Support Vector Machines

STAT 671: Statistical Learning

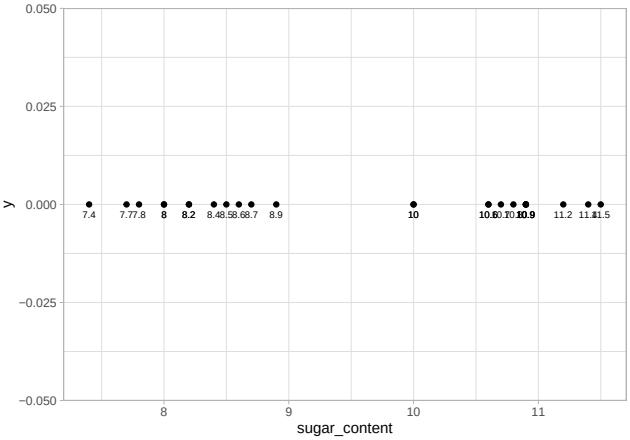
Di & Shen

- Kernel Methods:
  - Supervised Learning
    - Kernel ridge regression
    - Kernel logistic regression
    - Large-margin classifiers
    - Interlude: convex optimization and duality
    - Support vector machines

Introduction: A 1-dimension Example

25 soft drink sugar content measurements.

The distinct clusters for identifying candidate decision boundaries.



Find the maximal margin separator

Identified two distinct clusters (classes), the regular and the reduced sugar.

A dataset in which the classes do not overlap is called separable, the classes being separated by a decision boundary.

The maximal margin separator is the decision boundary that is furthest from both classes.

It is located at the mean of the relevant extreme points from each class.

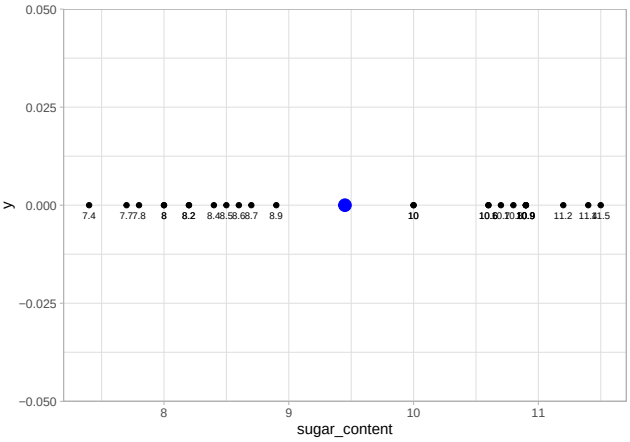
The relevant points are the highest valued point in the low sugar content class and the lowest valued point in the high sugar content class.

find the maximal margin separator for the sugar content dataset.

The maximal margin separator

```
#The maximal margin separator is at the midpoint of the two extreme points in each cluster.
mm_separator <- (8.9 + 10)/2
```

Add the maximal margin separator to the scatter plot.



Motivation

Consider the ℓ_2 regularized empirical risk function

$$J(\vec{w}, \lambda) = \sum i = 1^N L(y_i, \hat{y}_i) + \lambda \|\vec{w}\|^2$$

where $\hat{y}_i = \vec{w}^T \vec{x}_i + w_0$.

If L is quadratic loss, this is equivalent to ridge regression, and if L is the log-loss, this is equivalent to logistic regression.

If we replace the loss function with hinge loss function, we can ensure that the solution is sparse, so that predictions only depend on a subset of the training data, known as support vectors. This combination of the kernel trick plus a modified loss function is known as a support vector machine or SVM.

Definition

The hinge loss is the function $\mathbb{R} \to \mathbb{R}_+$:

$$\varphi_{hinge}(u) = \max(1 - u, 0) = \begin{cases} 0 & \text{if } u \geq 1, \\ 1 - u & \text{otherwise.} \end{cases}$$

SVM is the corresponding large-margin classifier, which solves:

$$\min_{f \in \mathcal{H}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \varphi_{hinge}(y_i f(x_i)) + \lambda \|f\|_{\mathcal{H}}^2 \right\}$$

Problem reformulation

By the representer theorem, the solution satisfies

$$\hat{f}(x) = \sum_{i=1}^{n} \hat{\alpha}_i K(x_i, x)$$

, where $\hat{\alpha}$ solves

$$\min_{\alpha \in \mathbb{R}^{\aleph}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \varphi_{hinge}(y_i [K\alpha]_i) + \lambda \alpha^T K \alpha \right\}$$

