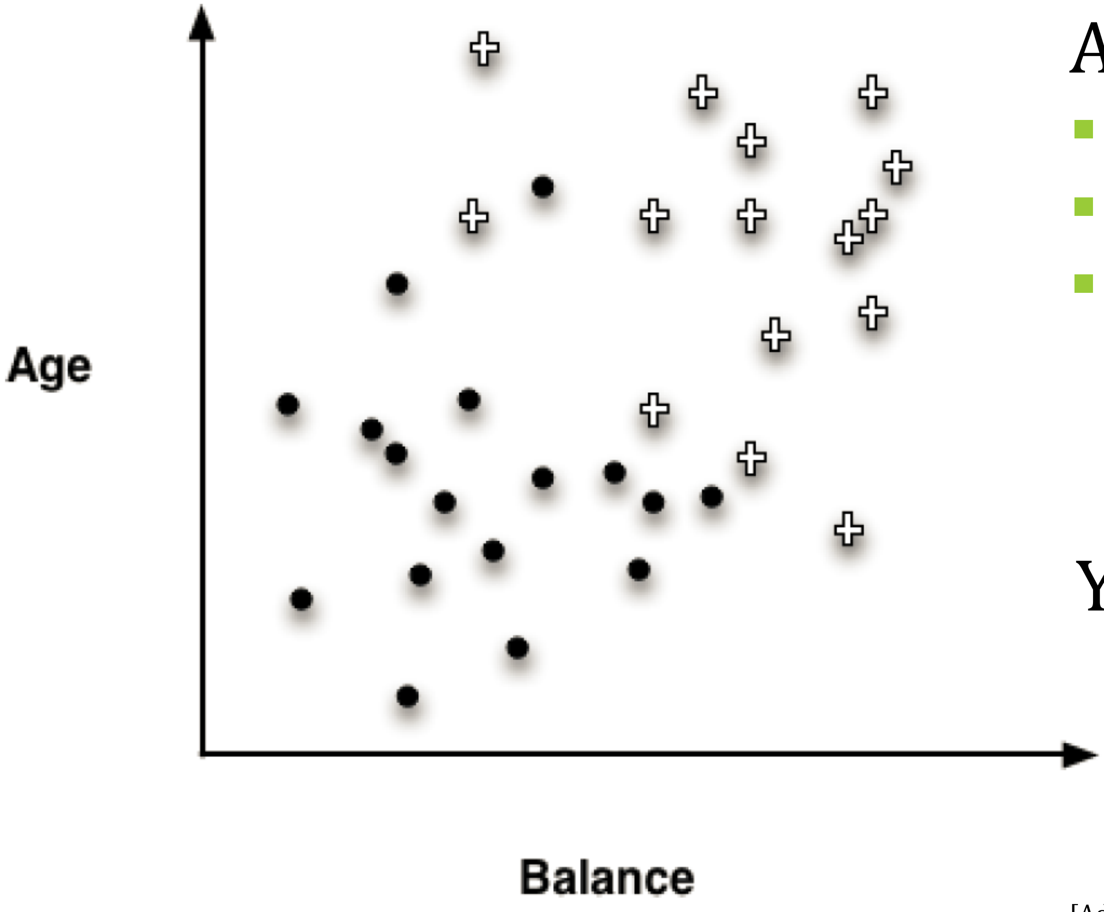
Support Vector Machines

Overview
Classifying – What Do You See?



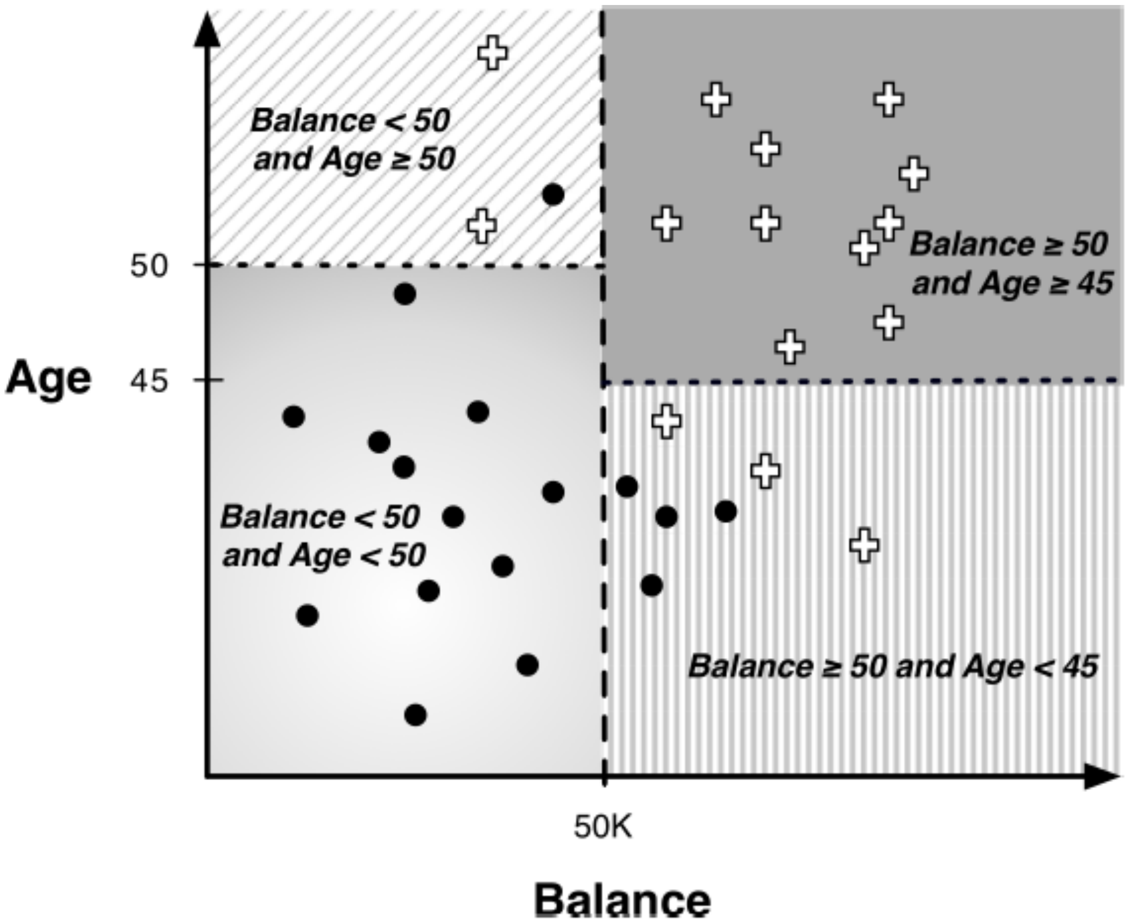
Artificial data, with 3 features:

- **age** of a customer (*numerical*)
- savings **balance** (*numerical*)
- whether or not they **defaulted** on a mortgage loan (*categorical*; · for default, + for no default).

Young borrowers with small balances
vs.
Old borrowers with large balances?

[Adapted from Foster & Provost's *Data Science for Business*]

Overview
Classifying Using a Decision Tree

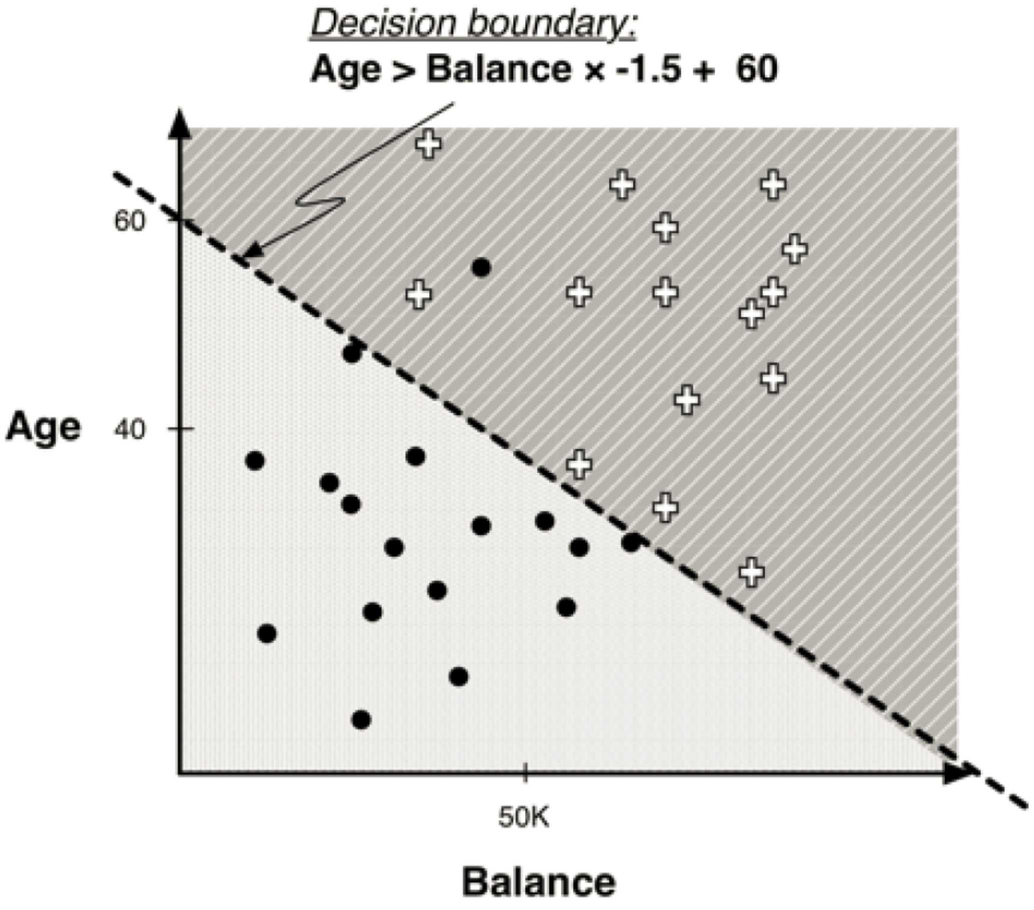


Simple **decision rule**:

- balance below 50,000$;
  50 y.o. or younger;
  default in 100% of the cases
- balance above 50,000$;
  45 y.o. or younger;
  default in 0% of the cases
- balance below 50,000$;
  50 y.o. or older;
  default in 33% of the cases
- balance above 50,000$;
  45 y.o. or younger;
  default in 57% of the cases

Overview

Classifying Using Decision Boundary



Simpler **decision rule**:

- (Balance,Age) below the boundary; default in 100% of the cases
- (Balance,Age) above the boundary; default in 7% of the cases

A **single borrower** is misclassified by this rule.

Perfect accuracy can be reached with non-linear curves, but that leads to **over-fitting**.

Overview

Support Vector Machines in a Nutshell

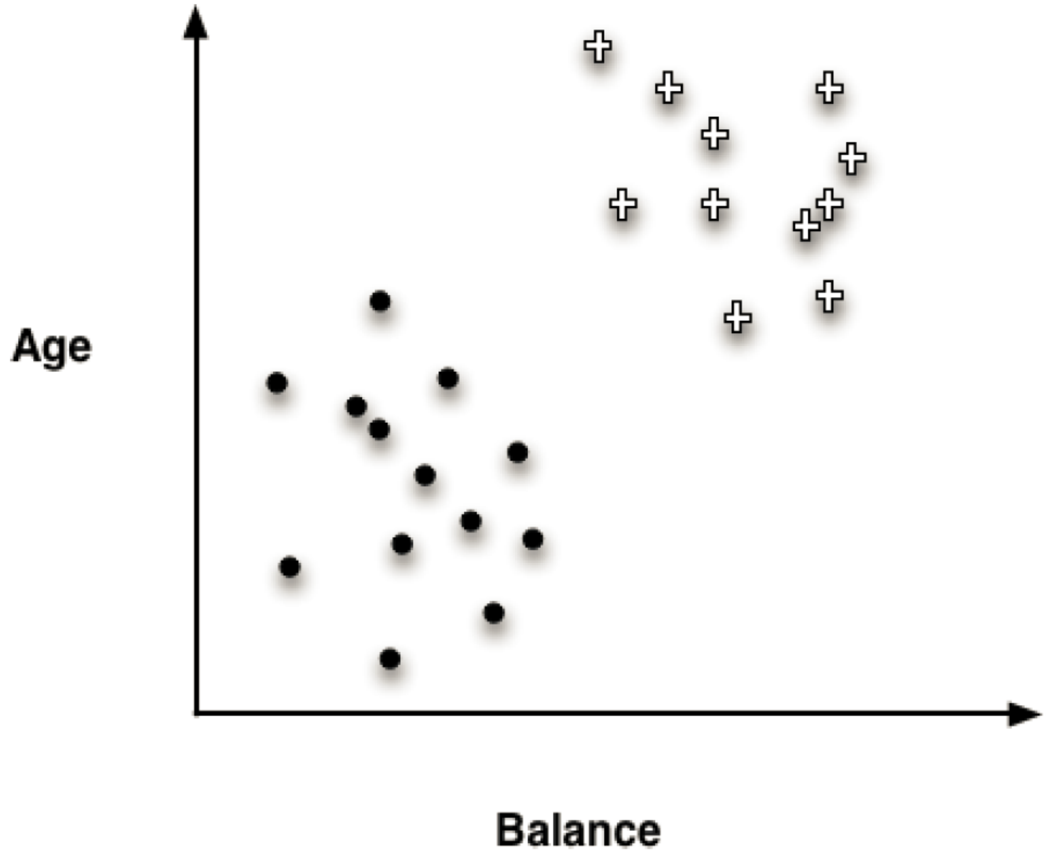
Support Vector Machines (SVMs) provide a protocol to train classifiers, using non-linear hyper-surfaces as required.

Not all data can be cleanly separated: the “best” classifier is the one which **minimizes the cost of making errors**.

SVMs are **numerically efficient** as they are typically trained on a small subset of the available observations.

SVMs have been **applied to**: text categorization, image classification, hand-writing recognition, smoothing and regression, outlier detection, (clustering?)

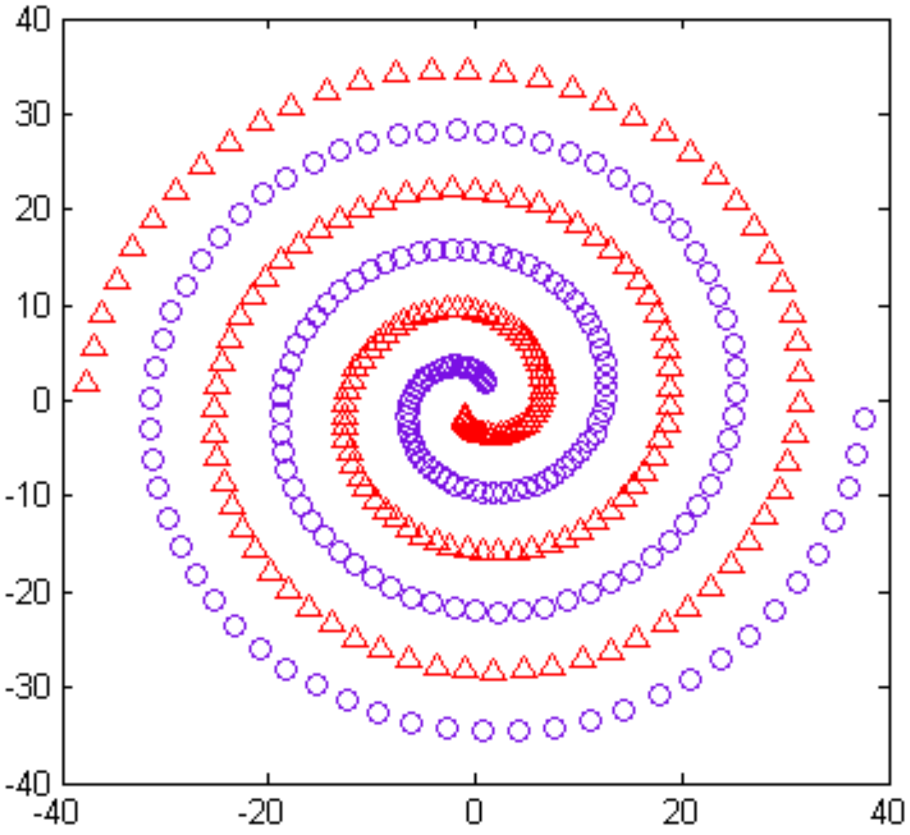
SVM Classification
Linear Discriminants



Here is a modified mortgage default dataset (this one is **linearly separable**).

