Homework 4

組別22

A052153 洪章瑋　　A051010 林奕汝　　0550205 許祖瑞

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

In SVM, we maximizing 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. By giving dual representation we can further elimination **w** and *b* from *L*(**w**,*b*,**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 $\boldsymbol{\nu}$-SVM.

Dual Representations

In mathematics, if G is a group and ρ is a linear representation of it on the vector space V, then the dual representation ρ* is defined over the dual vector space V* as follows: ρ*(g) is the transpose of $\rho(g^{-1})$, that is, $\rho^*(g) = \rho(g^{-1})^T$ for all $g \in G$.

Related to the use of kernel methods, a dual representation of a regularized sum-of-squares error function with the parameter vector **a**, is reformulated from the parameter vector **w**. Refers to 6.1 section in the textbook:

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

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

At the minimum of it, the gradient should be zero, and the solution of **w** can be

$$\mathbf{w} = -\frac{1}{\lambda}\sum_{n=1}^{N}\left\{\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) - t_n\right\}\boldsymbol{\phi}(\mathbf{x}_n) = \sum_{n=1}^{N} a_n\boldsymbol{\phi}(\mathbf{x}_n) = \boldsymbol{\Phi}^{\mathrm{T}}\mathbf{a} \tag{6.3}$$

We can then 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. We can avoid 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). Considering a zero-mean isotropic Gaussian governed by a single precision parameter α, we know that maximization of the posterior distribution with respect to

w is equivalent to the minimization of the sum-of-squares error function with the addition of a quadratic regularization term as below.

$$p(\mathbf{w}|\mathbf{t}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N) \tag{3.49}$$

where

$$\begin{aligned} \mathbf{m}_N &= \beta \mathbf{S}_N \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{t} & (3.53) \\ \mathbf{S}_N^{-1} &= \alpha \mathbf{I} + \beta \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi}. & (3.54) \end{aligned}$$

Then 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 .

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

We can then define a function k(x,x_n) , which is known as the *smoother matrix* or *equivalent kernel*.

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

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

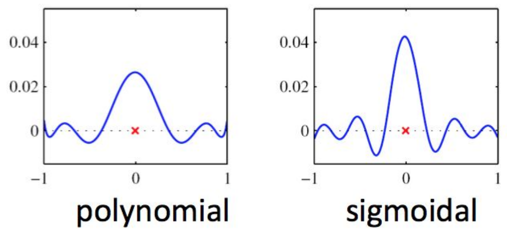


Figure1. local equivalent kernels among different basis functions

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.

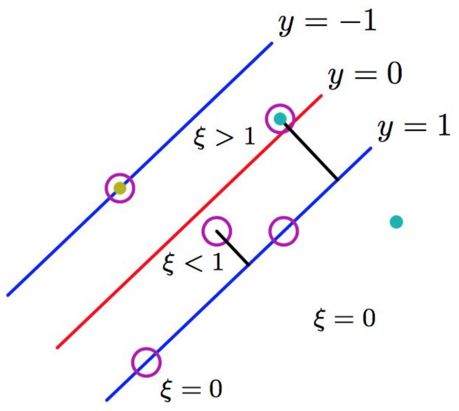
$$k(\mathbf{x}, \mathbf{z}) = \boldsymbol{\psi}(\mathbf{x})^{\mathrm{T}} \boldsymbol{\psi}(\mathbf{z})$$

where $\boldsymbol{\psi}(\mathbf{x}) = \beta^{1/2} \mathbf{S}_N^{1/2} \boldsymbol{\phi}(\mathbf{x})$

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 from C-SVM

Because in practice, the class-conditional distributions may overlap, we 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:

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

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

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

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

An alternative, equivalent formulation of the SVM

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

subject to the constraints

$$0 \leqslant a_n \leqslant 1/N \tag{7.39}$$

$$\sum_{n=1}^{N} a_n t_n = 0 \tag{7.40}$$

$$\sum_{n=1}^{N} a_n \geqslant \nu. \tag{7.41}$$

This approach has the advantage that the parameter **ν**, 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.

Performance

We found that the accuracy increases with *C* mostly. However, in polynomial function the increasing of degree makes the accuracy more sensitive to the parameter, *C* or *nu*, and harder for us to design the best.

