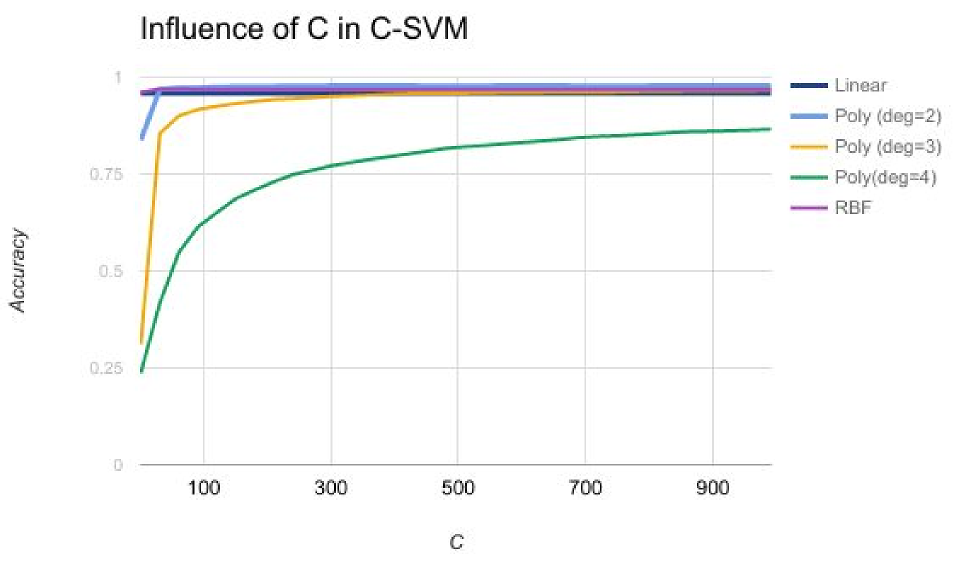
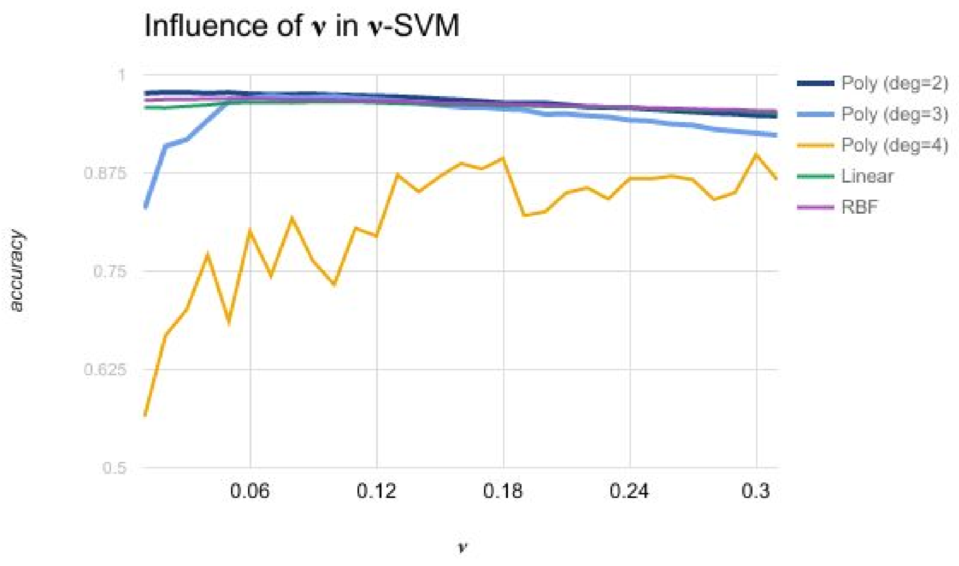


*Figure 3. influence of **ν** in **ν**-SVM*



In figure 2, it is obvious that hyper parameter C has no influence in Linear and RBF kernel except polynomial kernel. With polynomial kernel, accuracy of model increases along with C and nearly converges at 1000. Another interesting point is that polynomial kernel with higher degrees has lower performance.

In figure 3, most of the experiment results are same as figure 2. However, we can observe that performance of all kernels decrease when **ν** increasing.

Larger C makes SVM to find a small margin, smaller C make SVM to find a larger margin. **ν** controls margin errors and number of supporting vector simultaneously. Although C and **ν** are designed for soft penalizing, C is range from zero to infinity which is unbounded value and **ν** is bounded in 0 to 1. Moreover, **ν** is an upper bound on the fraction of *margin errors* and a lower bound on the fraction of support vectors. From hyper parameter searching perspective, it is easier to find optimal **ν** than to find C.