A straight line that cleanly separates the classes is called a **linear discriminant** (or a **separating hyperplane** in the case of multivariate data).

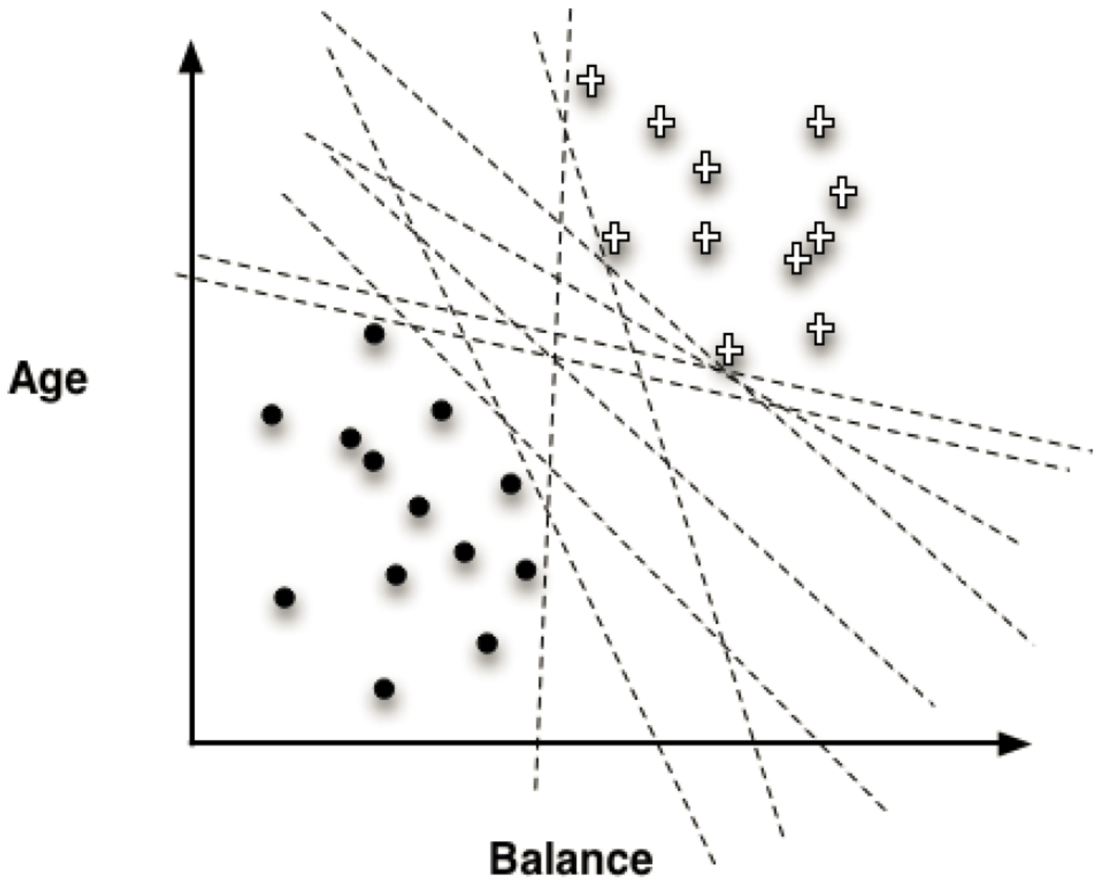
SVM Classification
Linear Discriminants



This spiral dataset is **not** linearly separable.

Linear separation is in some sense **maximally violated**.

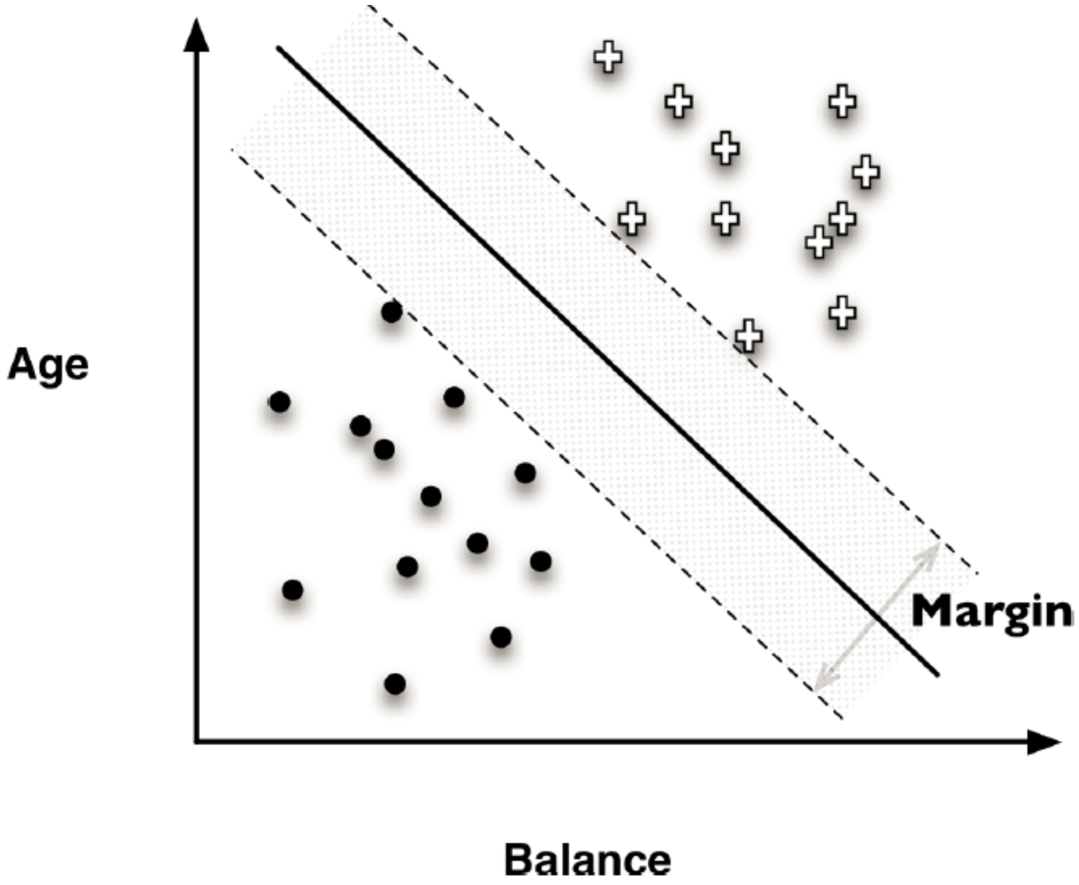
SVM Classification
Linear Discriminants



Linear discriminants need not be **unique**.

So which one is **optimal**?

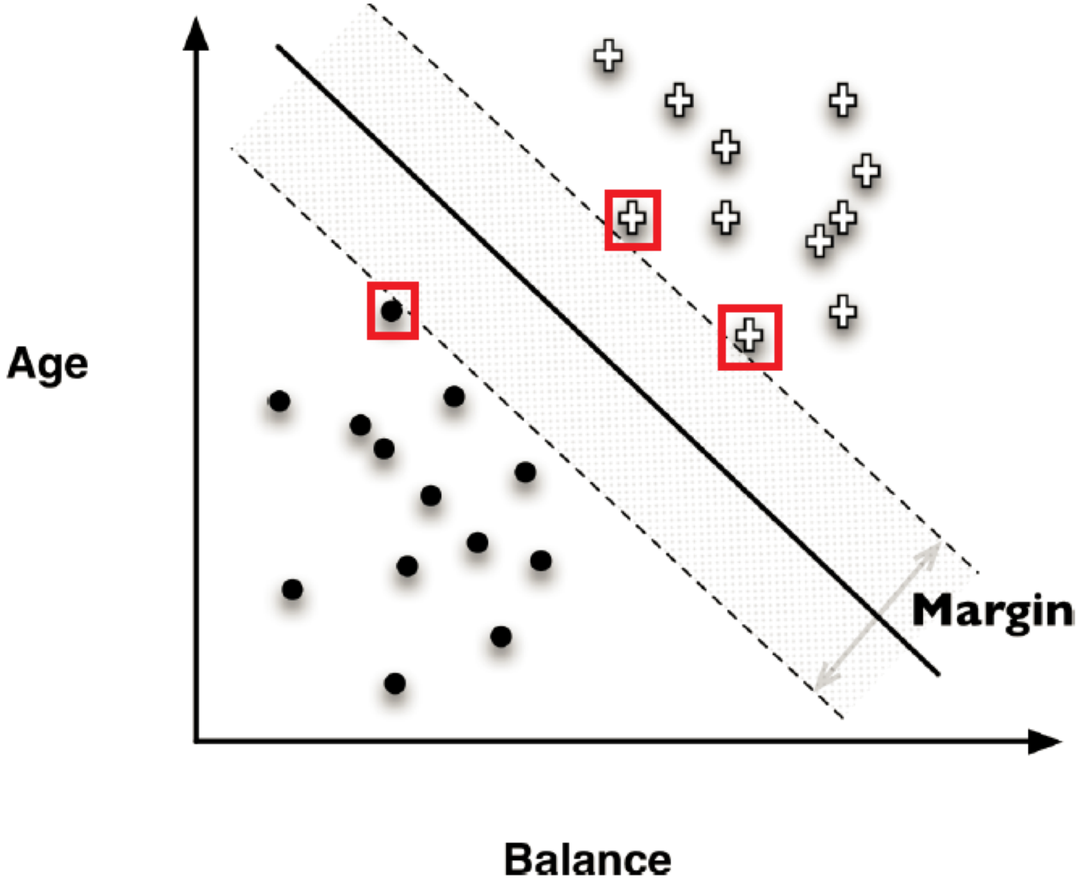
SVM Classification
Maximum-Margin Hyperplane



The **maximum-margin hyper-plane** is the “best” separating hyperplane — it is the one at the **center of the largest strip which separates the data points cleanly**.

Objective: maximize the margin to find the **optimal hyperplane**.

SVM Classification
Support Vectors



The **support vectors** are the observations closest to the margin on each side.

Usually, the number of support vectors is small; or at least relatively smaller than the number of observations.

SVM Classification
Linear Kernel – Formulation

1. Let $\boldsymbol{x}_i$ (vector) represent the data, with y_i (scalar) being the known classification of n observations.
2. The general equation of a hyperplane is given by

$$f(\boldsymbol{x}) = \beta_0 + \boldsymbol{\beta}^T \boldsymbol{x} = 0,$$

   where β_0 is the **bias** (intercept) and $\boldsymbol{\beta}$ is the **weight vector**.
3. There are infinitely many ways to express that equation: the **canonical hyperplane** is the one such that

$$|\beta_0 + \boldsymbol{\beta}^T \boldsymbol{x}| = 1,$$

   for any (eventual) support vector $\boldsymbol{x}$ in the training set.

SVM Classification
Linear Kernel – Formulation

4. From geometry, the distance from any point $\boldsymbol{z}$ to the canonical hyperplane is
$$\frac{|\beta_0 + \boldsymbol{\beta}^T \boldsymbol{z}|}{||\boldsymbol{\beta}||}.$$
5. If $\boldsymbol{z}$ is a support vector, that distance becomes
$$\frac{1}{||\boldsymbol{\beta}||}.$$

SVM Classification
Linear Kernel – Formulation

6. The margin M is twice that distance:

$$M = \frac{2}{||\boldsymbol{\beta}||}.$$

7. Finding the parameters $(\beta_0, \boldsymbol{\beta})$ which maximize M is equivalent to solving the QP:

$$\arg\min_{(\beta_0,\boldsymbol{\beta})} \left\{ \frac{1}{2} ||\boldsymbol{\beta}||^2 : \text{subject to } \ y_i(\beta_0 + \boldsymbol{\beta}^T \boldsymbol{x}_i) \geq 1 \text{ for each } \ \boldsymbol{x}_i \right\}$$

8. The constrained QP can be solved with Lagrange Multipliers.

SVM Classification
Linear Kernel – Dual Formulation

The **Representer Theorem** shows that we can write

$$\boldsymbol{\beta} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i \quad \text{with} \quad \sum_{i=1}^{n} \alpha_i y_i = 0.$$

The original QP is equivalent to solving:

$$\arg\min_{\boldsymbol{\alpha}} \left\{ \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j \boldsymbol{x}_i^T \boldsymbol{x}_j - \sum_{i=1}^{n} \alpha_i : \text{subj. to} \sum_{i=1}^{n} \alpha_i y_i = 0 \ \text{ for each } \ \boldsymbol{x}_i \right\}$$

The L support vectors $\boldsymbol{x}_{i_k}$ are those for which $\alpha_{i_k} \neq 0$.

SVM Classification
Linear Kernel – Decision Function

The **decision function** is defined by

$$T(\boldsymbol{x};\boldsymbol{\alpha}) = \sum_{k=1}^{L} \alpha_{i_k} y_{i_k} \boldsymbol{x}_{i_k}^T \boldsymbol{x} + \beta_0,$$

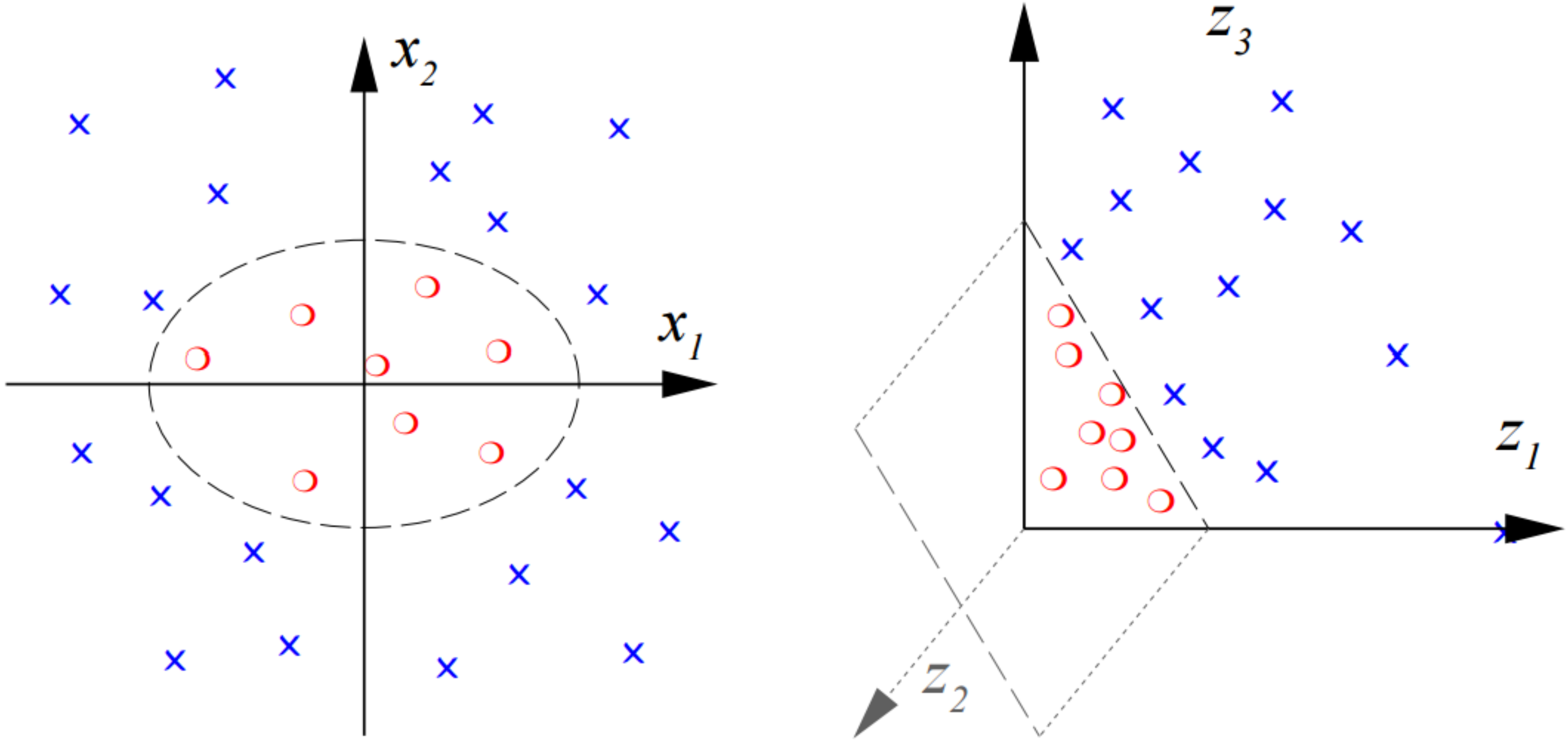
which is scaled so that

$$T\left(\boldsymbol{x}_{i_k};\boldsymbol{\alpha}\right)=y_{i_k}(=\pm 1)$$

for each support vector $\boldsymbol{x}_{i_k}$.

