The Kernel Trick and its Applications

Benjamin Roycraft

University of California

btroycraft@ucdavis.edu

June 1, 2018

Overview

Introduction

Kernel methods are a general class of procedures used to extend simple linear techniques into the non-linear setting.

They have seen use in a wide range of statistical fields:

- Classification
- Handwriting recognition
- Bioinformatics
- Image recognition

Introduction

Depending on the implimentation, kernel methods offer sufficient flexibility to adapt to a wide range of data, while utilizing relatively simple linear methods for a core foundation.

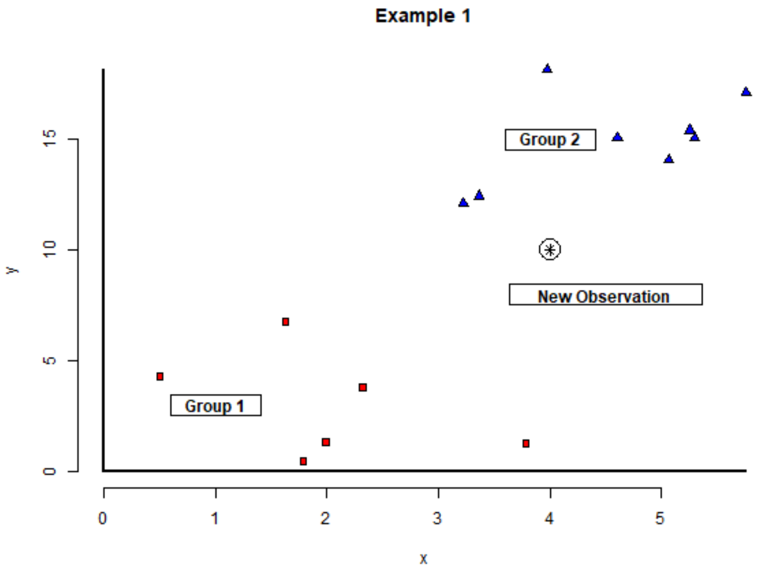
Kernel methods are computationally efficient, and perform competitively in classification accuracy and other metrics.

Linear Separability

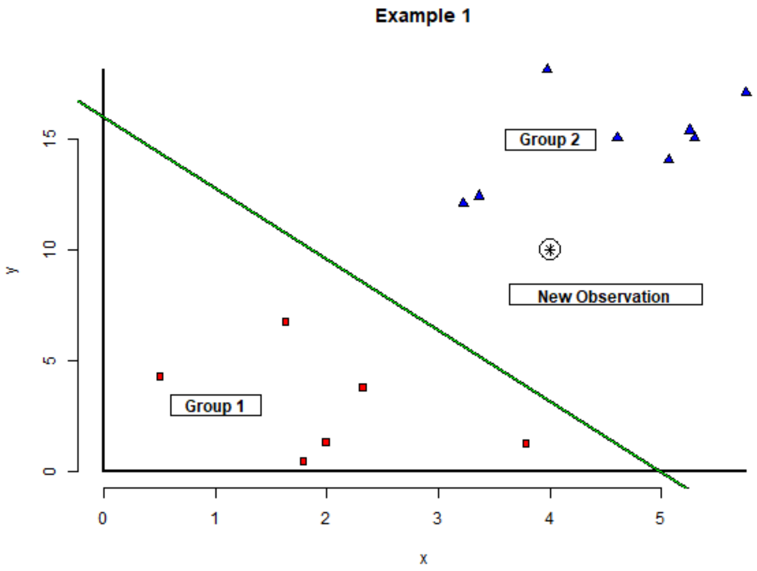
Goal:

- Use existing data to classify new observation.
- Fast, simple method

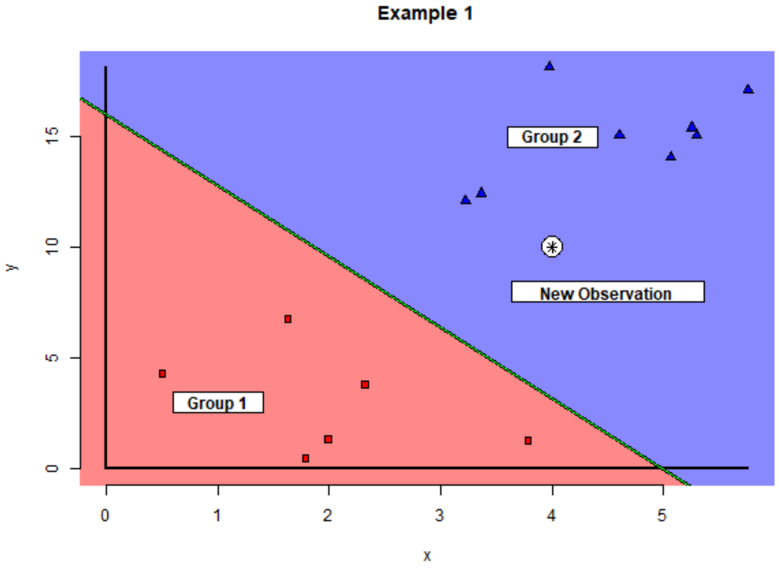
Linear Separability



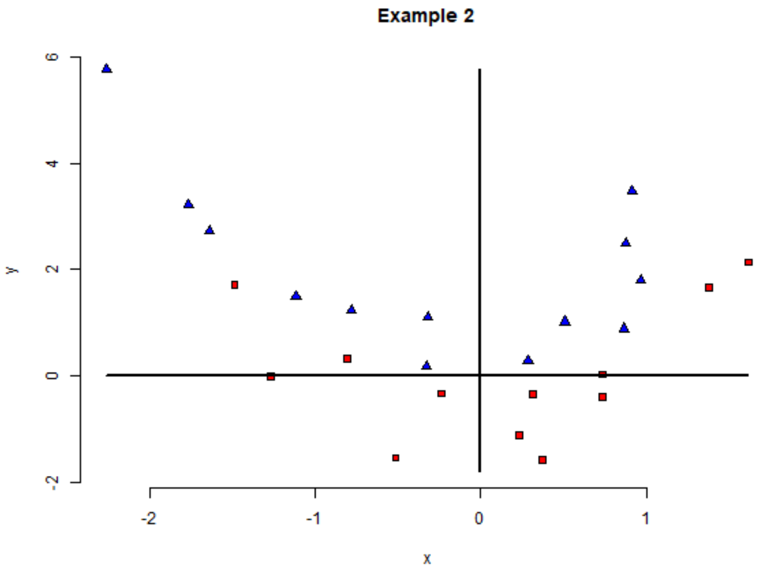
Linear Separability



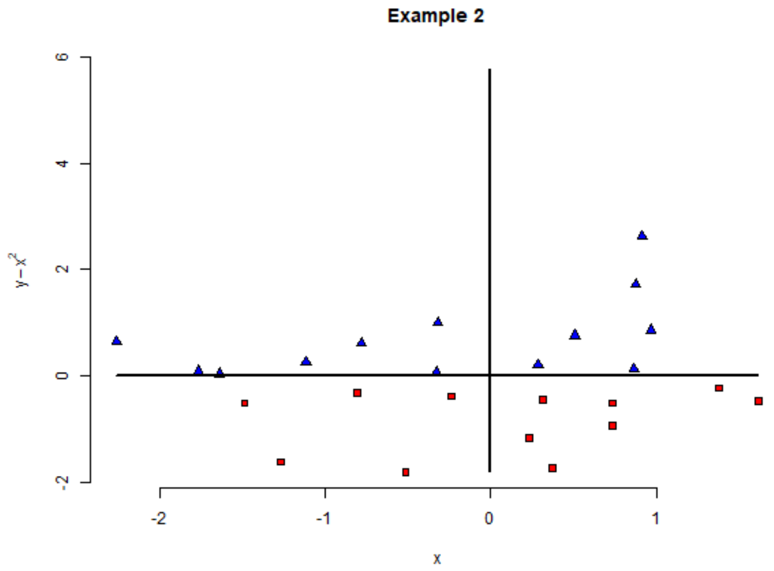
Linear Separability



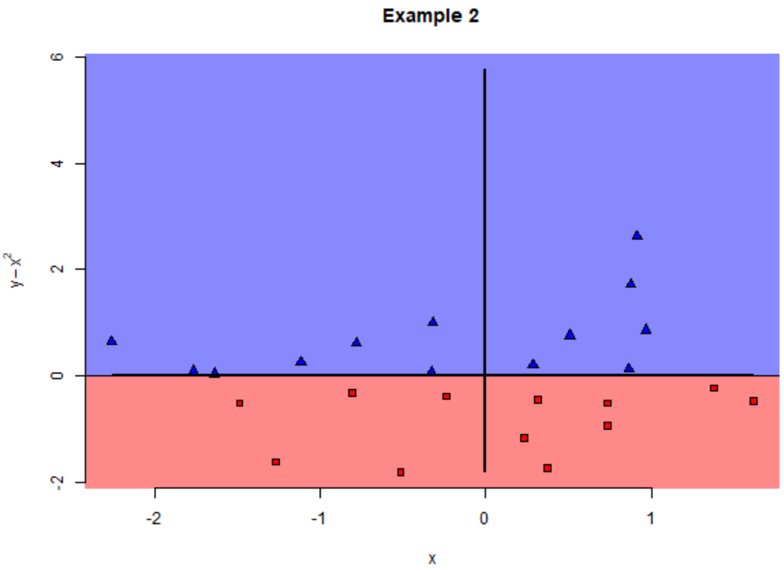
Linear Separability



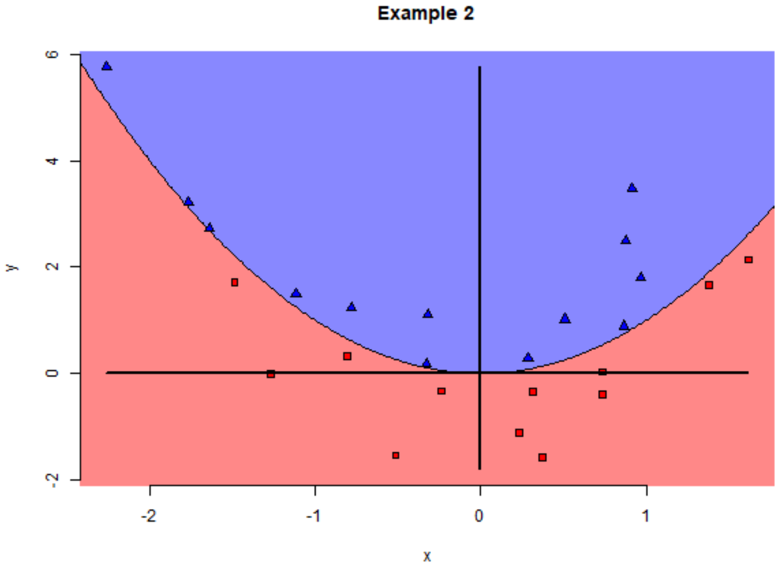
Linear Separability



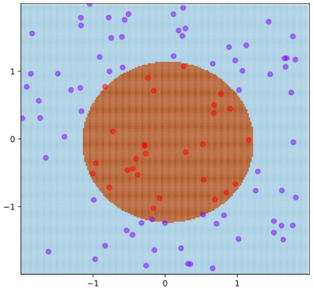
Linear Separability

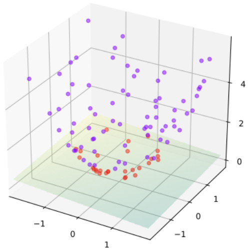


Linear Separability



Linear Separability





Example 3

Feature Vectors

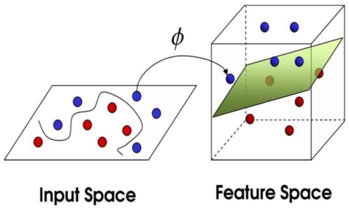
General ideas:

- Applying transformations to data turns a simple linear method into a more complex non-linear method. A non-linear transformation (quadratic, cubic, exponential, etc.) can better match the structure present in the data.
- Mapping into higher dimensions than the original generally yields an increase in separability, making it easier to distinguish data from multiple classes.