This is a convex optimization problem But the objective function is not smooth (because of the hinge loss)

Problem reformulation

Let us introduce additional slack variables $\xi_1, ..., \xi_n \in \mathbb{R}$. The problem is equivalent to:

$$\min_{\alpha \in \mathbb{R}^{\aleph}, \xi \in \mathbb{R}^{\aleph}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \xi_i + \lambda \alpha^T K \alpha \right\}$$

subject to:

$$\xi_i \geq \varphi_{hinge}(y_i [K\alpha]_i)$$

. The objective function is now smooth, but not the constraints However it is easy to replace the non-smooth constraint by a cunjunction of two smooth constraints, because:

$$u \geq \varphi_{hinge}(v) \iff \begin{cases} u \geq 1 - v \\ u \geq 0 \end{cases}$$

SVM (primal formulation)

In summary, the SVM solution is

$$\hat{f}(x) = \sum_{i=1}^{n} \hat{\alpha}_i K(x_i, x)$$

, where $\hat{\alpha}$ solves:

$$\min_{\alpha \in \mathbb{R}^{\aleph}, \xi \in \mathbb{R}^{\aleph}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \xi_i + \lambda \alpha^T K \alpha \right\}$$

subject to:

$$\begin{cases} (y_i [K\alpha]_i + \xi_i - 1 \geq 0, & \text{for} i = 1, ..., n, \\ \xi_i \geq 0, & \text{for} i = 1, ..., n \end{cases}$$

.

Solving the SVM problem

This is a classical quadratic program (minimization of a convex quadratic function with linear constraints) for which any out-of-the-box optimization package can be used.

The dimension of the problem and the number of constraints, however, are 2n where n is the number of points. General-purpose QP solvers will have difficulties when n exceeds a few thousands.

Solving the dual of this problem (also a QP) will be more convenient and lead to faster algorithms (due to the sparsity of the final solution).

SVM (dual formulation)

$$\max_{\alpha \in \mathbb{R}^n} \left\{ 2 \sum_{i=1}^{n} \alpha_i y_i - \sum_{i,j=1}^{n} \alpha_i \alpha_j K(x_i, x_j) \right\} = 2\alpha^T y - \alpha^T K \alpha$$

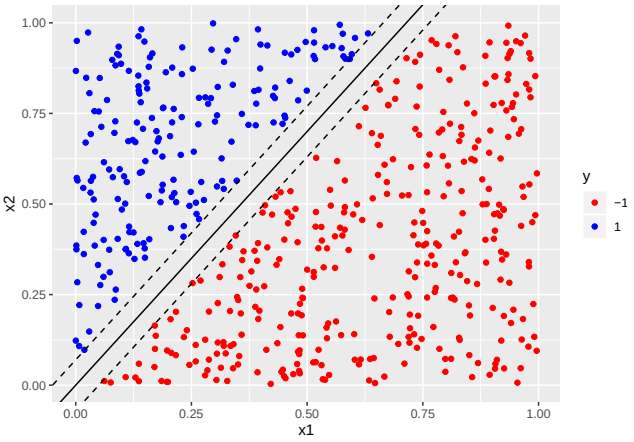
subject to:

$$0 \leq y_i \alpha_i \leq \frac{1}{2\lambda n}, \text{for } i = 1, ..., n$$

Complimentary slackness conditions

Analysis of KKT conditions

Geometric interpretation



Support vectors

- Consequence of KKT conditions

The training points with $\alpha_i \neq 0$ are called support vectors. Only support vectors are important for the classification of new points:

$$\forall x \in \mathcal{X}, \ f(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x) = \sum_{i \in SV} \alpha_i K(x_i, x)$$

where SV is the set of support vectors.

- Consequences

The solution is sparse in α, leading to fast algorithms for training (use of decomposition methods).

The classification of a new point only involves kernel evaluations with support vectors (fast).

Remark: C-SVM

Often the SVM optimization problem is written in terms of a regularization parameter C instead of λ as follows:

$$\arg\min_{f\in\mathcal{H}} \left\{ \frac{1}{2}\|f\|_{\mathcal{H}}^2 + C\sum_{i=1}^{n} L_{hinge}(f(x_i), y_i) \right\}$$

This is equivalent to our formulation with $C = \frac{1}{2\lambda n}$.