Class assignment for $\boldsymbol{x}$:

- If $T(\boldsymbol{x};\boldsymbol{\alpha}) > 0$ then $\text{class}(\boldsymbol{x}) = +1$
- If $T(\boldsymbol{x};\boldsymbol{\alpha}) < 0$ then $\text{class}(\boldsymbol{x}) = -1$



[Adapted from Watson's *Support Vector Machine Tutorial*]

SVM Classification
Transformations

The best **separating strip** may not be linear (underlying data structure is complex).

Solution: introduce transformation Φ from initial feature space to high-dimensional space, and train linear SVM to data $\boldsymbol{z_i} = \Phi(\boldsymbol{x}_i)$.

SVM Classification

Transformation – Dual Formulation and Decision Function

Simply replace $\boldsymbol{x} \leftrightsquigarrow \Phi(\boldsymbol{x})$ throughout.

Dual formulation:

$$\arg\min_{\boldsymbol{\alpha}} \left\{ \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j \Phi(\boldsymbol{x}_i)^T \Phi(\boldsymbol{x}_j) - \sum_{i=1}^{n} \alpha_i : \text{s.t.} \sum_{i=1}^{n} \alpha_i y_i = 0 \ \text{ for each } \boldsymbol{x}_i \right\}$$

SVM Classification
Transformation – Dual Formulation and Decision Function

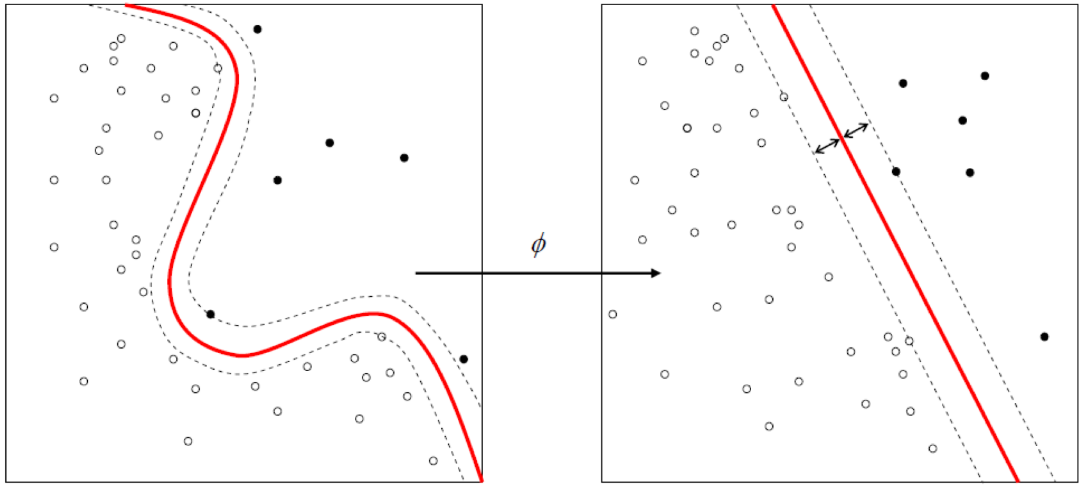
The decision function is defined by

$$T(\boldsymbol{x};\boldsymbol{\alpha}) = \sum_{k=1}^{L} \alpha_{i_k} y_{i_k} \Phi(\boldsymbol{x}_{i_k})^T \Phi(\boldsymbol{x}) + \beta_0,$$

for each support vector $\boldsymbol{x}_{i_k}$, and is scaled as above.

Class assignment for $\boldsymbol{x}$ follows the same rule.

SVM Classification
Nonlinear Kernels



Would you have recognized
$$\Phi(x_1, x_2) = \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)$$
as the "right" transformation in the previous example?

What about for this dataset?

[Adapted from Wikipedia]

It is not always feasible to find the proper transformation in a reasonable amount of time.

What then?

SVM Classification
Nonlinear Kernels

A **kernel function** $K(\boldsymbol{x}, \boldsymbol{w})$ is a function that generalizes the dot product $\Phi(\boldsymbol{x})^T \Phi(\boldsymbol{w})$ (in particular, it needs to be **semi-positive definite)**.

With the right kernel, we might be able to bypass the need to know Φ exactly, although we need to specify some kernel constants.

Commonly-used kernels include

- **simple linear**: $K(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{x}^T \boldsymbol{w}$
- **polynomial**: $K(\boldsymbol{x}, \boldsymbol{w}) = (\boldsymbol{x}^T \boldsymbol{w} + c)^d$, where $c \in \mathbb{R}$, $d \in \mathbb{N}$
- **gaussian**: $K(\boldsymbol{x}, \boldsymbol{w}) = \exp(-\gamma \|\boldsymbol{x} - \boldsymbol{w}\|^2)$, where $\gamma > 0$ (often the default kernel in applications)
- **sigmoïd**: $K(\boldsymbol{x}, \boldsymbol{w}) = \tanh(\kappa \boldsymbol{x}^T \boldsymbol{w} - \delta)$, for allowable κ, δ

SVM Classification
Nonlinear Kernels – Dual Formulation and Decision Function

Simply replace $\Phi(\boldsymbol{x})^T\Phi(\boldsymbol{w}) \leftrightsquigarrow K(\boldsymbol{x}, \boldsymbol{w})$ throughout.

Dual formulation:

$$\arg\min_{\boldsymbol{\alpha}} \left\{ \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{n} \alpha_i : \text{s.t.} \sum_{i=1}^{n} \alpha_i y_i = 0 \ \text{ for each } \boldsymbol{x}_i \right\}$$

SVM Classification

Nonlinear Kernels – Dual Formulation and Decision Function

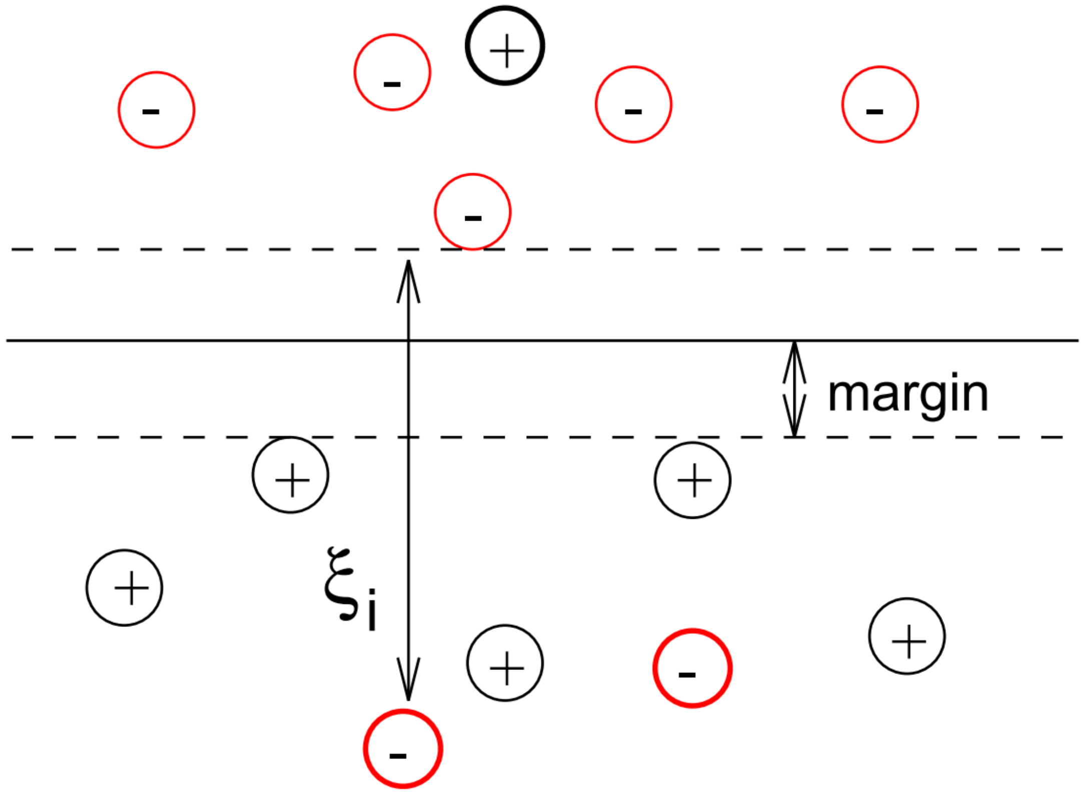
The decision function is defined by

$$T(\boldsymbol{x};\boldsymbol{\alpha}) = \sum_{k=1}^{L} \alpha_{i_k} y_{i_k} K(\boldsymbol{x}_{i_k}, \boldsymbol{x}) + \beta_0,$$

for each support vector $\boldsymbol{x}_{i_k}$, and is scaled as above.

Class assignment for $\boldsymbol{x}$ follows the same rule.

SVM Classification
Loss Functions and Soft Margins



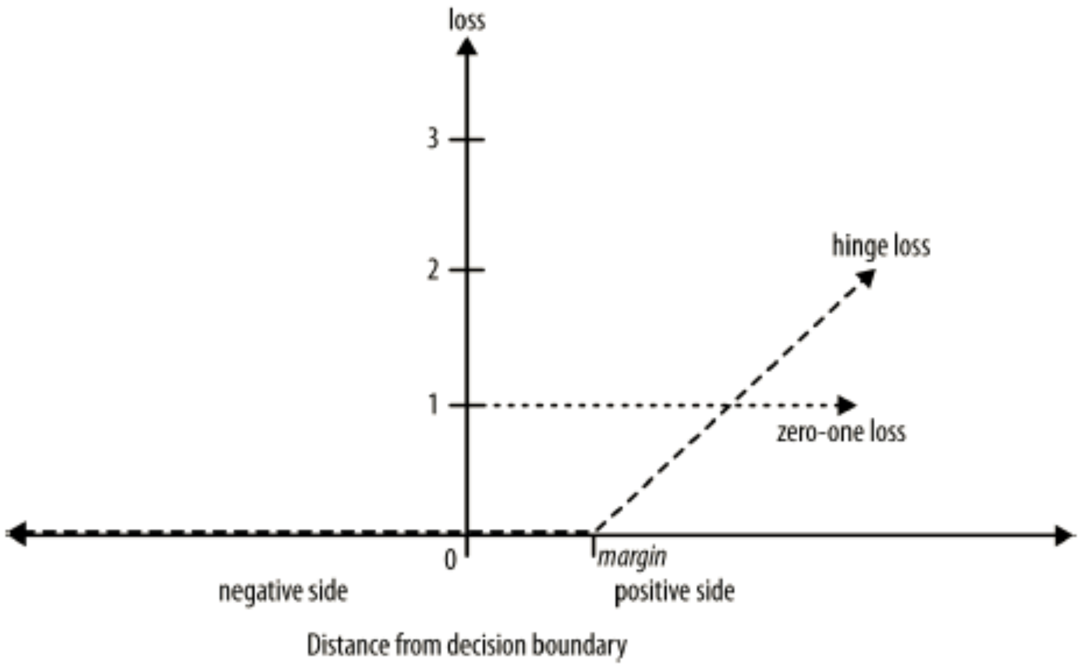
When dealing with **data which is not separable,** another approach is to introduce a cost ξ_i associated with making a classification mistake at $\boldsymbol{x}_i$.

[Author unknown]

The cost is specified by a **loss function**.

In general, training a model is equivalent to minimizing a loss function; **machine learning is really optimization in disguise**.

SVM Classification
Loss Functions



The **Heaviside** loss function penalizes any and all instances appearing on the wrong side of the separating curve at the same rate.

The **hinge** loss function only penalizes instances appearing outside the other margin, and it increases with distance.

[Adapted from Foster & Provost's *Data Science for Business*]

SVM Classification
Soft Margins – Formulation

Given an upper limit on the misclassification cost C, we solve

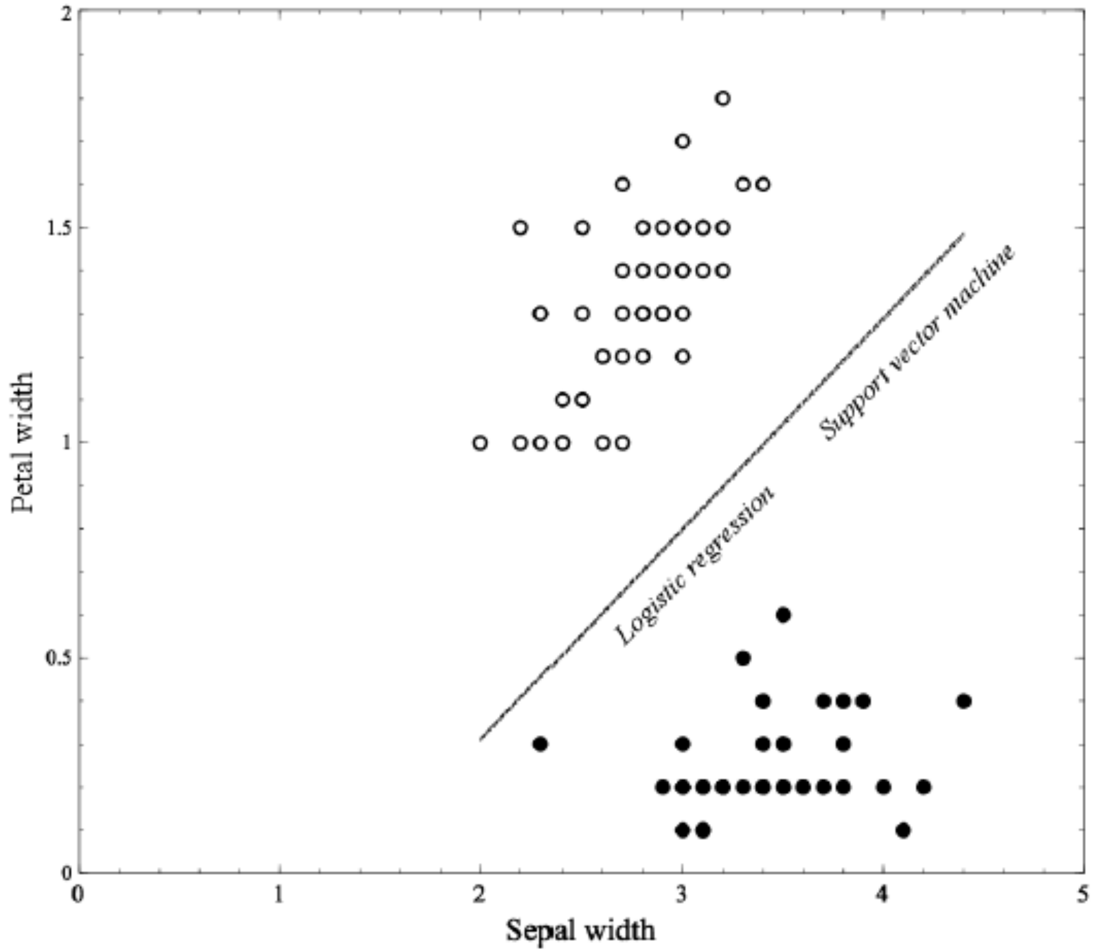
$$\arg\min_{\boldsymbol{\alpha}} \left\{ \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{n} \alpha_i : \text{s.t.} \sum_{i=1}^{n} \alpha_i y_i = 0 \text{ and } 0 \le \alpha_i \le C \right\}$$

The decision function is defined by

$$T(\boldsymbol{x}; \boldsymbol{\alpha}) = \sum_{k=1}^{L} \alpha_{i_k} y_{i_k} K(\boldsymbol{x}_{i_k}, \boldsymbol{x}) + \beta_0,$$

is scaled as above and satisfies $\left|T\left(\boldsymbol{x}_{i_k}; \boldsymbol{\alpha}\right)\right| \ge 1 - \boldsymbol{\xi}_{i_k}$ for each support vector $\boldsymbol{x}_{i_k}$.