Chart1. Error Rate Performance

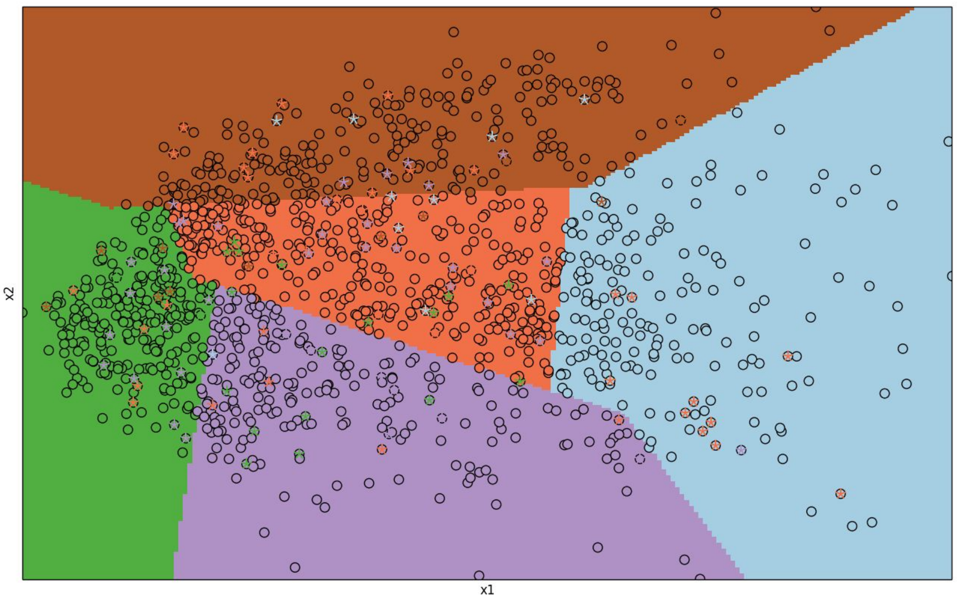
Kernel types		**ν**-SVM	C-SVM
Linear function		4.12%	5.00%
Polynomial function	2-Degree	2.44%	2.88%
	3-Degree	2.24%	2.96%
	4-Degree	5.72%	4.68%
Radial basis function		3.04%	3.04%

Supporting Vectors(SV) and Outliers

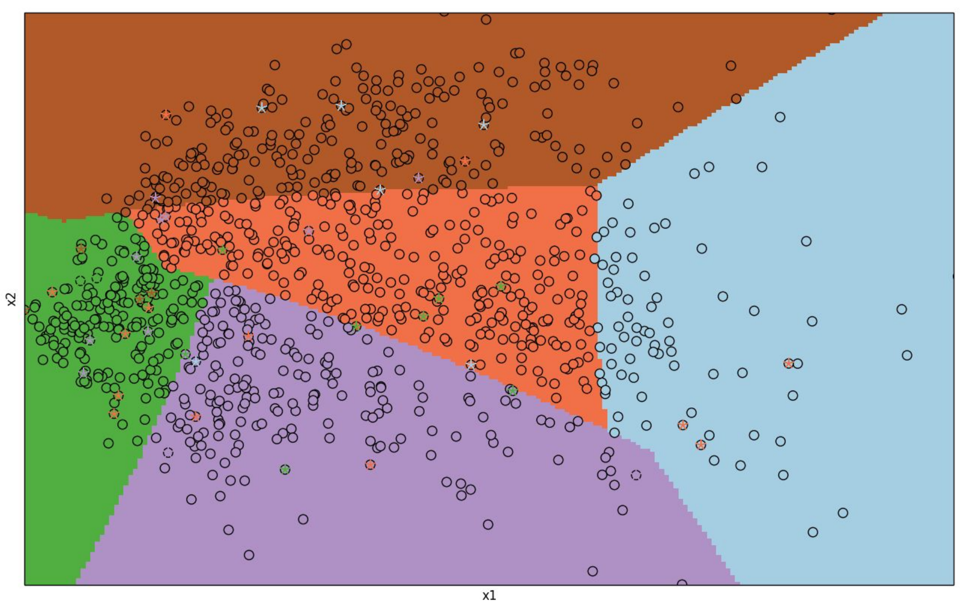
With Matplotlib, we do PCA on SV and Outliers, then visualize them with circles and star sign separately.