The SVM optimization problem is then:

$$\max_{\alpha\in\mathbb{R}} \left\{ 2\sum_{i=1}^{n} \alpha_i y_i - \sum_{i,j=1}^{n} \alpha_i \alpha_j K(x_i, x_j) \right\}$$

subject to: $0 \leq y_i\alpha_i \leq C, \text{for } i = 1, ..., n$ This formulation is often called C-SVM.

Remark: 2-SVM

A variant of the SVM, sometimes called 2-SVM, is obtained by replacing the hinge loss by the square hinge loss:

$$\min_{f\in\mathcal{H}} \left\{ \frac{1}{n}\sum_{i=1}^{n} \varphi_{hinge}(y_i f(x_i))^2 + \lambda\|f\|_{\mathcal{H}}^2 \right\}$$

The dual problem of the 2-SVM is:

$$\max_{\alpha\in\mathbb{R}} \left\{ 2\alpha^T y - \alpha^T (K + n\lambda I)\alpha \right\}$$
$$0 \leq y_i\alpha_i, \text{ for } i = 1, ..., n$$

This is therefore equivalent to the previous SVM with the kernel $K + n\lambda I$ and $C = +\infty$

Another version: SVMs for classification

Primal form

Representation

$$\mathcal{H} : y = f(\vec{x}) = \text{sign}(\vec{w}\vec{x} + b)$$

Evaluation

$$\min_{\vec{w},b} \frac{1}{2}\|\vec{w}\|^2 \quad \text{s.t.} \quad y_i(\vec{w}\vec{x}_i + b) \geqslant 1, i = 1, 2, \ldots, N$$

Dual form

Representation

$$\mathcal{H} : y = f(\vec{x}) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i (\vec{x} \cdot \vec{x}_i) + b\right)$$

Evaluation

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (\vec{x}_i \cdot \vec{x}_j) - \sum_{i=1}^{N} \alpha_i$$

s.t. $\sum_{i=1}^{N} \alpha_i y_i = 0 \quad \alpha_i \geqslant 0, i = 1, 2, \ldots, N$

Primal form with slack variables

Representation

$$\mathcal{H} : y = f(\vec{x}) = \text{sign}(\vec{w}\vec{x} + b)$$

Evaluation

$$\min_{\vec{w},b} C \sum_{i=1}^{N} \xi_i + \frac{1}{2} \|\vec{w}\|^2$$

s.t.

$$y_i(\vec{w}\vec{x}_i + b) \geqslant 1 - \xi_i \quad \xi_i \geqslant 0, \quad i = 1, 2, \ldots, N$$

Dual form with slack variables

Representation

$$\mathcal{H} : y = f(\vec{x}) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i (\vec{x} \cdot \vec{x}_i) + b\right)$$

Evaluation

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (\vec{x}_i \cdot \vec{x}_j) - \sum_{i=1}^{N} \alpha_i$$

s.t.

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \quad 0 \leqslant \alpha_i \leqslant C, i = 1, 2, \ldots, N$$

$$\begin{aligned} \alpha_i = 0 &\Rightarrow y_i(\vec{w} \cdot \vec{x}_i + b) \geqslant 1 \\ \alpha_i = C &\Rightarrow y_i(\vec{w} \cdot \vec{x}_i + b) \leqslant 1 \\ 0 < \alpha_i < C &\Rightarrow y_i(\vec{w} \cdot \vec{x}_i + b) = 1 \end{aligned}$$

Hinge Loss

Linear support vector machines can also be interpreted as hinge loss minimization:

$$\min_{\vec{w},b} \sum_{i=1}^{N} L(y_i, f(\vec{x}_i)) + \lambda \|\vec{w}\|^2$$

where $L(y, f(\vec{x}))$ is a hinge loss function:

$$L(y, f(\vec{x})) = \begin{cases} 1 - yf(x), & 1 - yf(x) > 0 \\ 0, & 1 - yf(x) \leqslant 0 \end{cases}$$

Introduction

Some basic concepts of support vector machines.

Example 2: A 2d dataset.