Performance with different kernels

All results are reported with optimal hyper parameter settings for each model. The best model is **ν**-SVM with polynomial with degree 3 kernel. In most of cases, **ν**-SVM wins C-SVM, except for the polynomial with degree 4 kernel. In theory, both **ν**-SVM and C-SVM have the similar performance, but it is harder to find optimal hyper parameters for C-SVM.

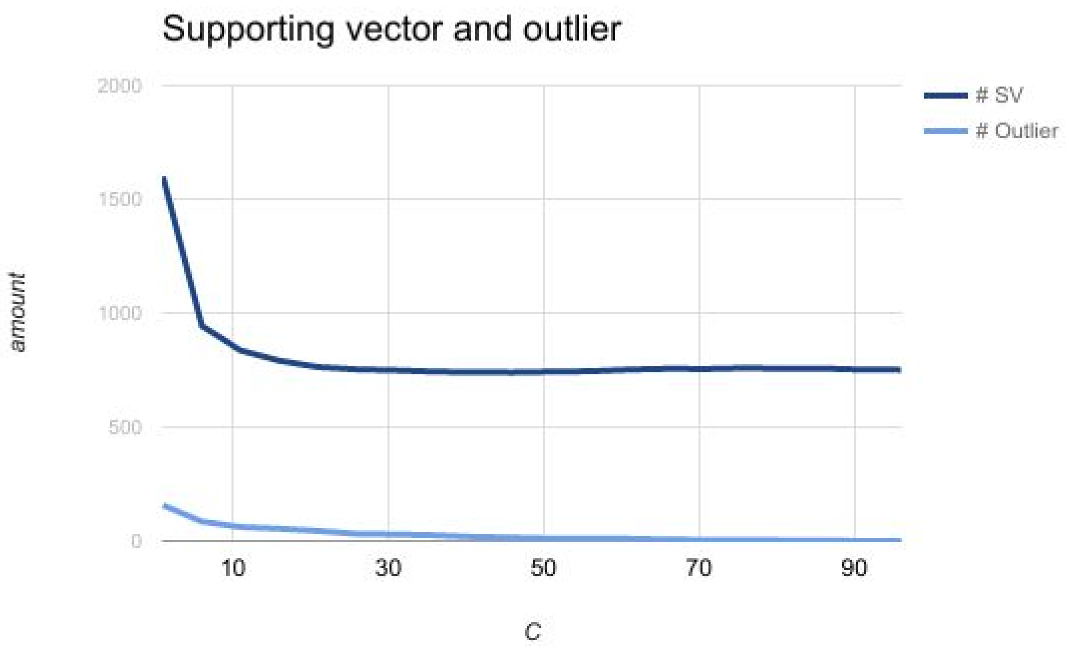
Chart 1. Error Rate Performance

Kernel		**ν**-SVM	C-SVM
Linear		**4.12%**	5.00%
Polynomial	2-Degree	**2.44%**	2.88%
	3-Degree	**2.24%**	2.96%
	4-Degree	5.72%	**4.68%**
Radial basis		**2.88%**	**2.88%**

Supporting Vectors(SV), Outliers with different hyper parameter

In this section, we would discuss about how hyper parameters affect supporting vector and outlier and what the connection between supporting vector, outlier and performance. We use C-SVM with RBF kernel to do discussion. **(small circle = training data, big circle = supporting vector, cross = outlier)**

Figure 4. C and number of SV, outlier



According to figure above, we can learn that when C increasing, number of supporting vectors and outliers decrease. This experiment result is consistent with our intuition of C since larger C should decrease number of supporting vector and outliers due to shallow margin.

Figure 5. Supporting vector and outlier of C-SVM (C = 1.0)