Feature Vectors

A *feature vector* is any form of new data gained from a transformation of the original.

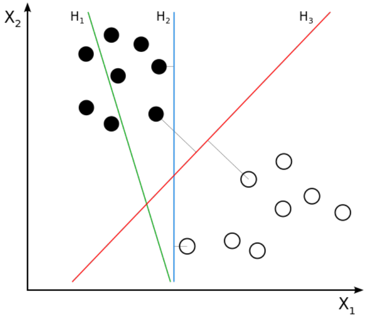
$$x \rightarrow \phi(x)$$



Feature vectors can either be tailored to fit a specific dataset, or a more flexible method can be applied generally.

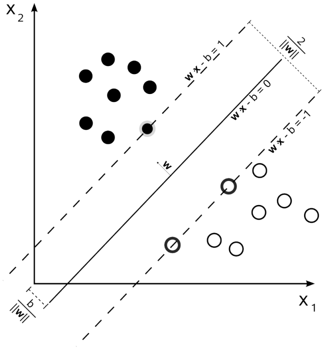
Support Vector Machines

A *Support Vector Machine* is a method that seeks to find the "best" linear separator between two sets of data.



Support Vector Machines

The “best” linear separator in this case is the line/plane/hyperplane that maximizes the *margin* to the observations. The margin is the minimum distance between the decision boundary and the closest point of data in each class. The resulting boundary is both relatively as distant as possible from the training data and balanced towards each class.



x_2
x_1
$\mathbf{w} \cdot \mathbf{x} - b = 1$
$\mathbf{w} \cdot \mathbf{x} - b = 0$
$\mathbf{w} \cdot \mathbf{x} - b = -1$
$\frac{2}{\|\mathbf{w}\|}$
$\frac{b}{\|\mathbf{w}\|}$
$\mathbf{w}$

SVM Formulation

It has been shown that SVM is equivalent to the following optimization problems:

Optimization

$$\begin{aligned} \max_{\gamma, w, b} \quad & \gamma \\ s.t. \quad & y^{(i)}\left(w^T x^{(i)} + b\right) \geq \gamma \quad , \quad i = 1, \ldots, m \\ & \|w\| = 1 \end{aligned}$$

Each $x^{(i)}$ is a training observation and $y^{(i)}$ is a label (-1, 1) to denote the group to which the observation belongs.

Note: we must optimize over a choice of direction w, which becomes computationally intractible in high dimensions.

SVM Formulation

An alternative second form of the optimization problem has some unique benefits.

Optimization

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{m} \alpha_i - \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle \\ s.t. \quad & \alpha_i \geq 0 \quad , \ i = 1, ..., m \\ & \sum_{i=1}^{m} \alpha_i y^{(i)} = 0 \end{aligned}$$

Here, we only have m parameters over which to optimize, and the only information required is the inner product $K\left(x^{(i)}, x^{(j)}\right) = \langle x^{(i)}, x^{(j)} \rangle$.
Explicit vectors $x^{(i)}$ not necessarily required!

Kernel Trick

Many methods based on seemingly complicated, high-dimensional optimizations can be significantly reduced in complexity, down to an easily computable form.
- Support vector machines are a key example.

This allows us to use the convenient properties of feature vectors in high dimensions without suffering the consequences in computation.