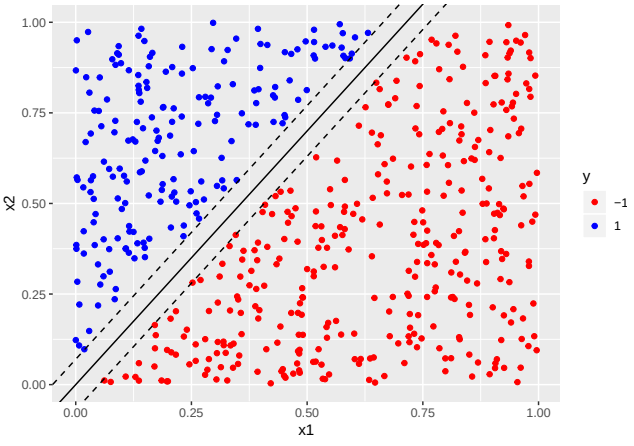
A 2 dimensional uniformly distributed dataset containing 600 datapoints.

Create a decision boundary

Add a class variable to that dataset by creating a variable y whose value is -1 or +1 depending on whether the point (x1, x2) lies below or above the straight line that passes through the origin and has slope 1.4.

Introduce a margin in the dataset

Create a margin in the dataset and then display the margin in a plot. The slope of the linear decision boundary is 1.4.



Support Vector Classifiers - Linear Kernels

Split into training and test sets

The dataset generated in previous chapter is in dataframe df.

Split dataset into training and test sets

Random 80/20 split

Decision boundaries and kernels

Decision boundaries can have different shapes - lines, polynomials or more complex functions.

Type of decision boundary is called a kernel.

Kernel must be specified upfront.

This chapter focuses on linear kernels.

SVM with Linear Kernel

We'll use the svm function from the e1071 library.

The function has a number of parameters. We'll set the following explicitly:

formula - a formula specifying the dependent variable. y in our case.

data - dataframe containing the data - i.e. trainset.

type - set to C-classification(classification problem).

kernel - this is the form of the decision boundary, linear in this case.

cost and gamma - these are parameters that are used to tune the model.

scale - Boolean indicating whether to scale data.

Building a Linear SVM

Model Accuracy

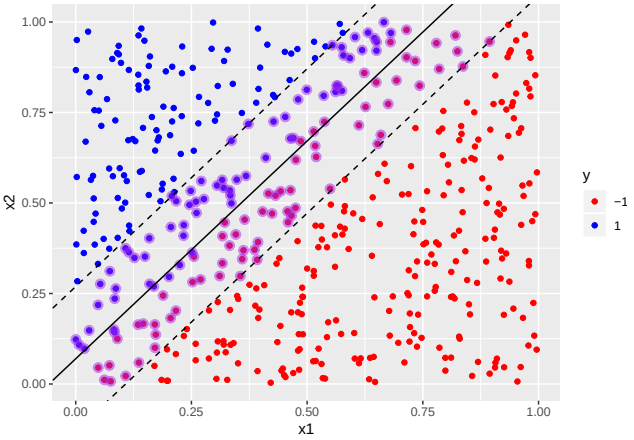
Obtain class predictions for training and test sets.

Evaluate the training and test set accuracy of the model.

```
## [1] 0.975
```

```
## [1] 0.974
```

Visualizing support vectors



Soft margin classifiers

Allow for uncertainty in location / shape of boundary

Never perfectly linear

Usually unknown

Our decision boundary is linear, so we can reduce margin

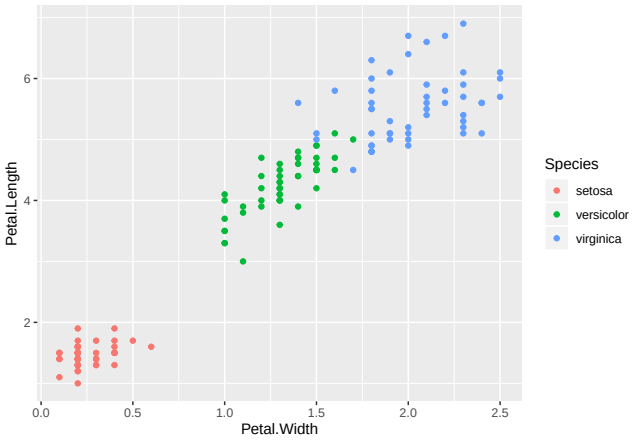
Tuning linear SVMs

```
##
## Call:
## svm(formula = y ~ ., data = trainset, type = "C-classification",
##     kernel = "linear", cost = 100, scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  100
##
## Number of Support Vectors:  30
```

```
## [1] 1
```

Multiclass problems

150 measurements of 5 attributes;
Petal width and length - number (predictor variables);
Sepal width and length - number (predictor variables);
Species - category: setosa, virginica or versicolor (predicted variable)



How does the SVM algorithm deal with multiclass problems?