Further Considerations
Pros



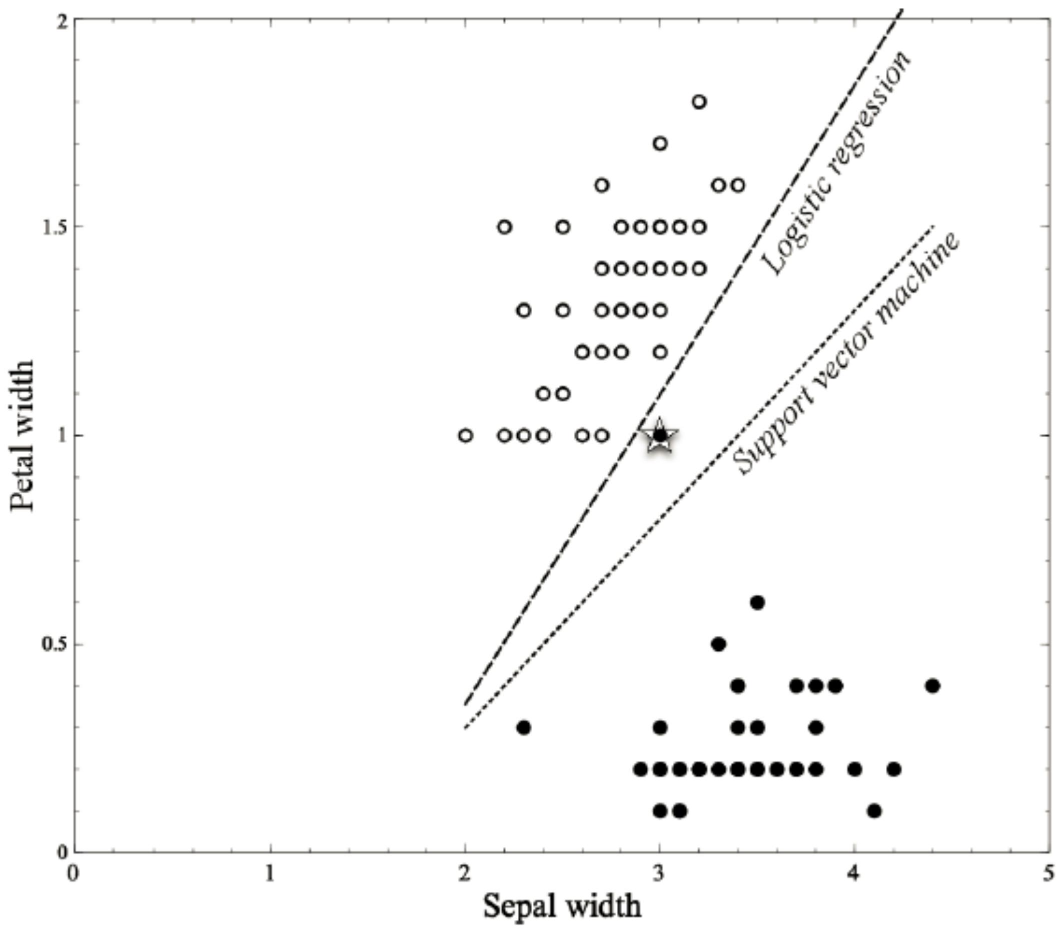
SVMs are well-protected against the risk of **overfitting** (due to the small number of support vectors).

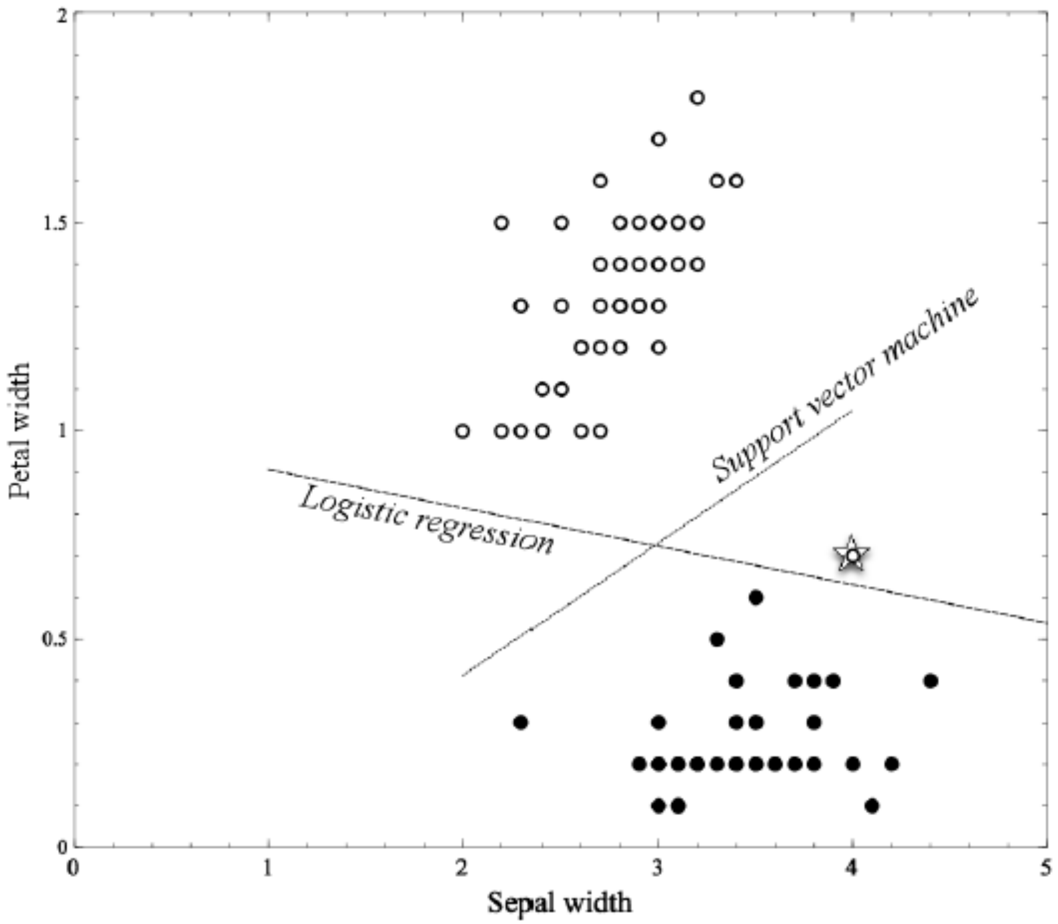
Consider the following subset of the iris dataset. The decision boundaries for logistic regression and SVM coincide.

Watch what happens when we add observations which leave the clean separation in question.

[Adapted from Foster & Provost's *Data Science for Business*]

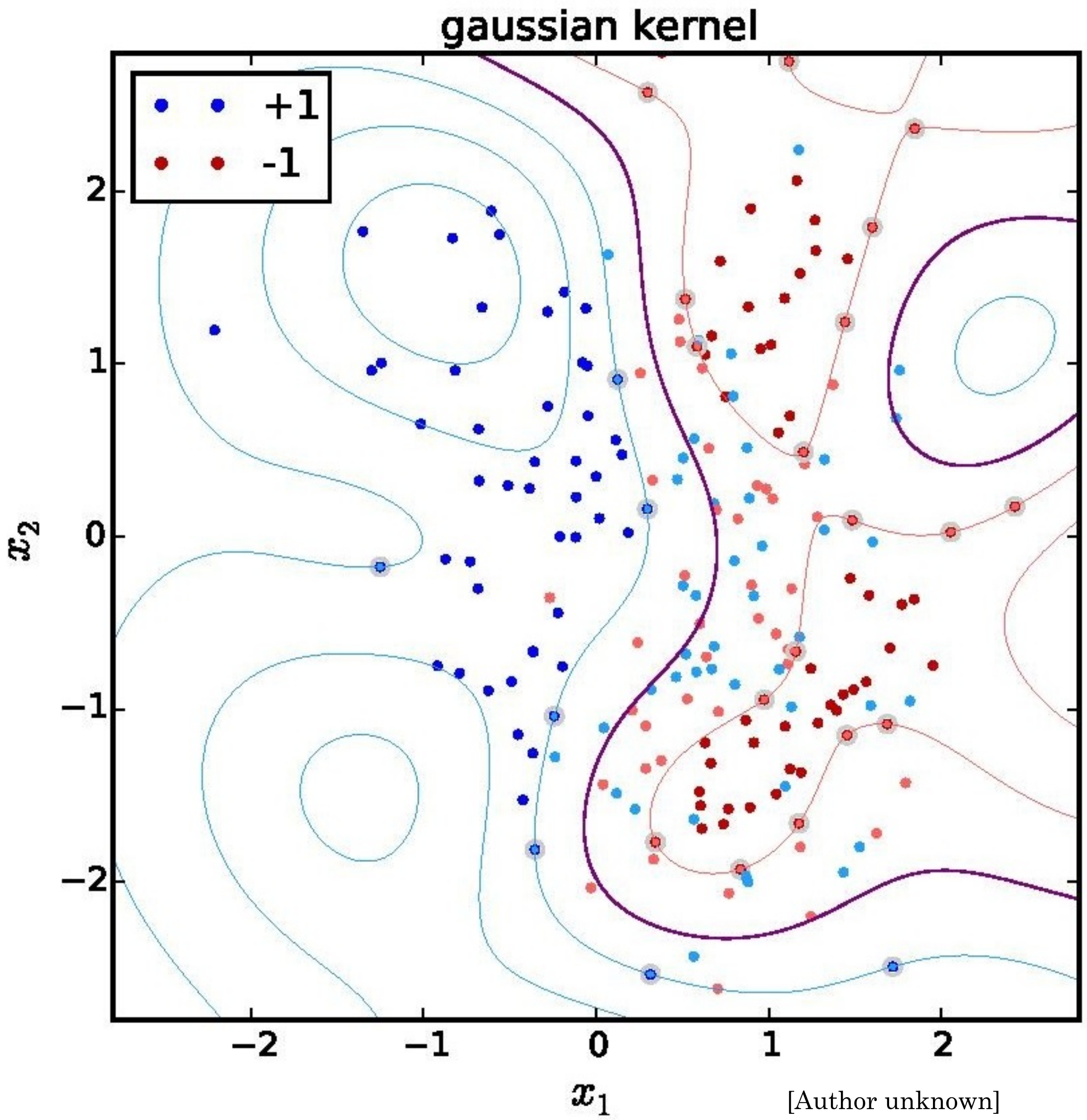
Further Considerations
Pros and Cons





The SVM decision boundary is smarter about its reactions to the addition of outlying observations.

Further Considerations
Cons



[Author unknown]

SVMs have uncalibrated class membership probabilities

Only directly applicable to two classes

It's hard to imagine how we would describe the decision procedure for **this** dataset.

Remember the *No-Free Lunch Theorem.*

Further Considerations
Implementation and Extensions

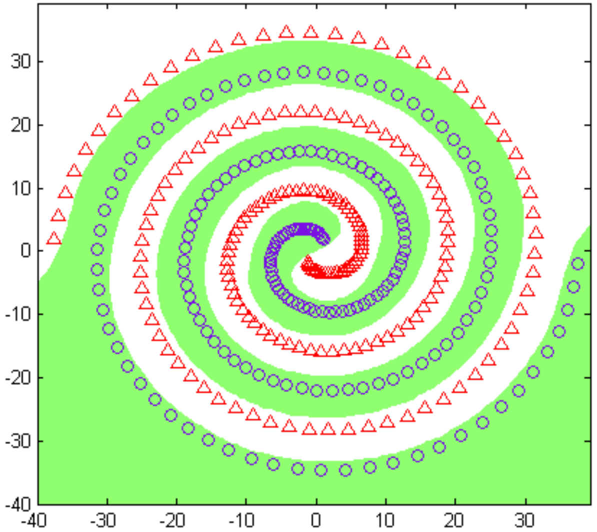
Using kernels saves computational storage space

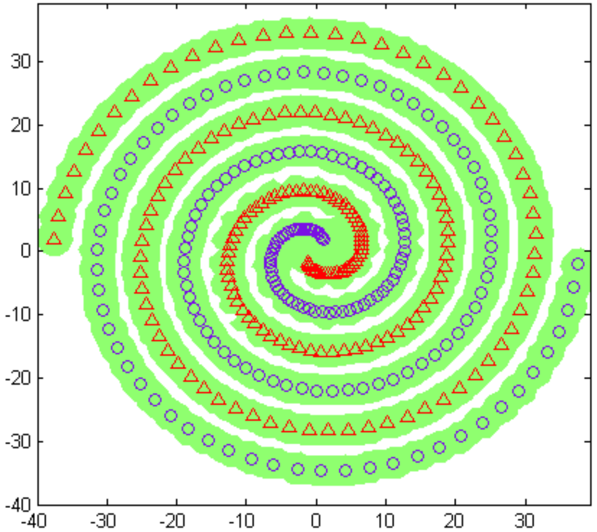
Specialized QP solvers can be used

Kernel SVMs are available in most machine learning toolkits

- LIBSVM
- MATLAB
- SAS
- kernlab
- etc.