ν-SVM

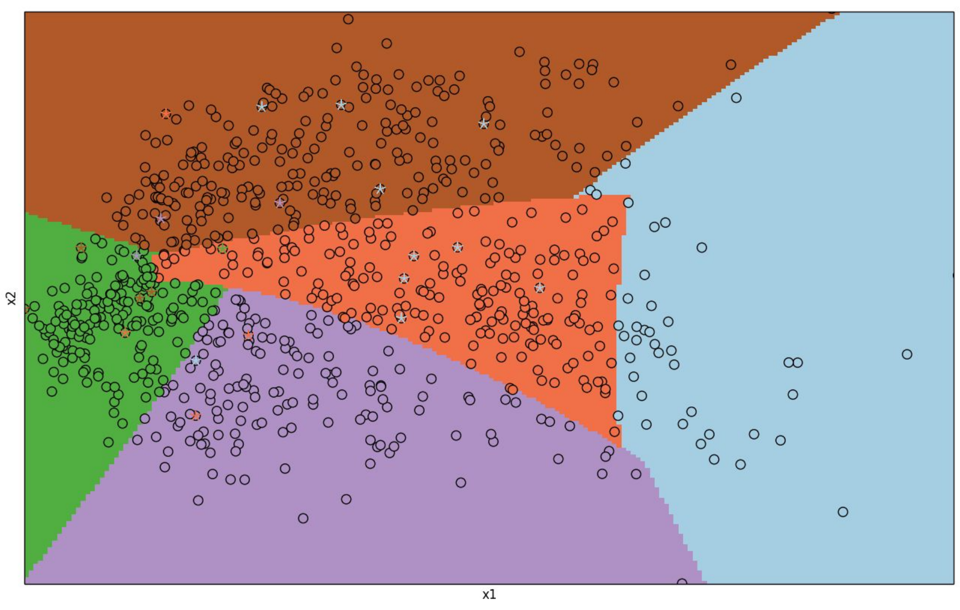
◦ Linear function: *nu*=0.11



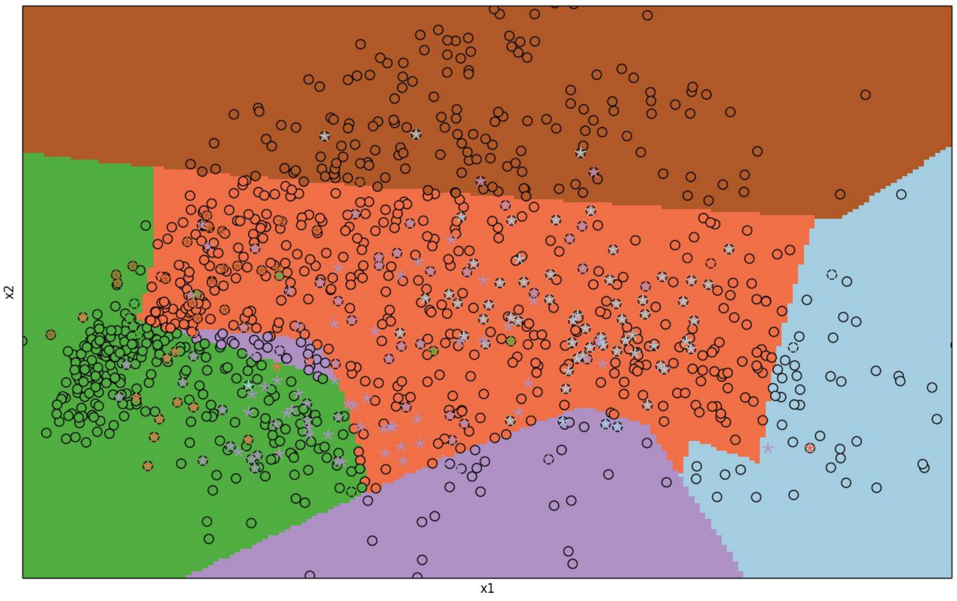
- Polynomial function with degree = 2: *nu*=0.09



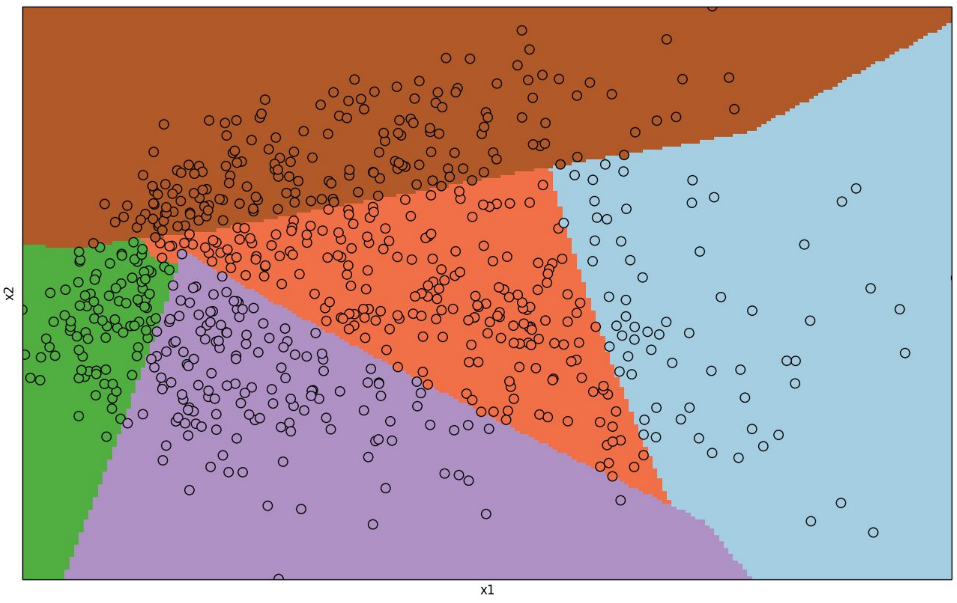
- Polynomial function with degree = 3: *nu*=0.09