SVMs are essentially binary classifiers.

Can be applied to multiclass problems using the following voting strategy:

- Partition the data into subsets containing two classes each.
- Solve the binary classification problem for each subset.
- Use majority vote to assign a class to each data point.

Called one-against-one classification strategy.

Polynomial Kernels

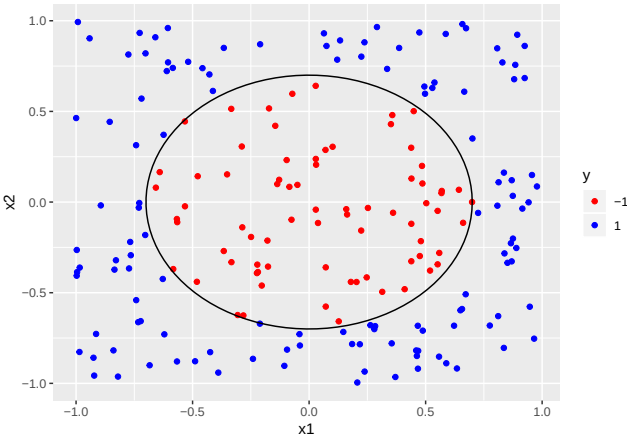
Generating a radially separable dataset

Generating a 2d uniformly distributed set with 200 points

2 predictors x1 and x2, uniformly distributed between -1 and 1.

Create a circular decision boundary of radius 0.7 units

Categorical variable y is +1 or -1 depending on the point lies outside or within boundary

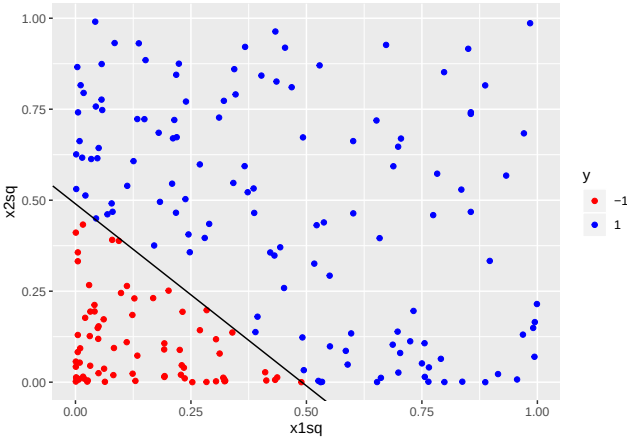


Transforming the problem

Equation of boundary is x1^2 + x2^2 = 0.49

Map x1^2 to a new variable X1 and x2^2 to X2

The equation of boundary in the X1-X2 space becomes... X1 + X2 = 0.49 (a line!!)



The Polynomial Kernel - Part 1

Polynomial kernel: (gamma*(u.v)+coef0)^degree

- degree = degree of polynomial
- gamma and coef0- tuning parameters
- u, v - vectors (datapoints) belonging to the dataset

We can guess we need a 2nd degree polynomial (transformation)

Kernel functions

The math formulation of SVMs requires transformations with specific properties.

Functions satisfying these properties are called kernel functions

Kernel functions are generalizations of vector dot products

Basic idea* - use a kernel that separates the data well!

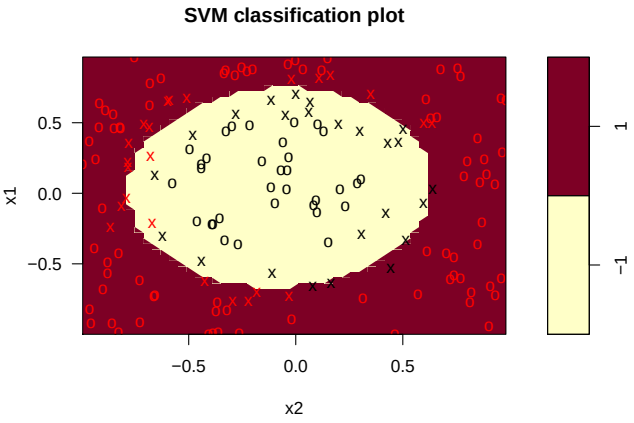
Radially separable dataset - quadratic kernel

80/20 train/test split

Build a quadratic SVM for the radially separable dataset:

- degree =2
- default values of cost, gamma and coef0 (1, 1/2 and 0)

```
## [1] 0.872
```