Least-Square SVMs (spirals), Regression (examples)





SVM Regression
Formulation

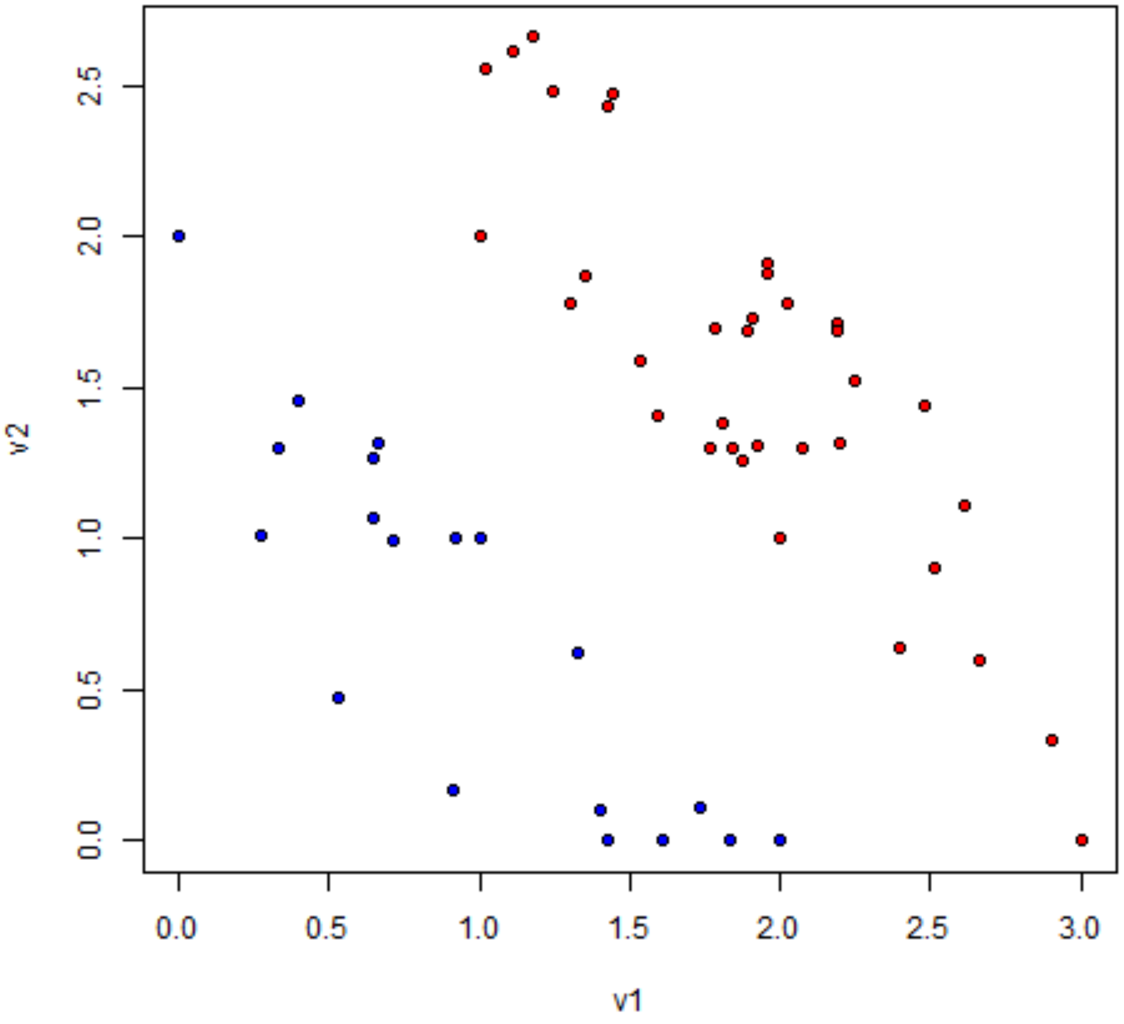
Given an upper limit C on the weights and an error ε, we solve

$$\arg\min_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \left\{ \begin{array}{c} \frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)K(\boldsymbol{x}_i, \boldsymbol{x}_j) + \varepsilon\sum_{i=1}^{n}(\alpha_i + \alpha_i^*) - \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)y_i \\ \text{subject to} \sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0 \text{ and } 0 \leq \alpha_i, \alpha_i^* \leq C \end{array} \right\}$$

The regression function is defined by

$$T(\boldsymbol{x}; \boldsymbol{\alpha}, \boldsymbol{\alpha}^*) = \sum_{k=1}^{L}(\alpha_i - \alpha_i^*)K(\boldsymbol{x}_{i_k}, \boldsymbol{x}) + \beta_0$$