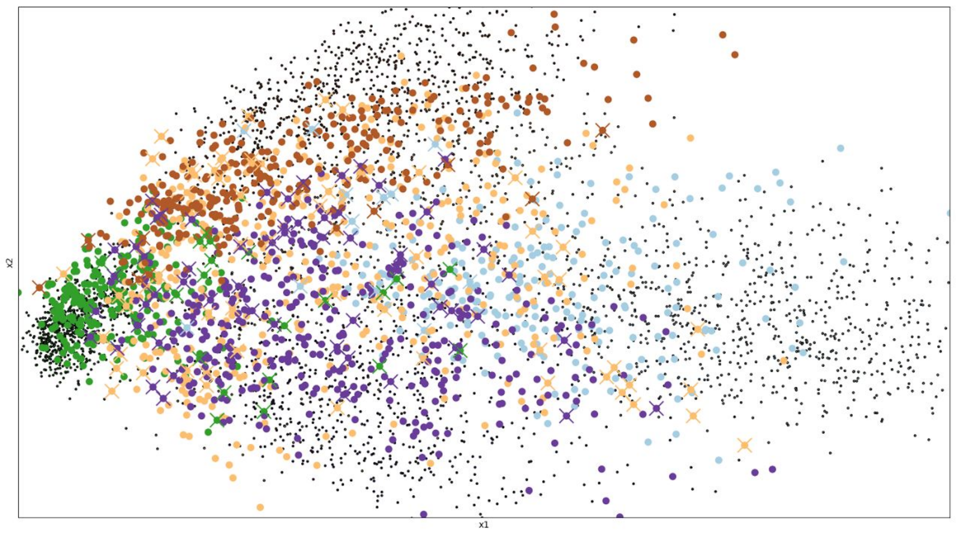
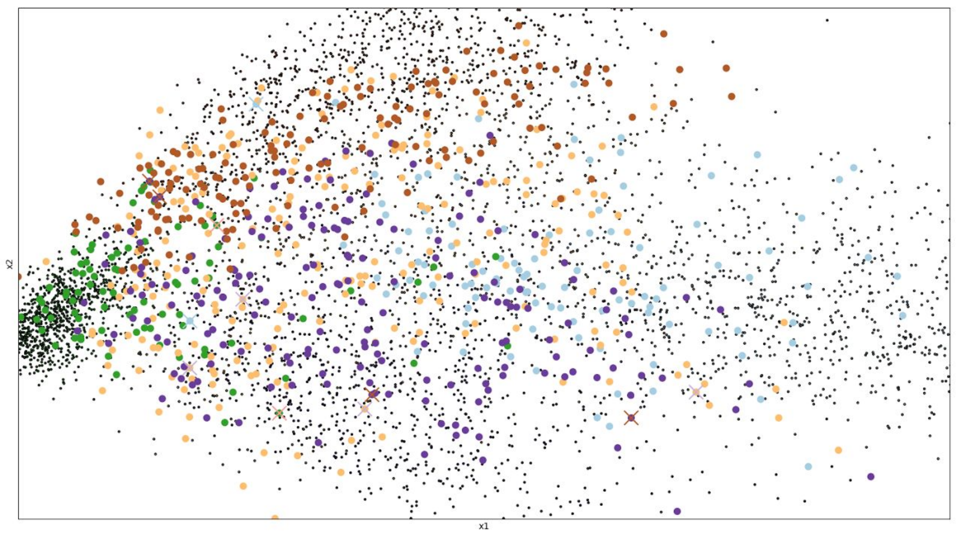


Figure 6. Supporting vector and outlier of C-SVM (C = 50.0)



In addition to numerical proof in figure 4, the above two visualization results give a straightforward proof for our intuition of C. In **ν**-SVM, number of supporting vector increases with **ν** increasing and effect of supporting vector is almost same as C-SVM. Hence we ignore the experiment result of **ν**-SVM here.

Performance and Supporting Vectors(SV), Outliers

Now that number of supporting vector can implicitly infer margin error, can we claim that number of supporting vector is proportional to error rate ? We show that by experimenting with C-SVM with polynomial kernel and C-SVM with RBF kernel.

Figure 6. Performance with different number of supporting vectors (C-SVM, C = 1, Kernel = Polynomial-degree-2)