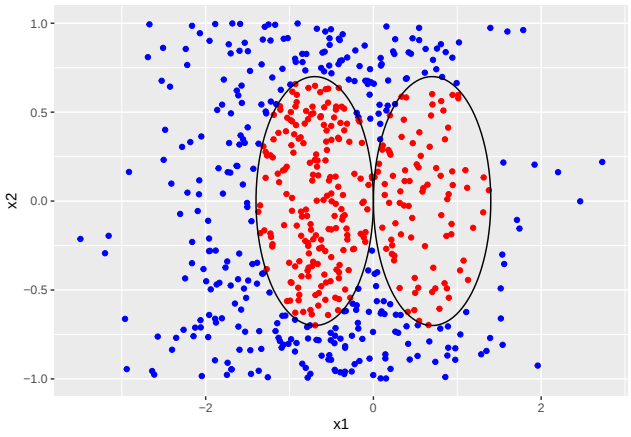
Tuning SVMs

Omitted

Radial Basis Function Kernels

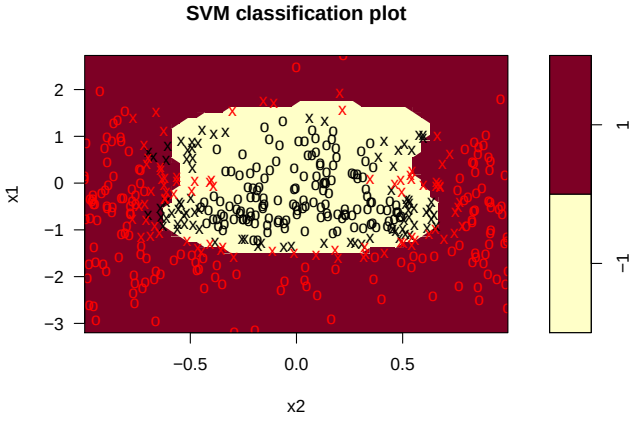
Generate a complex dataset

600 points (x1,x2) x1 and x2 distributed differently

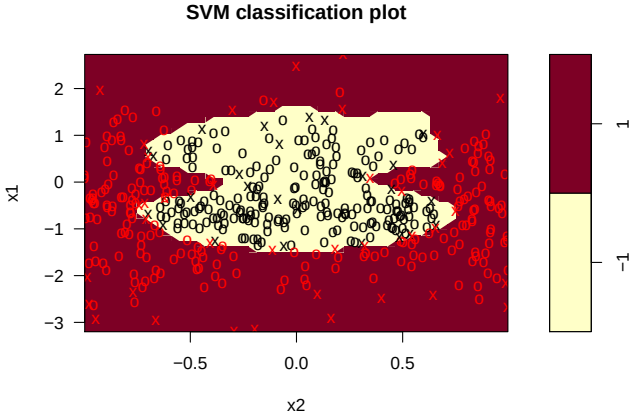


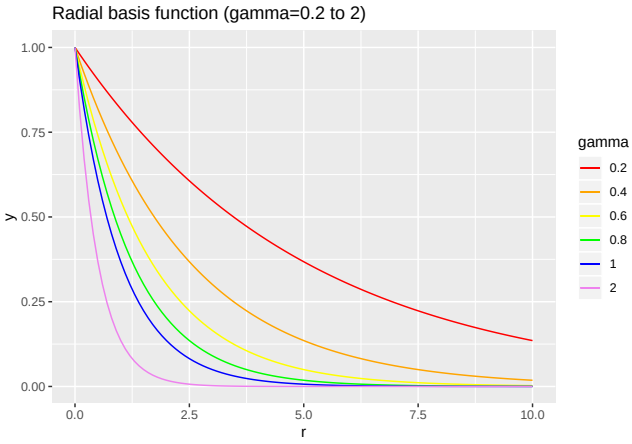

```
## [1] 0.933
```

```
## [1] 0.935
```




```
## [1] 0.935
```





Note

Historically the first "kernel method" for pattern recognition, still the most popular.

Often state-of-the-art in performance.

One particular choice of loss function (hinge loss).

Leads to a sparse solution, i.e., not all points are involved in the decomposition (compression).

Particular algorithm for fast optimization (decomposition by chunking methods).

Note

SVMs are very unnatural from a probabilistic point of view.

- First, they encode sparsity in the loss function rather than the prior.
- Second, they encode kernels by using an algorithmic trick, rather than being an explicit part of the model.
- Finally, SVMs do not result in probabilistic outputs, which causes various difficulties, especially in the multi-class classification setting.