- So called "Kernel trick"

Kernel Trick

Method:

1. Map data onto feature vectors. $(x_i, ..., x_n) \to (\phi(x_i), \phi(x_i))$. The new vectors could have very high (or infinite) dimensions.
2. Calculate dot products $\langle \cdot, \cdot \rangle$
3. Perform optimization

- If a convenient form for $\langle \phi(x), \phi(y) \rangle$ exists, we can combine the first two steps, and perform the optimization without explicitly calculating the feature vector.

Kernel Trick

Kernel

A function that can be written as

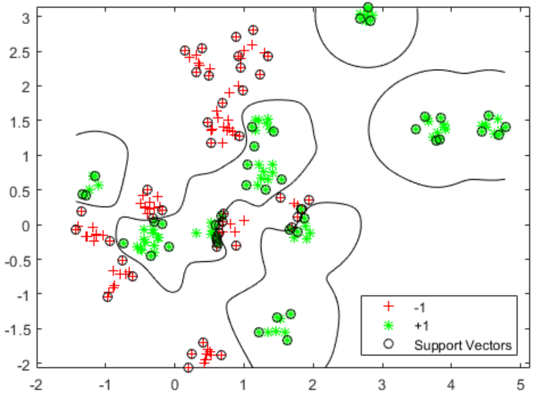
$$K(x, y) = \langle \phi(x), \phi(y) \rangle$$

is called a *kernel*, and are a measure of "similarity" between datapoints.

Popular kernels: $e^{-\frac{1}{2\gamma}||x-y||^2}$, $||x \cdot y + 1||^d$

- Choice of kernel affects the flexibility of the final method

Kernel Trick



Gaussian kernel SVM

Linear Regression Classifiers

Let y be a new observation compared against several classes:

$$\begin{gathered} x_{(1,1)}, \dots, x_{(1,m_1)} \\ \vdots \\ x_{(k,1)}, \dots, x_{(k,m_k)} \end{gathered}$$

- Assign y to group i if y is "closest" to that group.
- Alternatively, if the members of group i can be used to "best-approximate" y.

Linear Regression Classifiers

Linear Regression Classifier:

1. Approximate y as a linear combination of the members in group i.

$$y \approx A_i \alpha_i$$

$$A_i = \left[x_{(i,1)}, \dots, x_{(i,m_i)}\right]$$

.

2. The least squares solution is $\left(A_i^T A_i\right)^{-1} A_i^T y = \hat{y}_i$.
3. Assign y to group i if $||y - \hat{y}_i||$ is smallest.

Kernelized LASSO Classifier

Kernelized LASSO Classifier:

1. Approximate y as a linear combination of the members in group i.

$$y \approx A_i \alpha_i$$

$$A_i = \left[x_{(i,1)}, \ldots, x_{(i,m_i)}\right]$$

.

2. The KLASSO solution $\hat{y}_i$ solves the optimization problem:

KLASSO

$$\text{minimize } ||y - K\alpha||^2 + \lambda ||\alpha||_1$$

where K is a matrix with $K_{ij} = K\left(x_{(i)}, x_{(j)}\right)$.

3. Assign y to group i if $||y - \hat{y}_i||$ is smallest.

Kernelized LASSO Classifier

Benefits:

- Sparsity - only a select number of examples will be used for classification, useful for large, diverse training classes.
- Nonlinearity
- Computationally efficient

Conclusion

Kernel methods offer a way of extending linear methods into the non-linear setting.

They offer flexibility to adapt existing methods to more complex data, while maintaining computational efficiency.

A wide variety of methods are available (scikit-learn, R/e1071)

- Caveat: Recent developments in optimization (stochastic gradient descent) have made explicitly calculating feature vectors more promising when scaling to large data sizes.

References

V. Roth (2004)
The generalized LASSO
IEEE Transactions on Neural Networks 15(1), 16 – 28.

J. Xu and J. Yin (2013)
Kernel least absolute shrinkage and selection operator regression classifier for pattern classification
IET Computer Vision 7(1), 48 – 55.

Cortes, C. & Vapnik (1995)
Support-Vector Networks
V. Machine Learning 20(273)