- Polynomial function with degree = 4: *nu*=0.15213

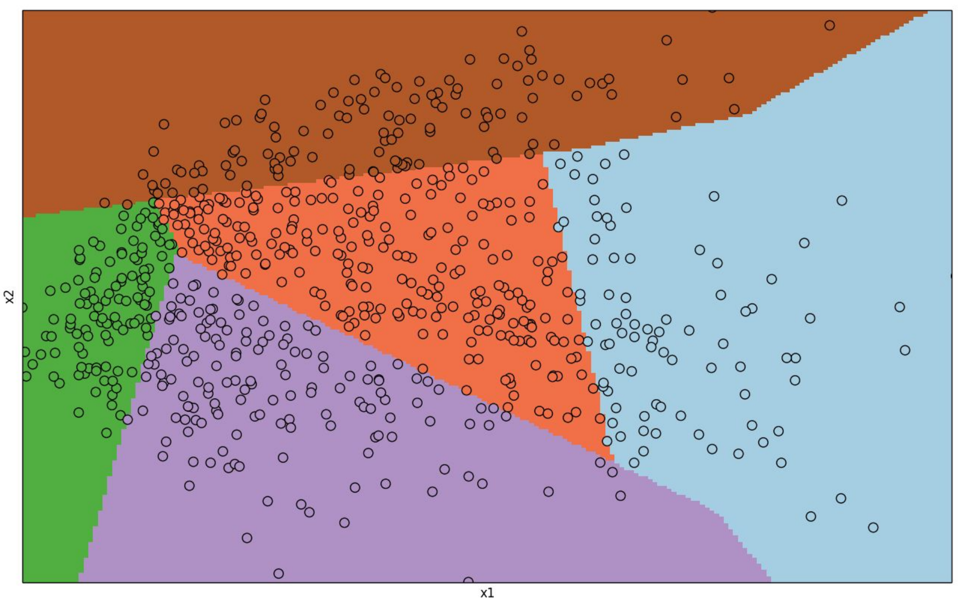


- Radial basis function: *nu*=0.03

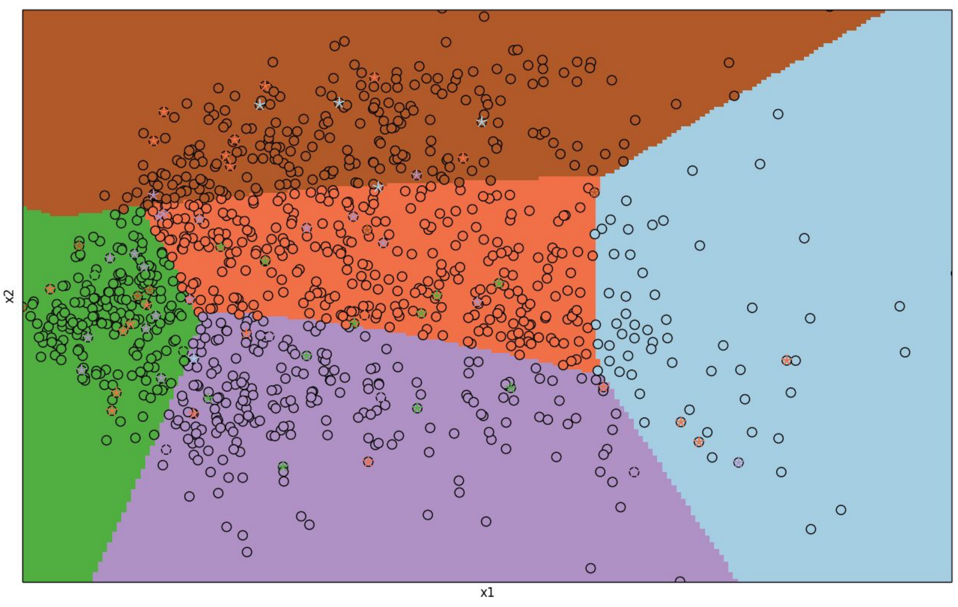


C-SVM

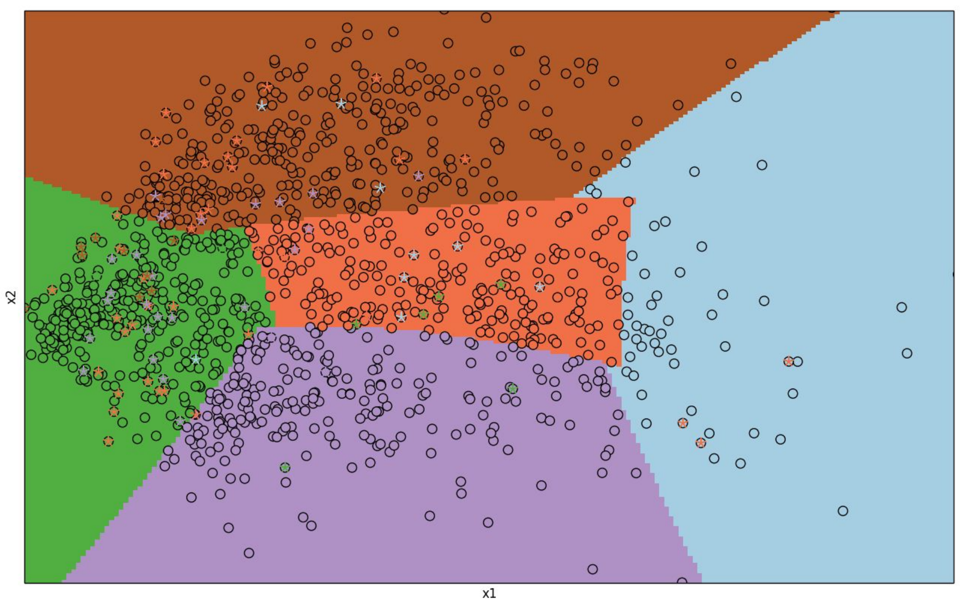