Examples
Artificial Dataset

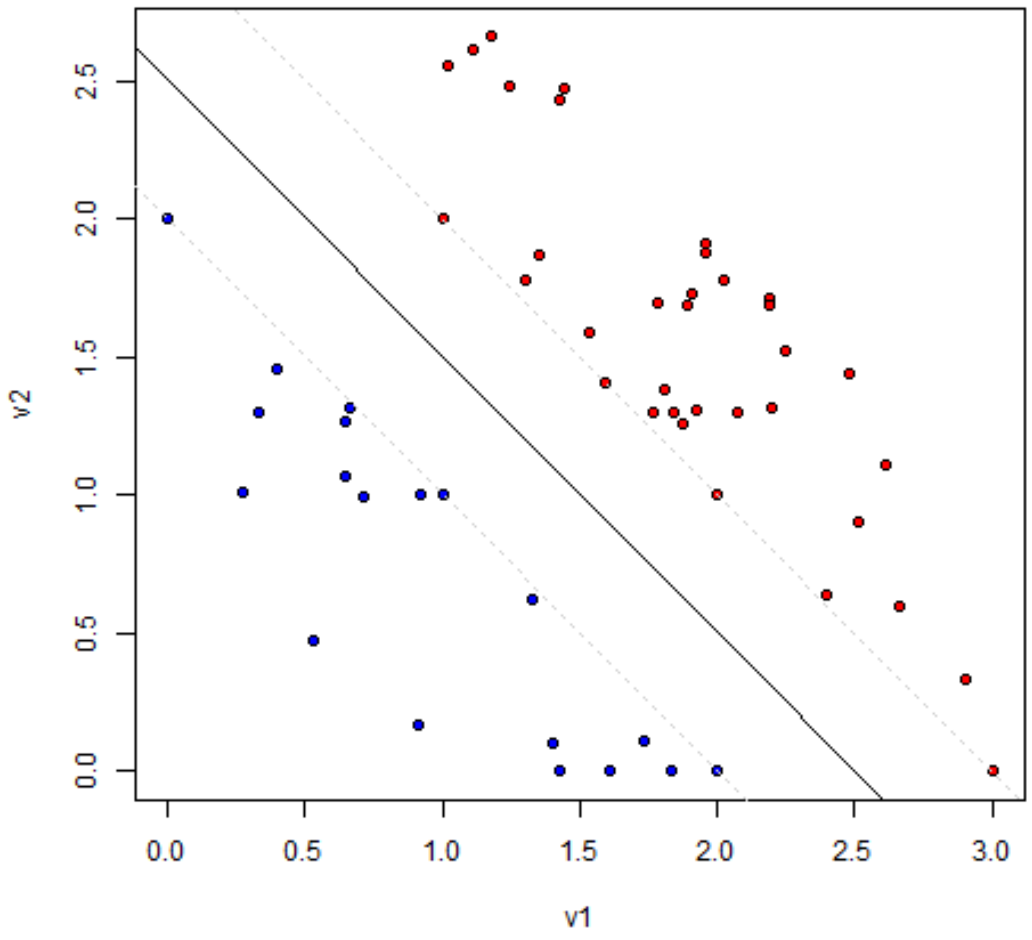


Artificial dataset, with 2 classes

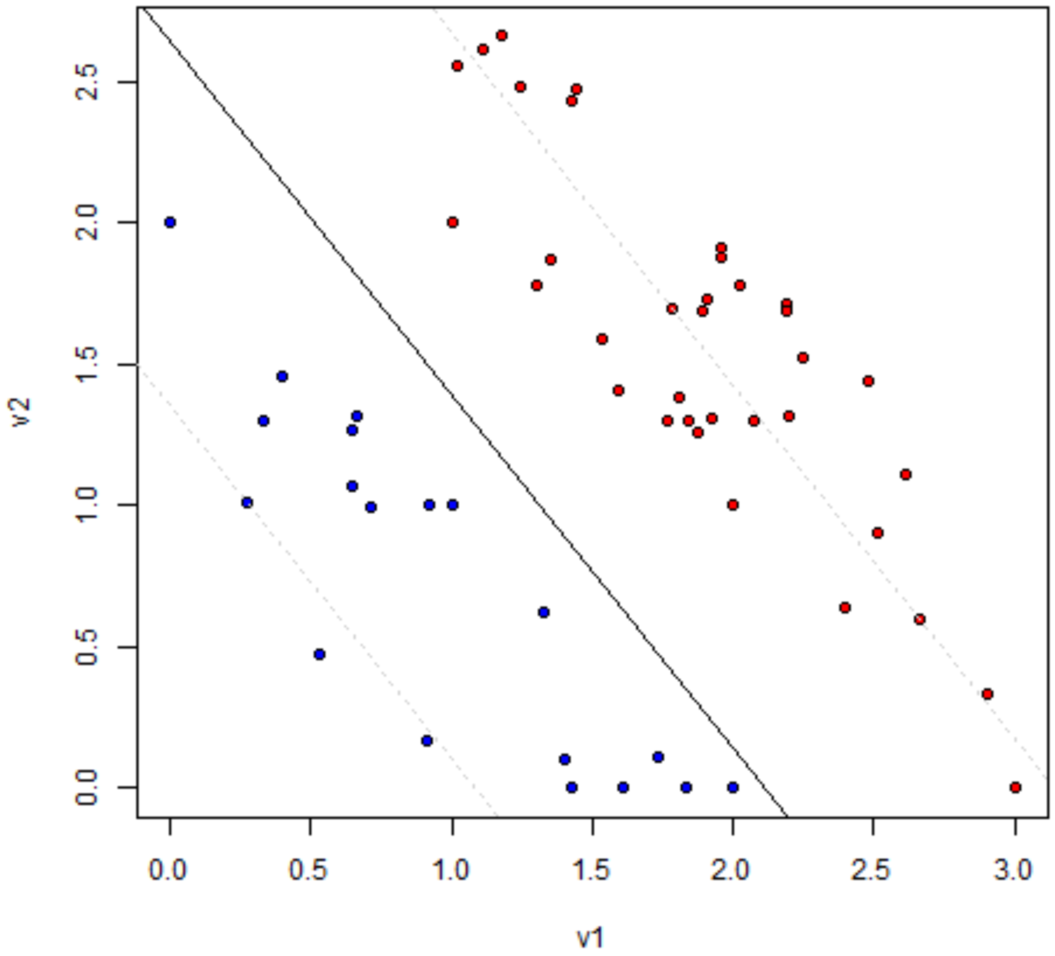
Linearly separable

Examples
Artificial Dataset

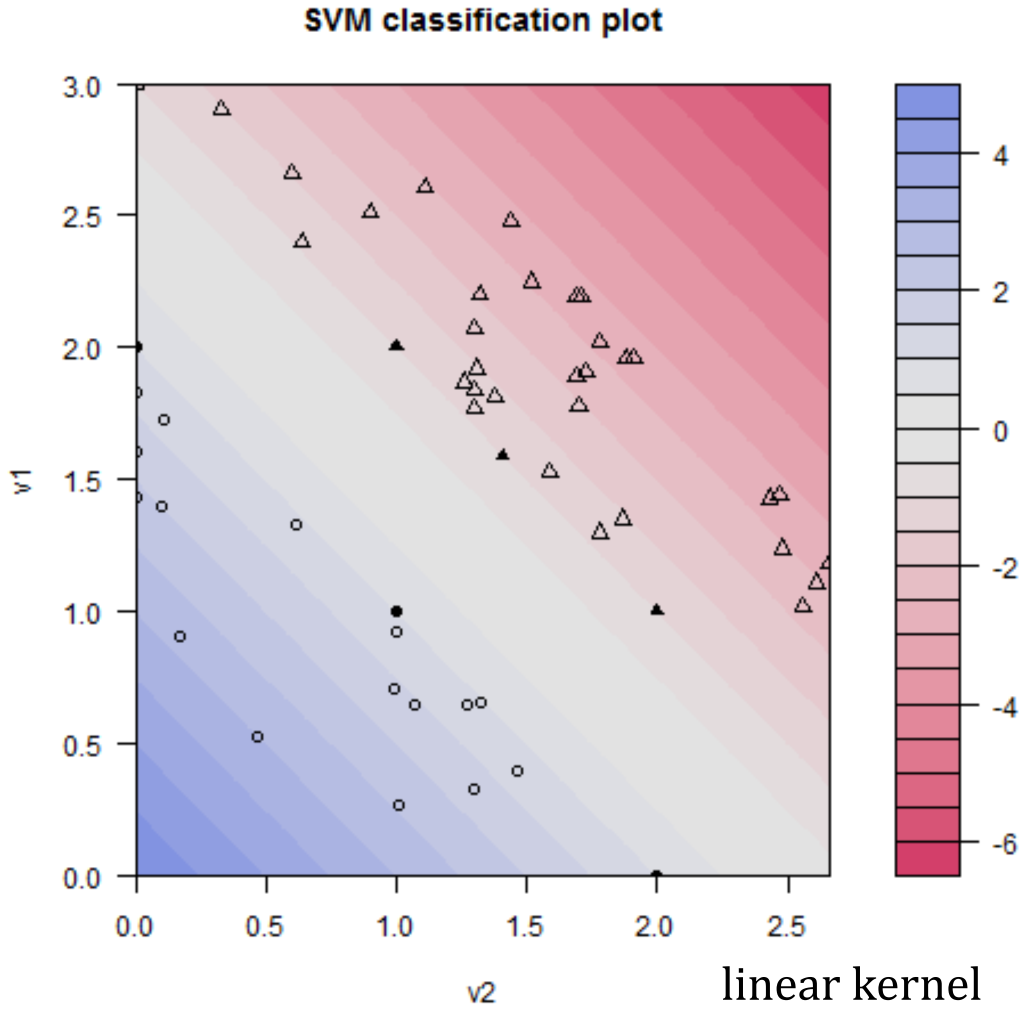
hard margin



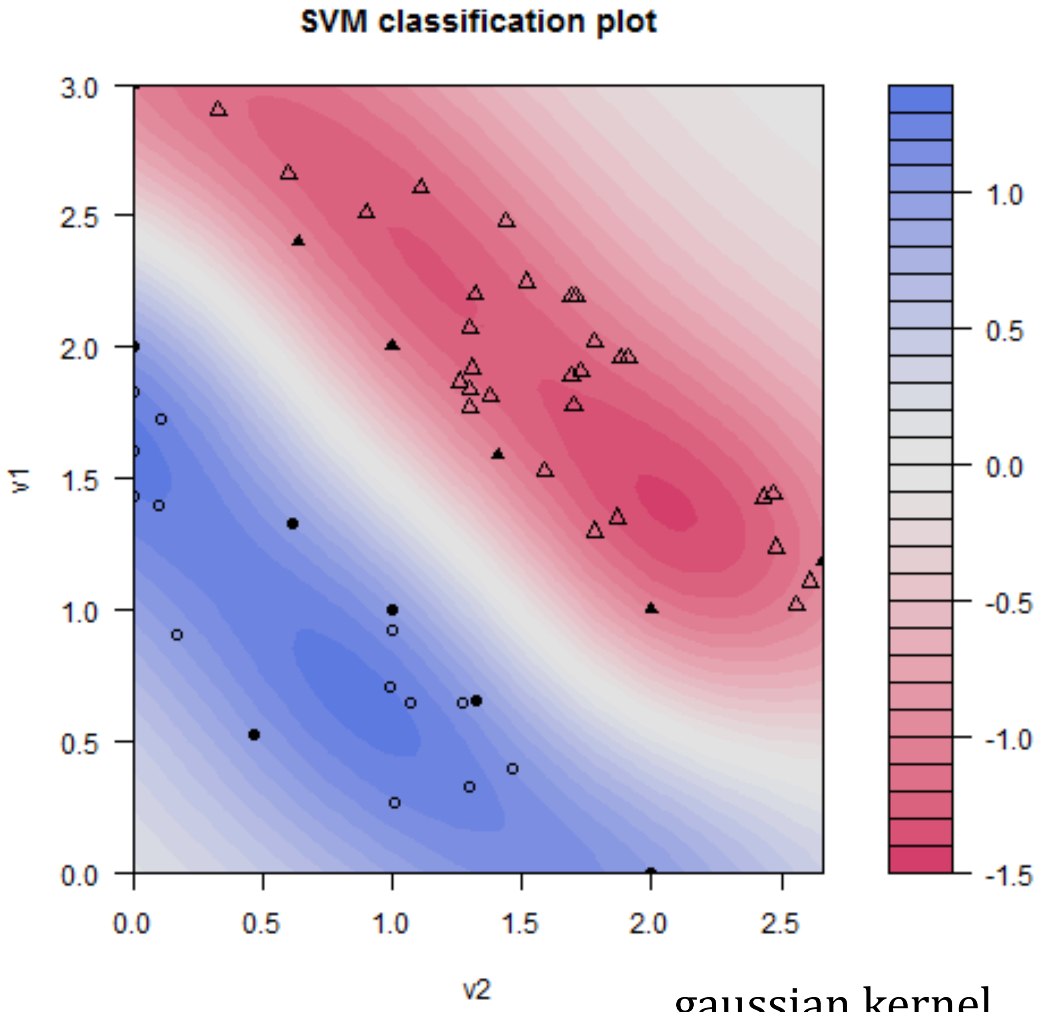
soft margin



Examples
Artificial Dataset

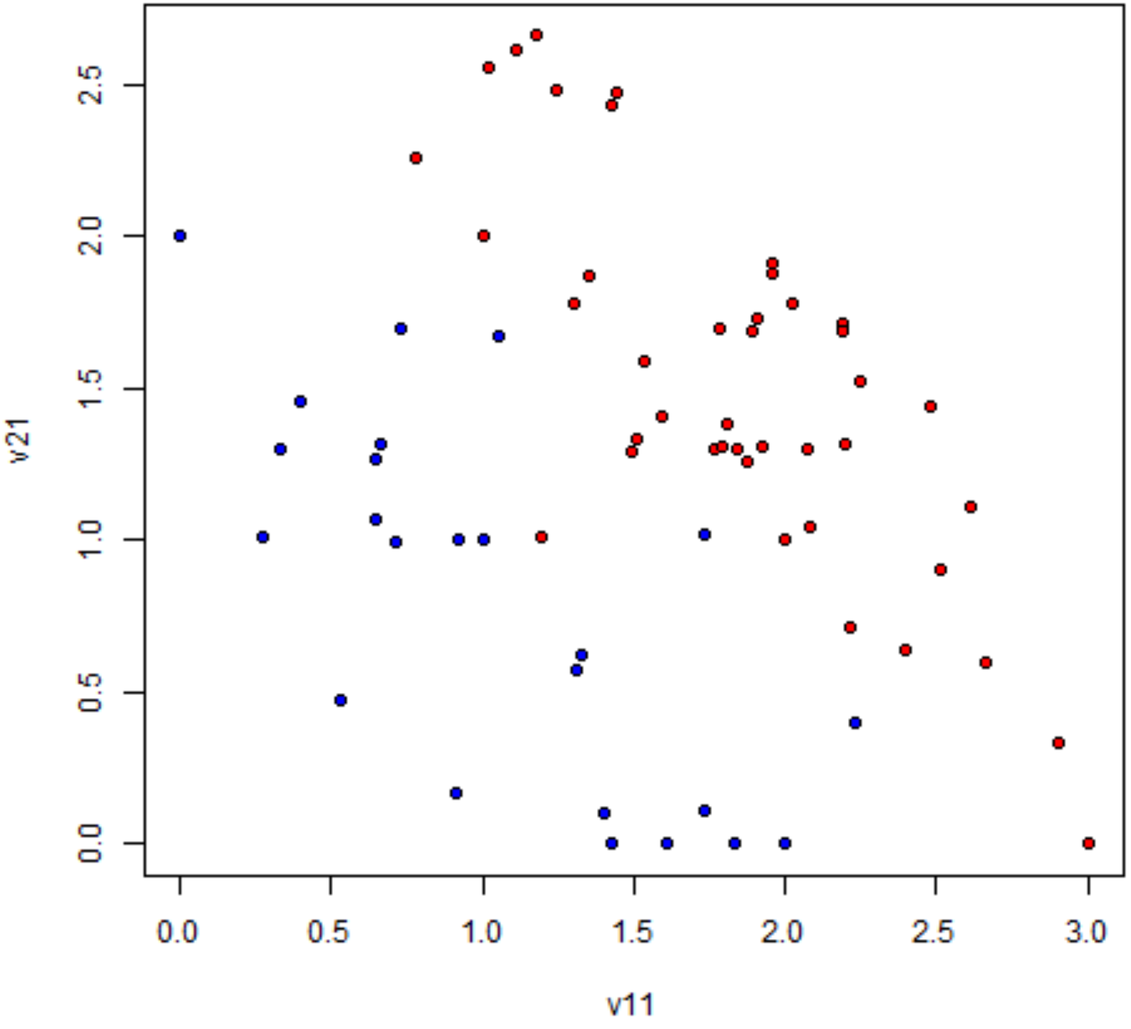


linear kernel



gaussian kernel

Examples
Artificial Dataset