It is possible to obtain sparse, probabilistic, multi-class kernel-based classifiers, which work as well or better than SVMs, using techniques such as the L1VM or RVM.

Summary

SVM classifiers involve three key ingredients: the kernel trick, sparsity, and the large margin principle .

The kernel trick is necessary to prevent underfitting, i.e., to ensure that the feature vector is sufficiently rich that a linear classifier can separate the data.

If the original features are already high dimensional, it suffices to use a linear kernel, $K(x, x') = x^T x'$, which is equivalent to working with the original features.

The sparsity and large margin principles are necessary to prevent overfitting, i.e., to ensure that we do not use all the basis functions. These two ideas are closely related to each other,and both arise (in this case) from the use of the hinge loss function.

However, there are other methods of achieving sparsity (such as ℓ_1), and also other methods of maximizing the margin(such as boosting).

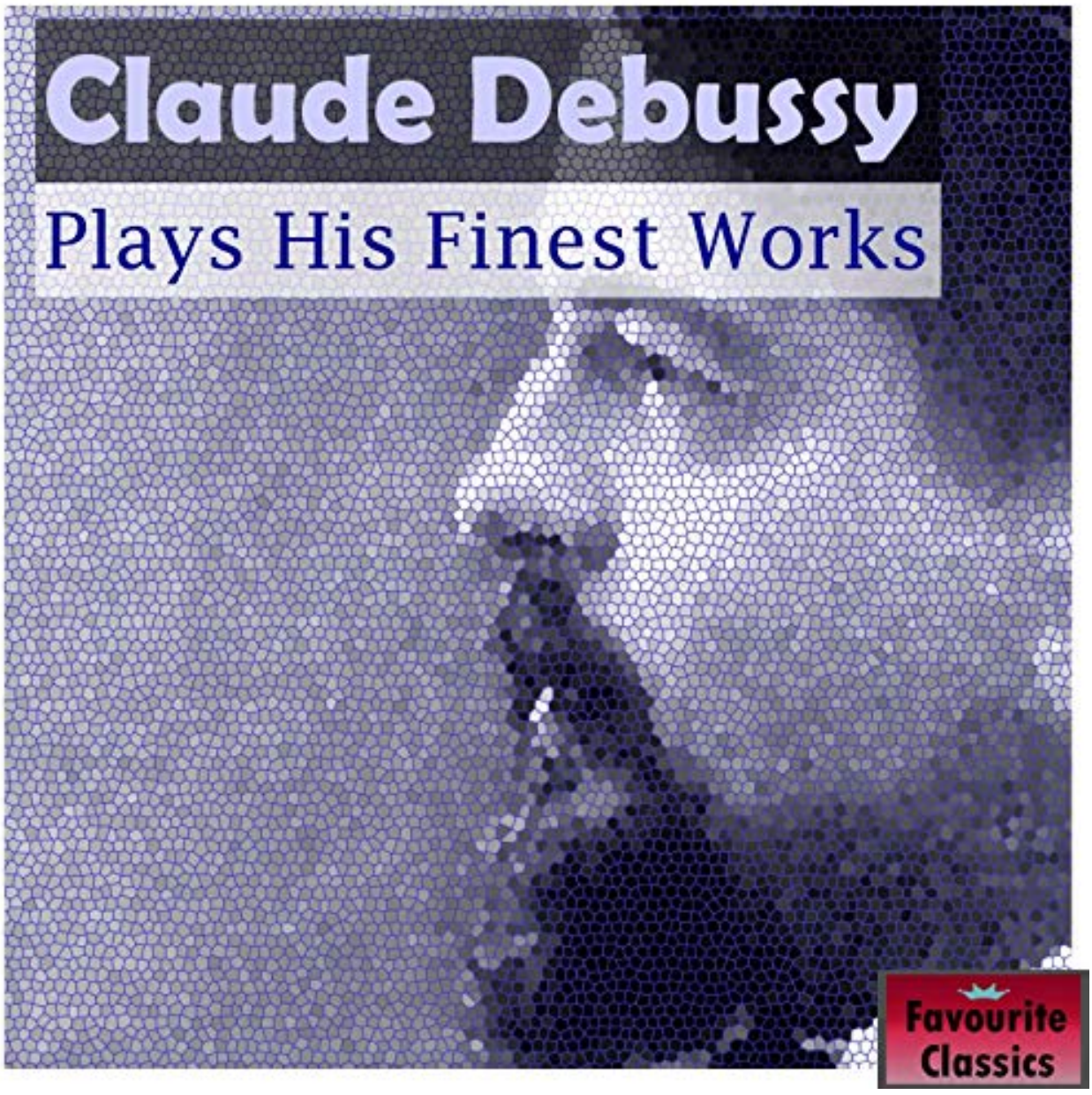


Figure 1: Reverie

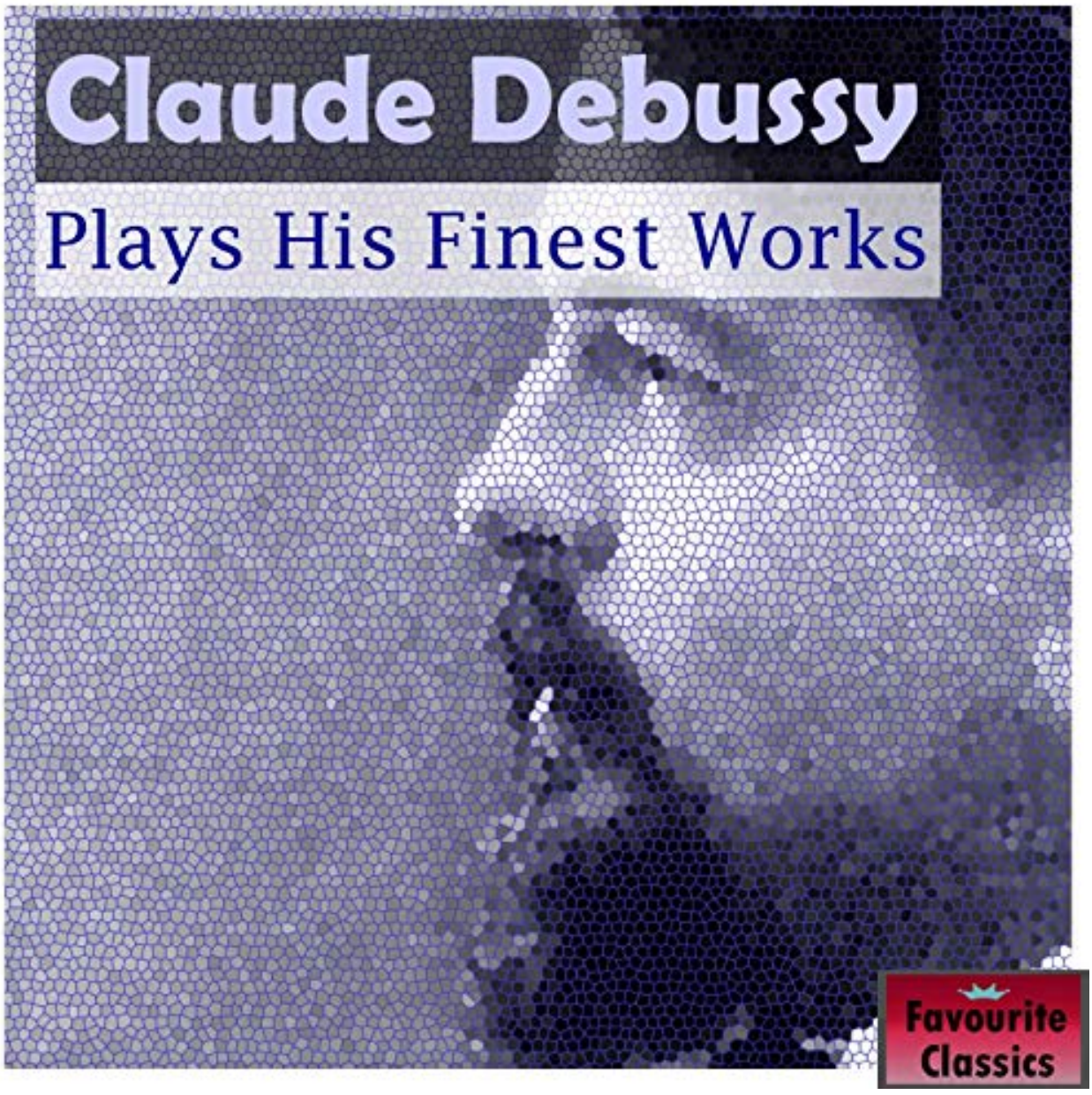


Figure 2: Reverie