- Linear function: *C*=50



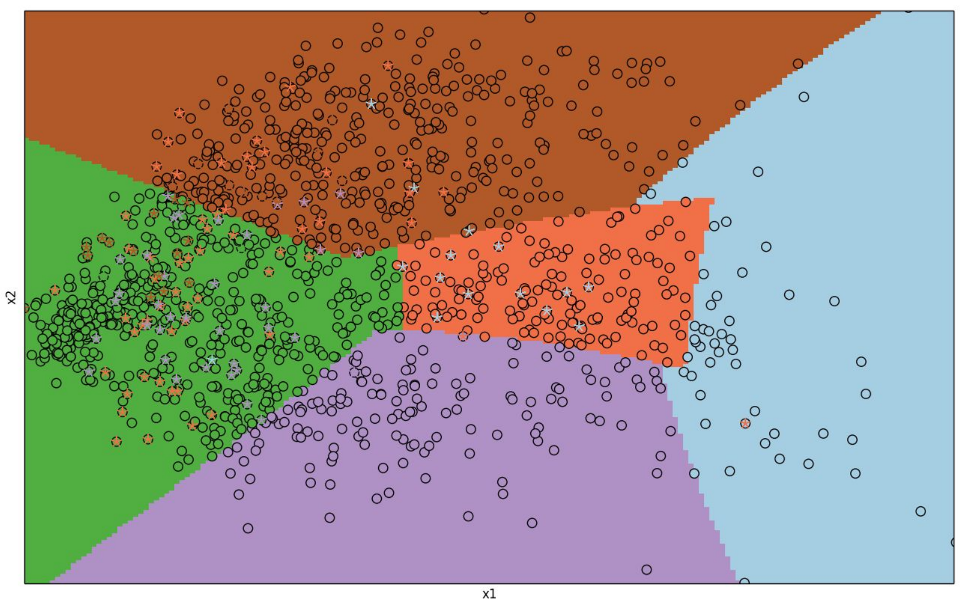
- Polynomial function with degree = 2: *C*=50



- Polynomial function with degree = 3: *C*=450



- Polynomial function with degree = 4: *C*=7000



- Radial basis function: *C*=50