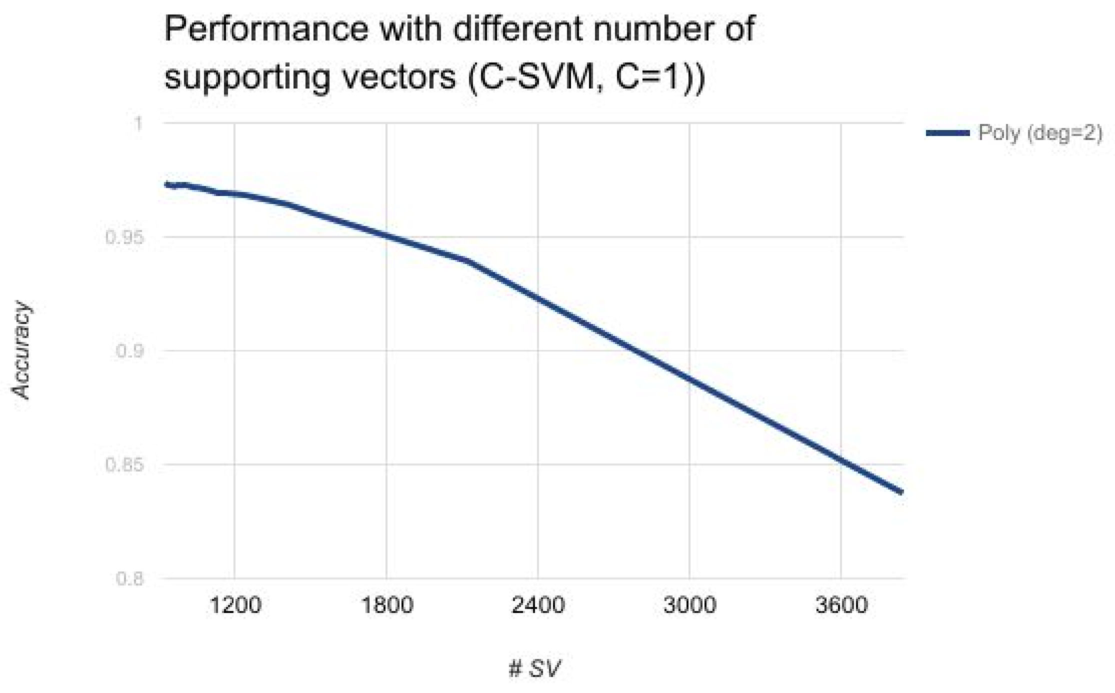
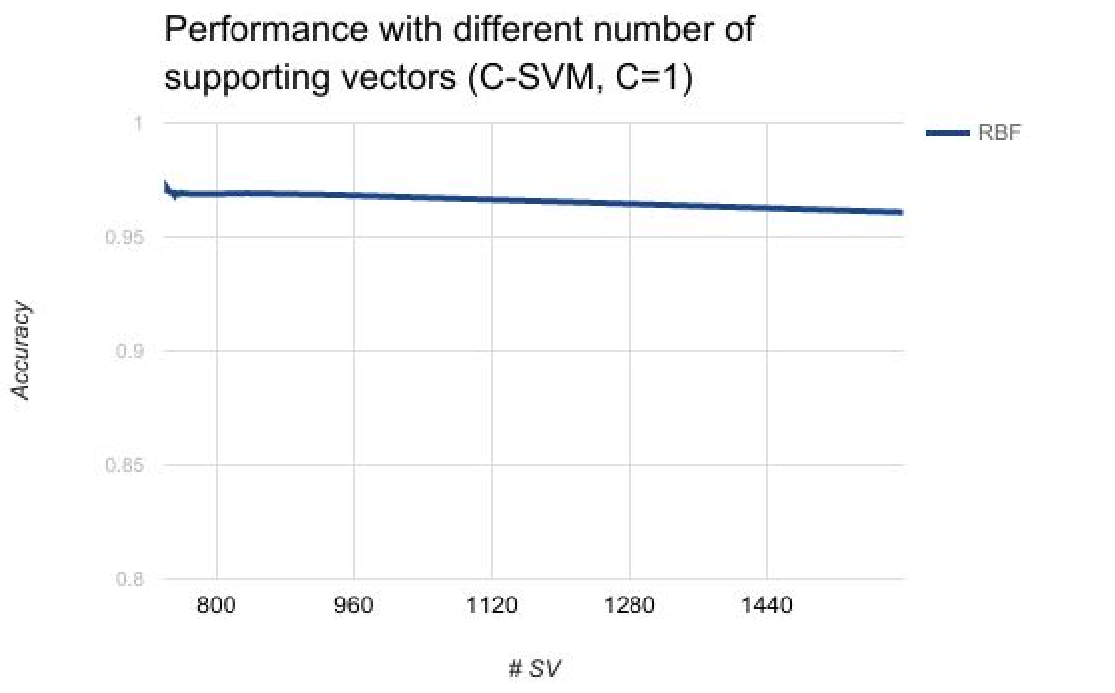


Figure 7. Performance with different number of supporting vectors (C-SVM, C = 1, Kernel = RBF)



It is consistent with our proposition, the number of supporting vector can directly affect the accuracy of model. Furthermore, supporting vector selection highly depends on kernel since RBF kernel has better performance than polynomial kernel when the other settings are same.