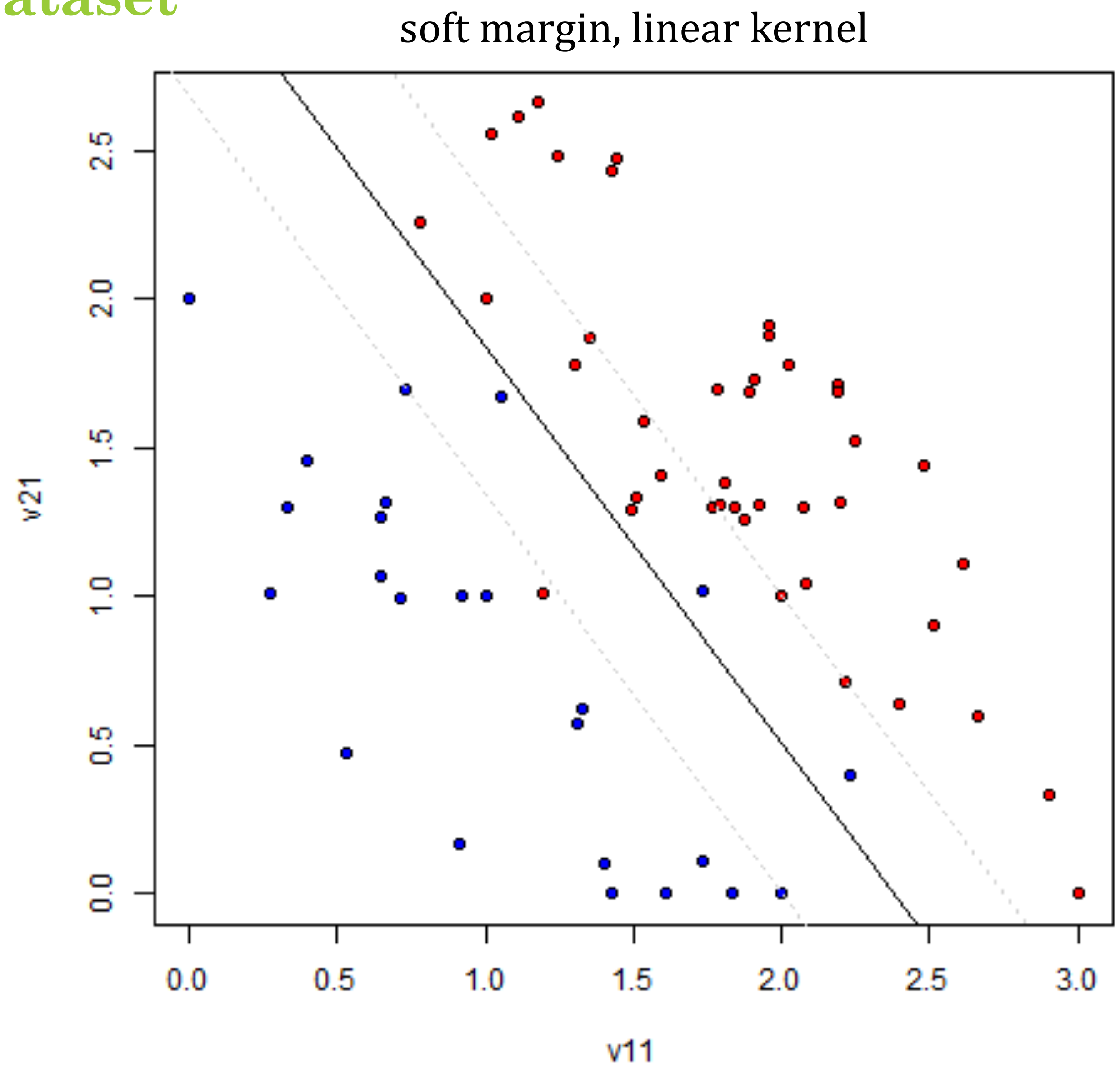


Artificial dataset, with 2 classes

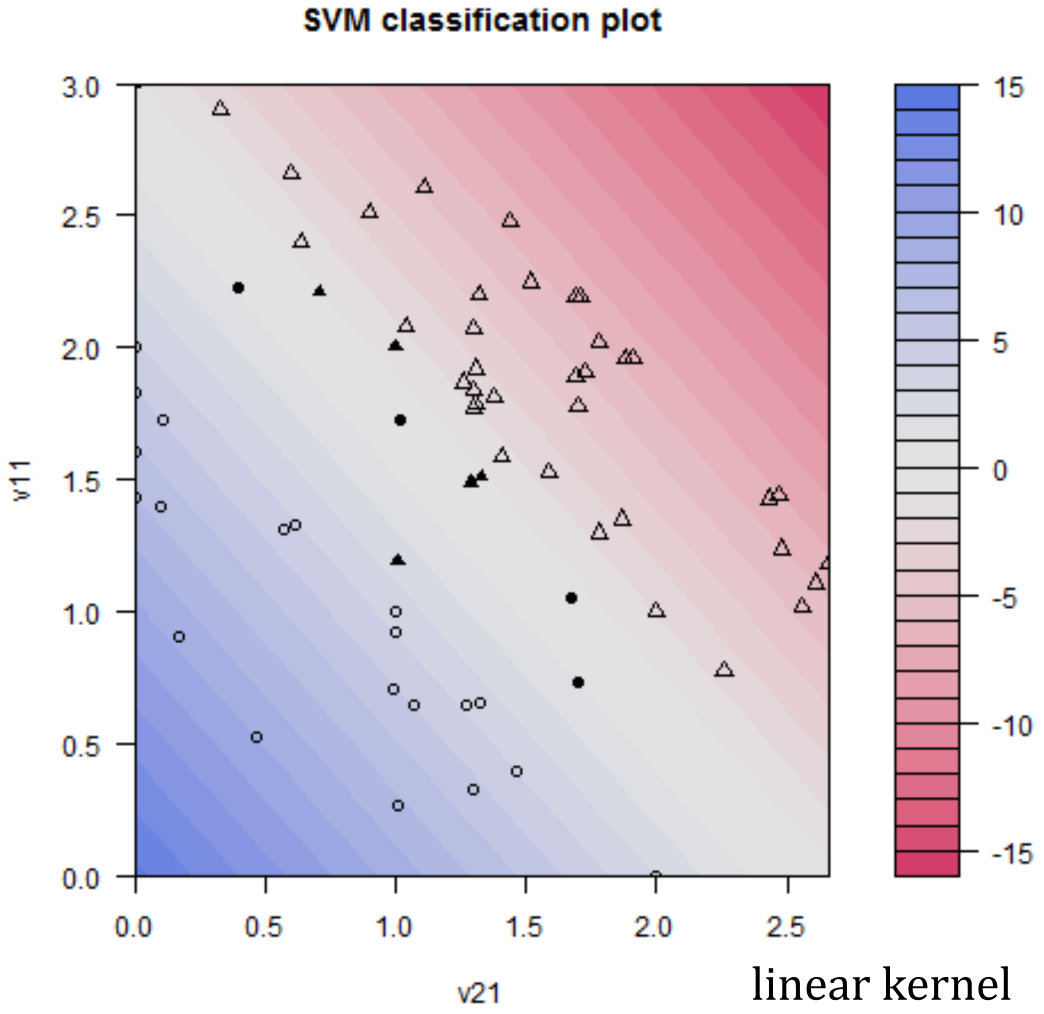
Not linearly separable

Examples

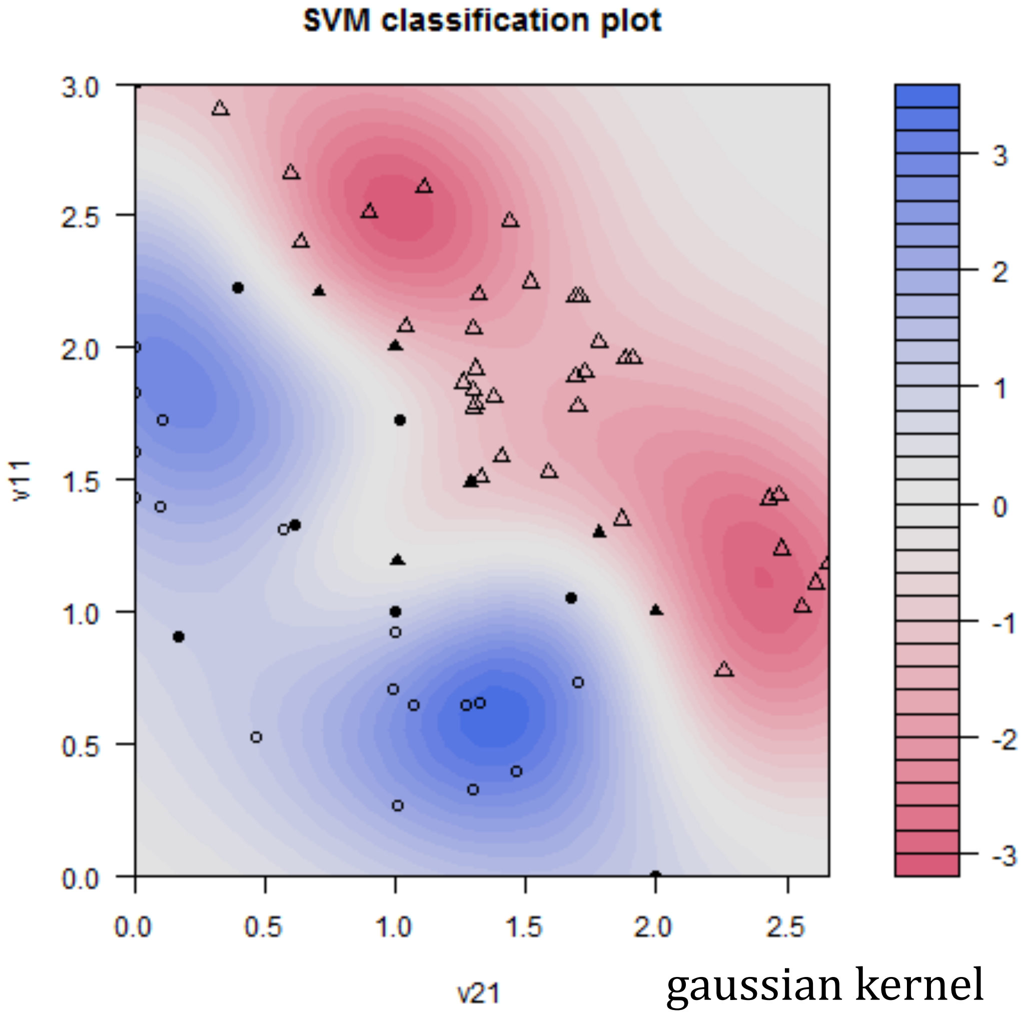
Artificial Dataset



Examples
Artificial Dataset



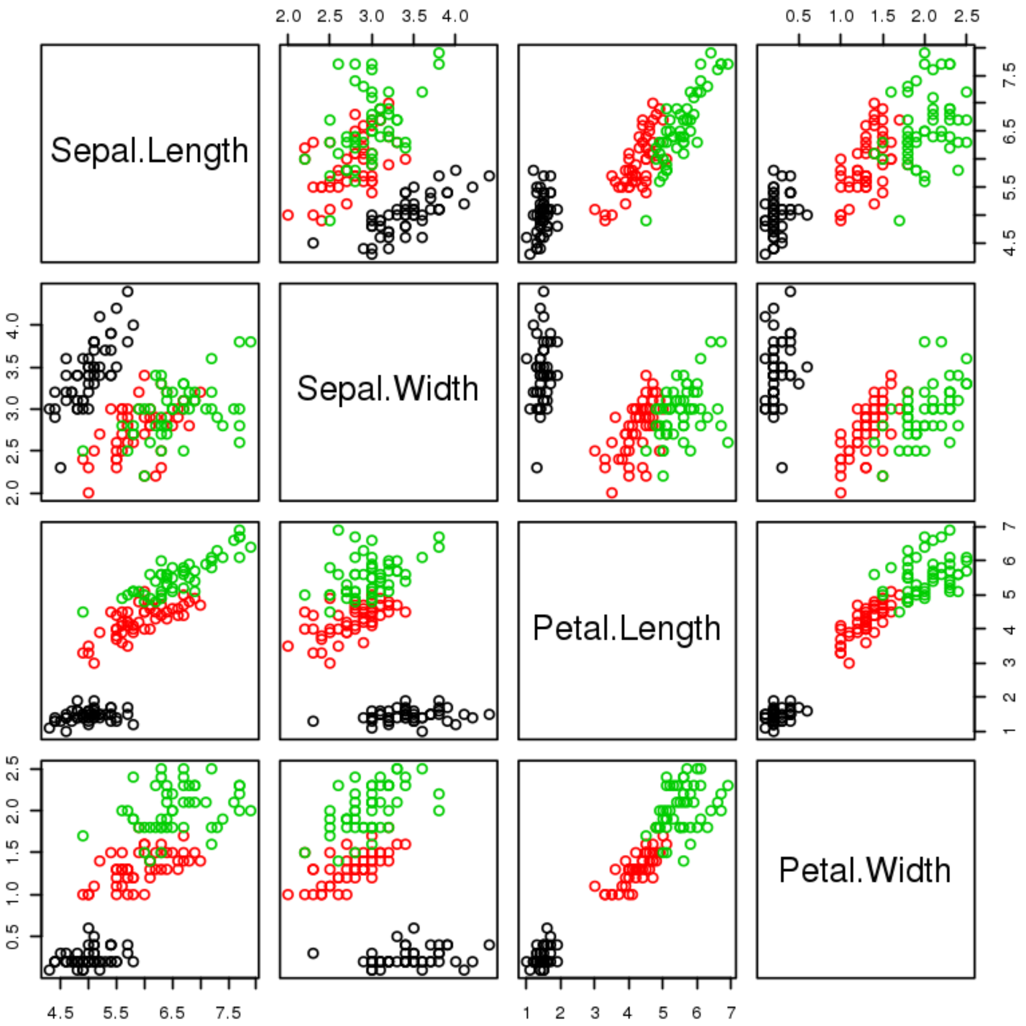
linear kernel

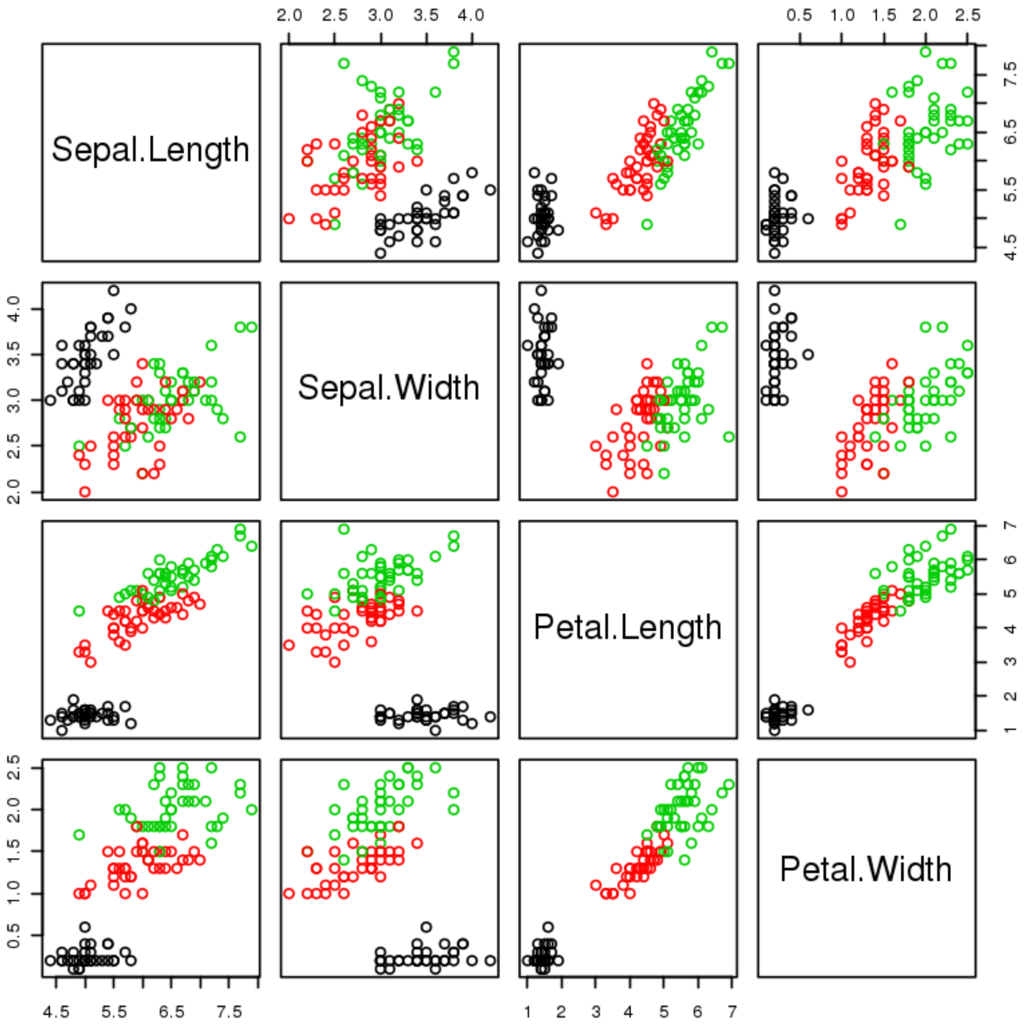


gaussian kernel

Examples

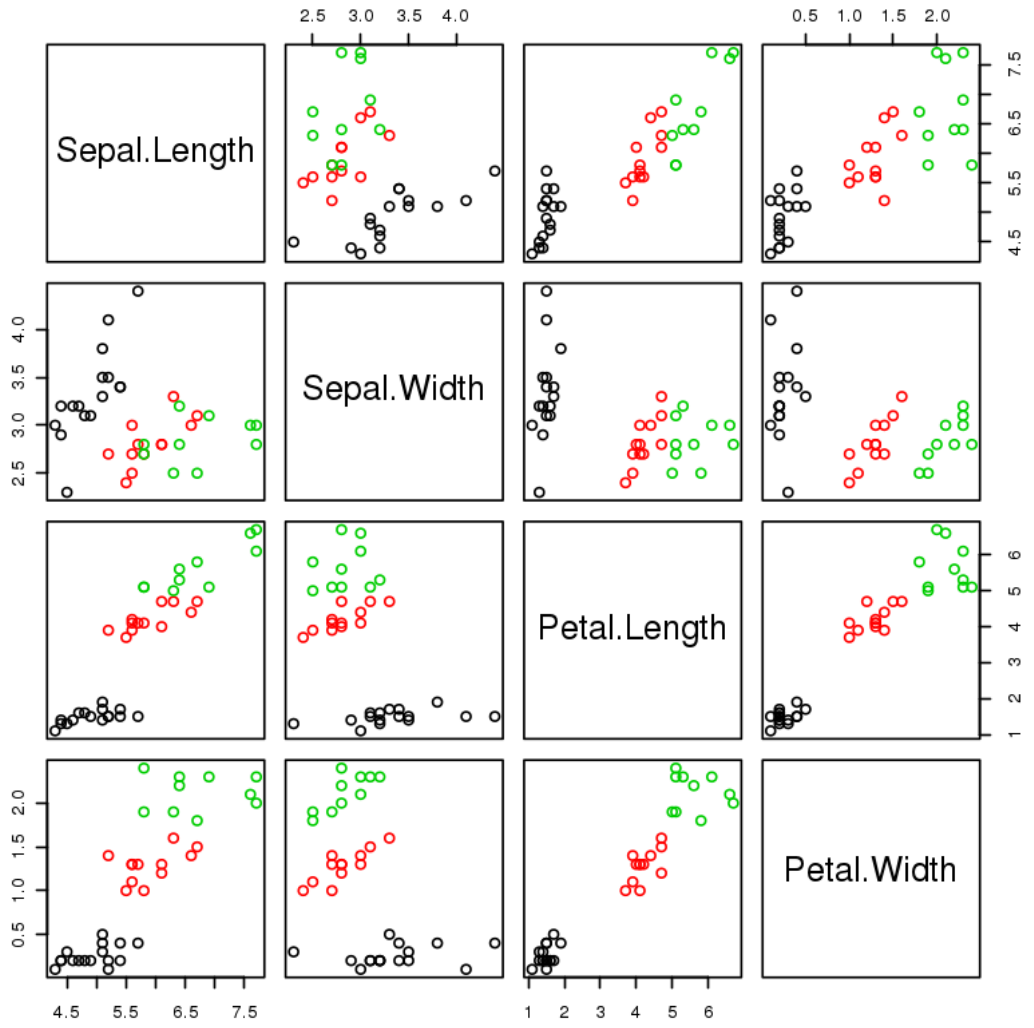
Iris Dataset



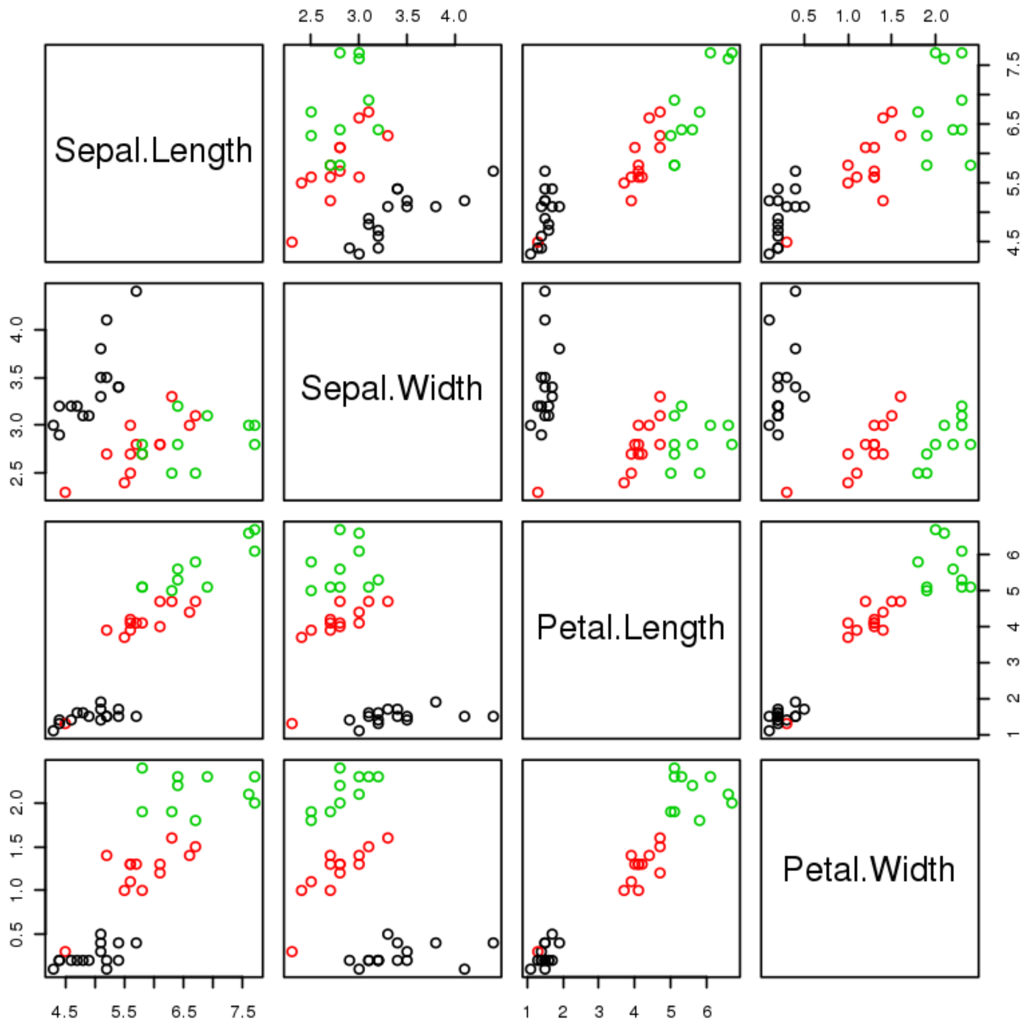


training set

Examples
Iris Dataset

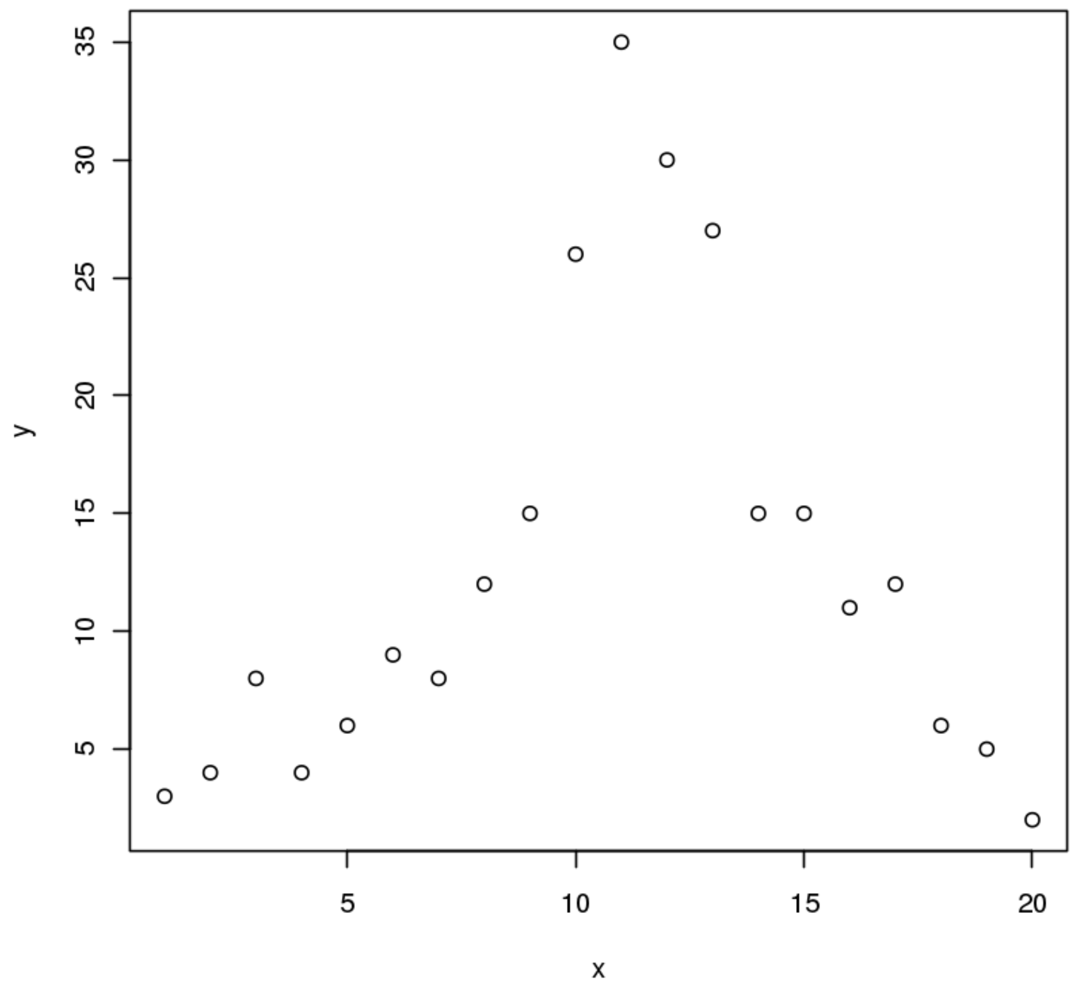


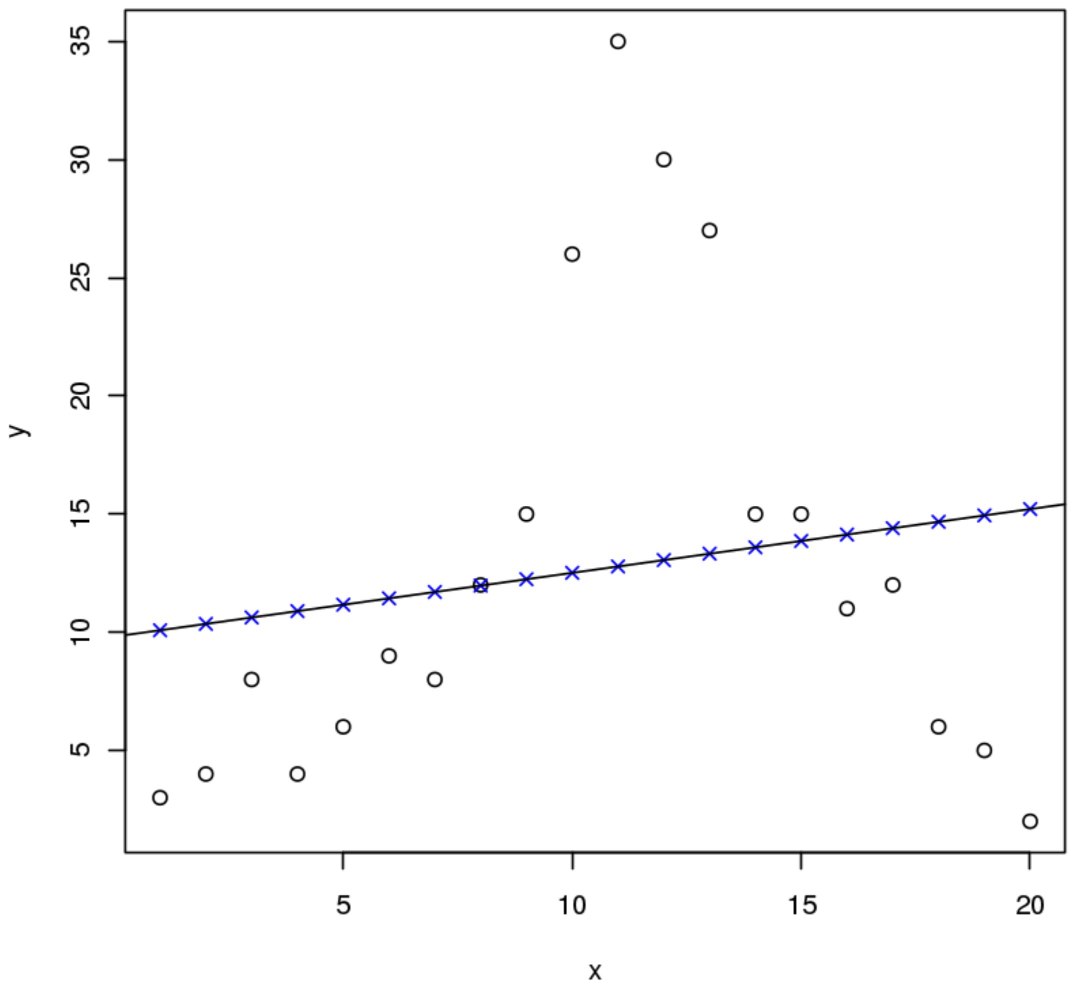
testing set



predictions

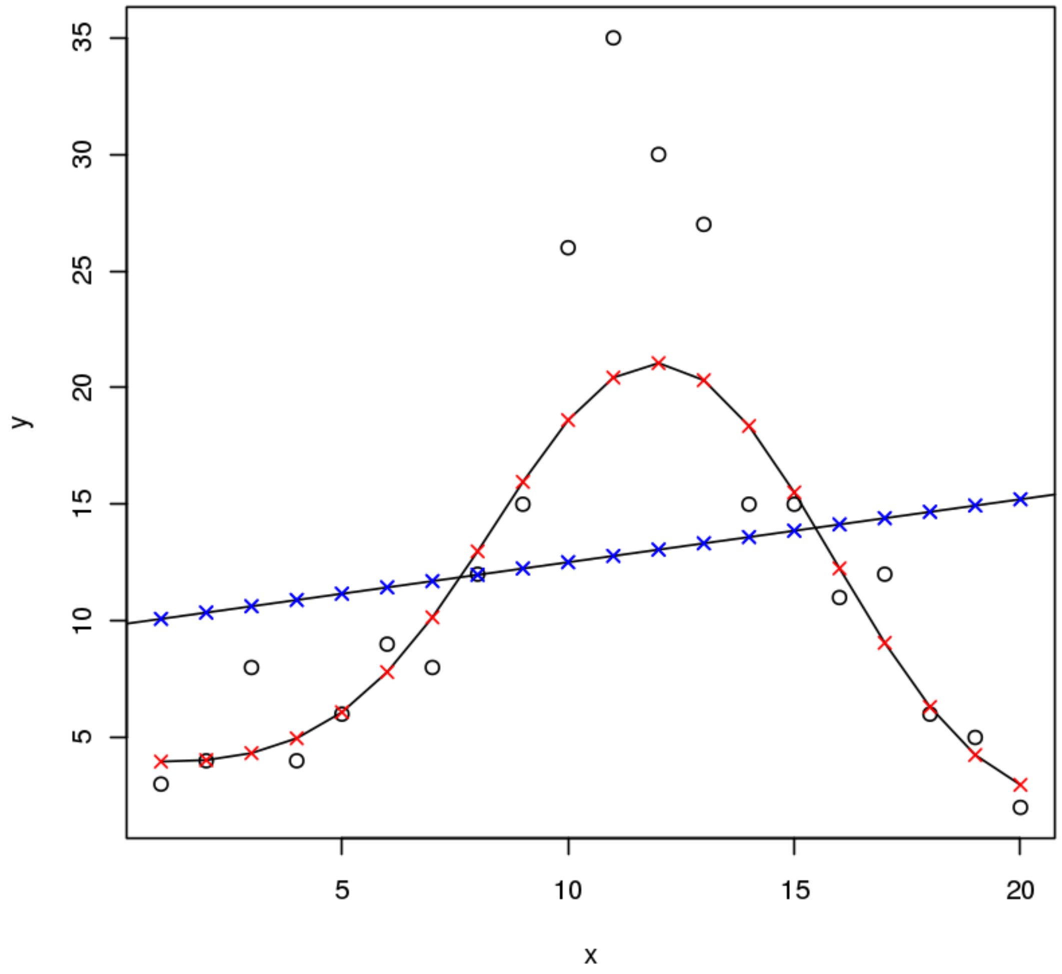
Examples
SVM Regression

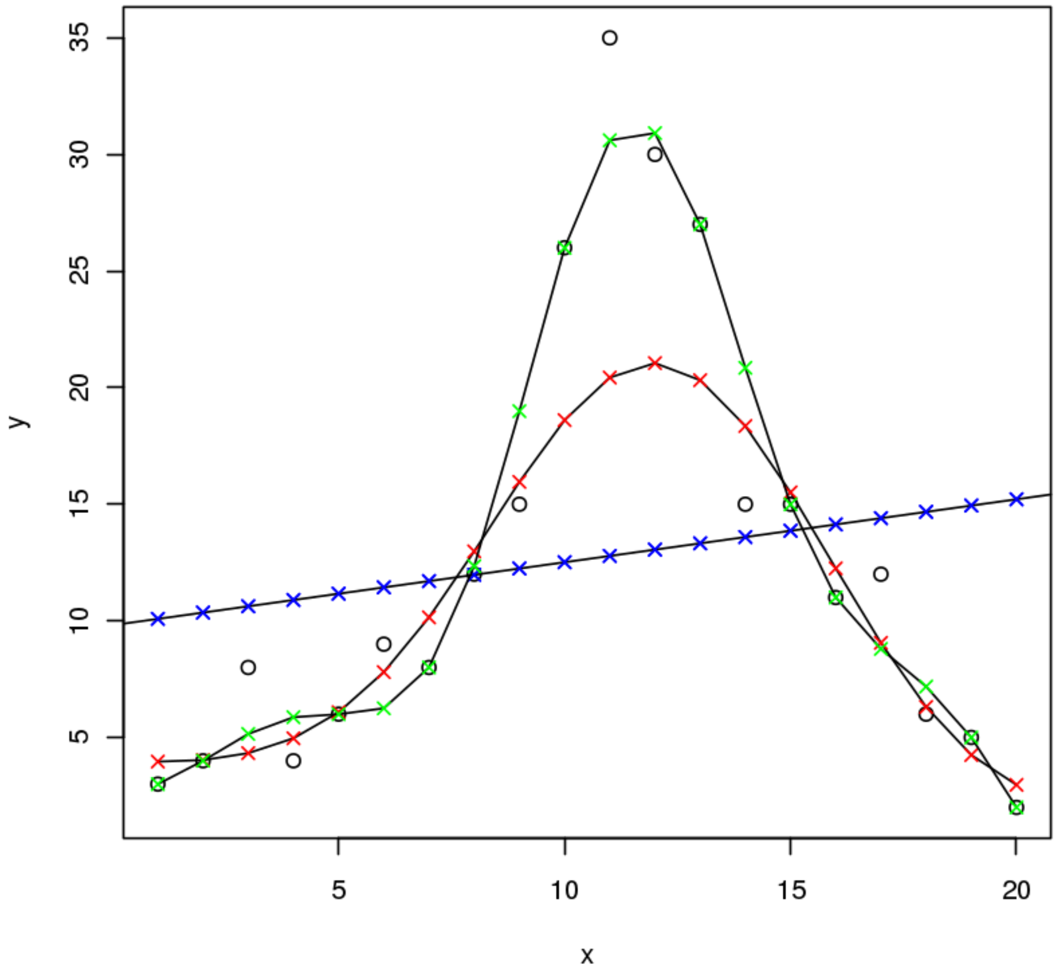




Examples

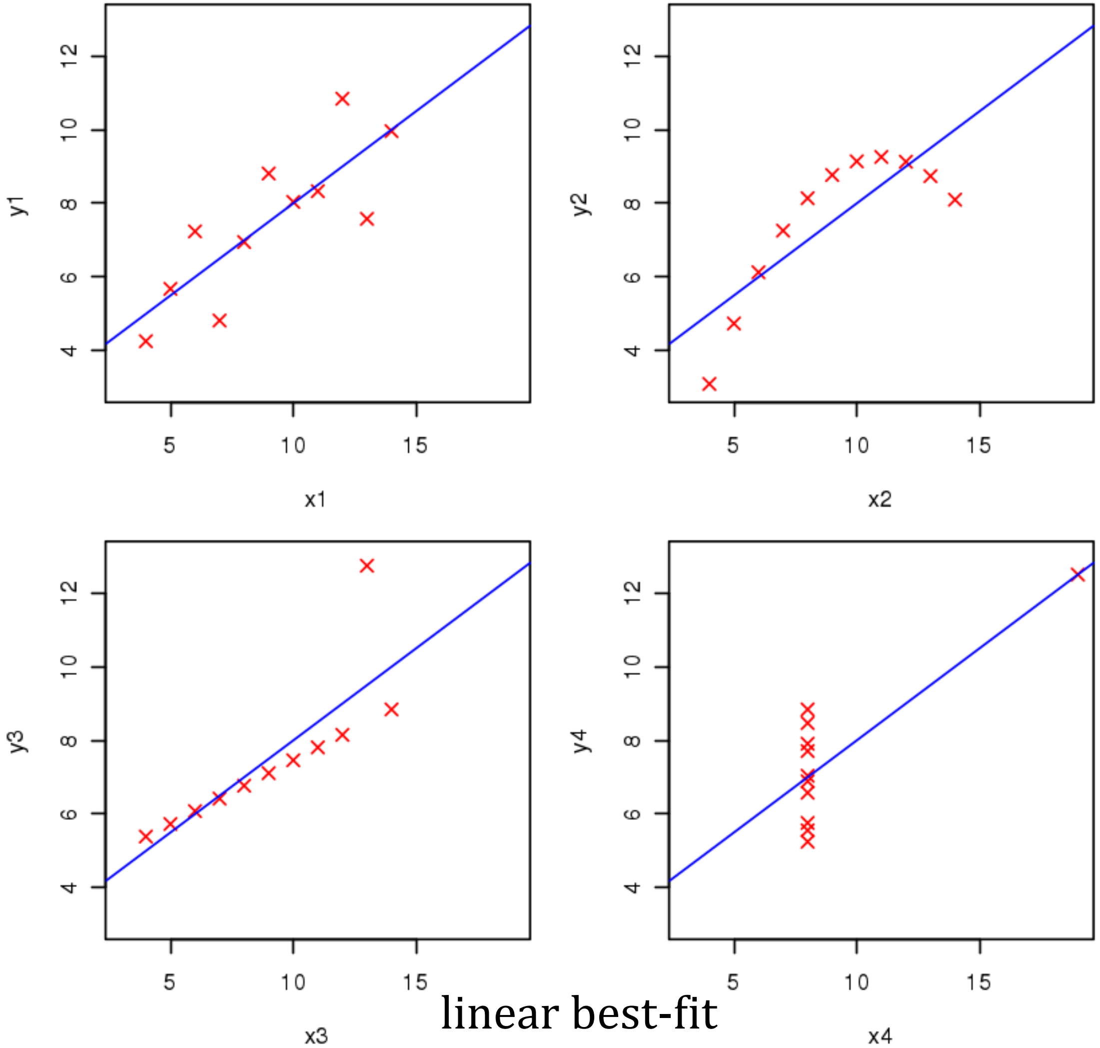
SVM Regression



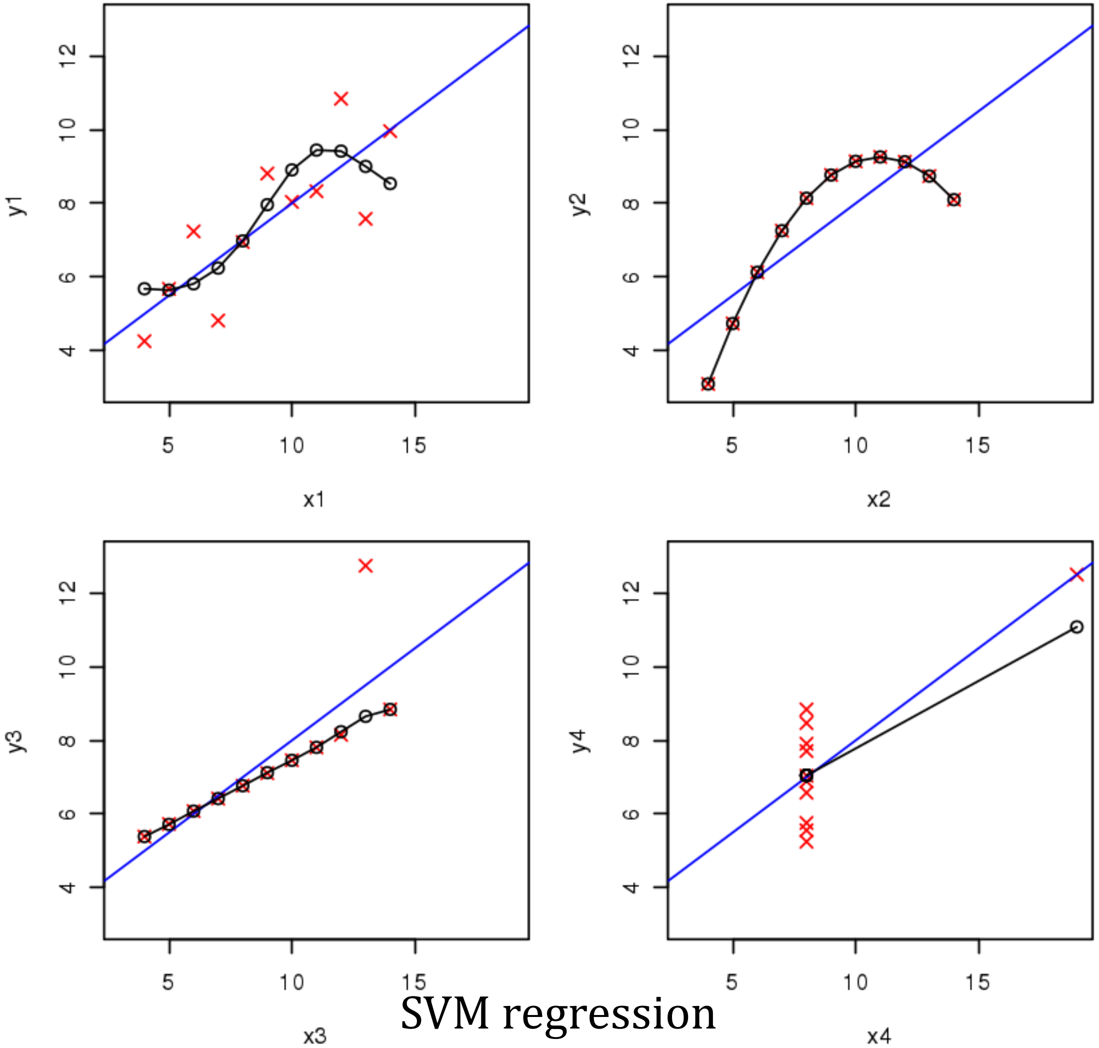


Examples

Anscombe's Quartet



linear best-fit



SVM regression