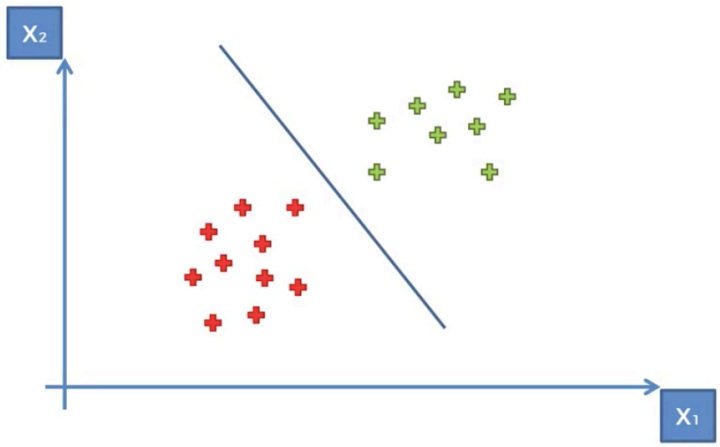
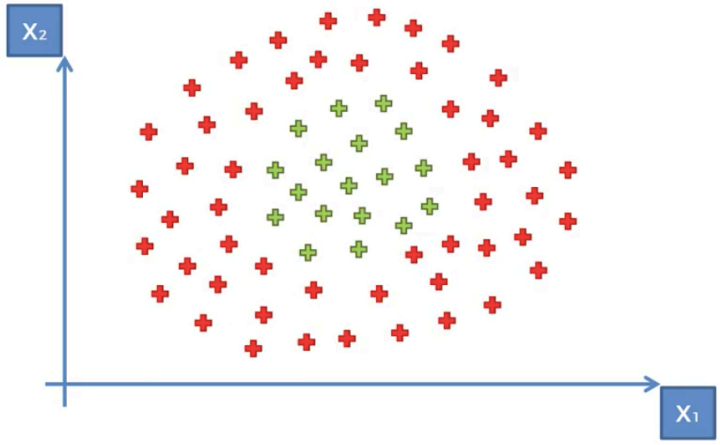
# Kernel SVM Intuition



## SVM separates well these points



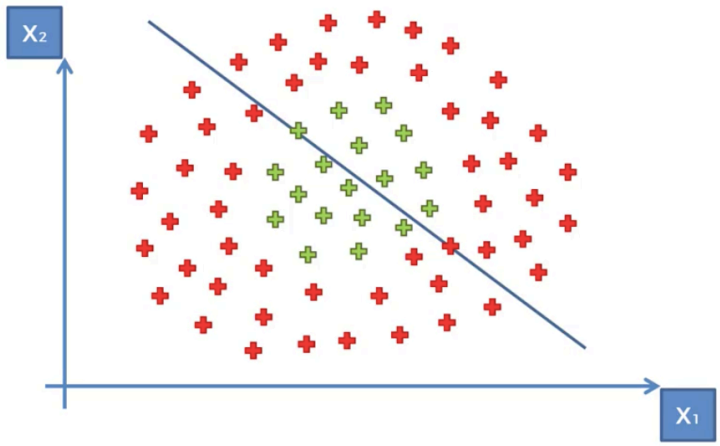
# What about these points ?



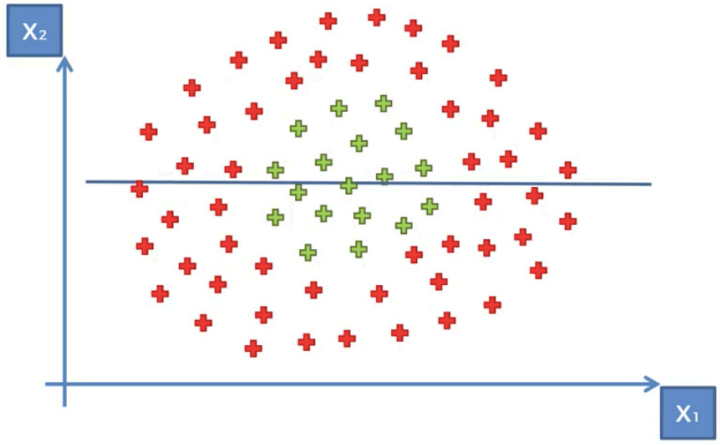




# What about these points ?



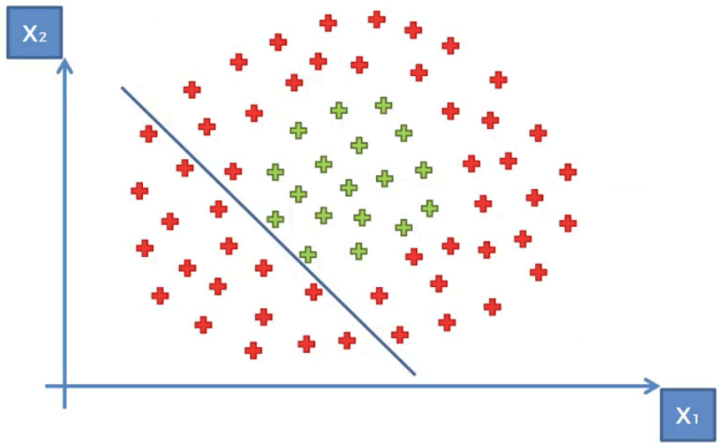
# What about these points ?



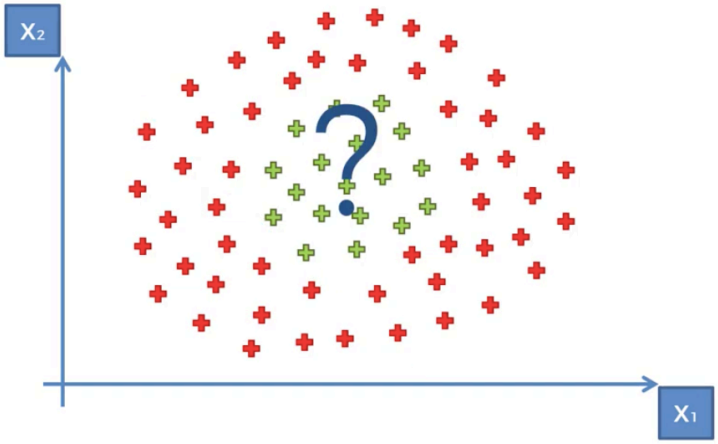




# What about these points ?



# What about these points ?

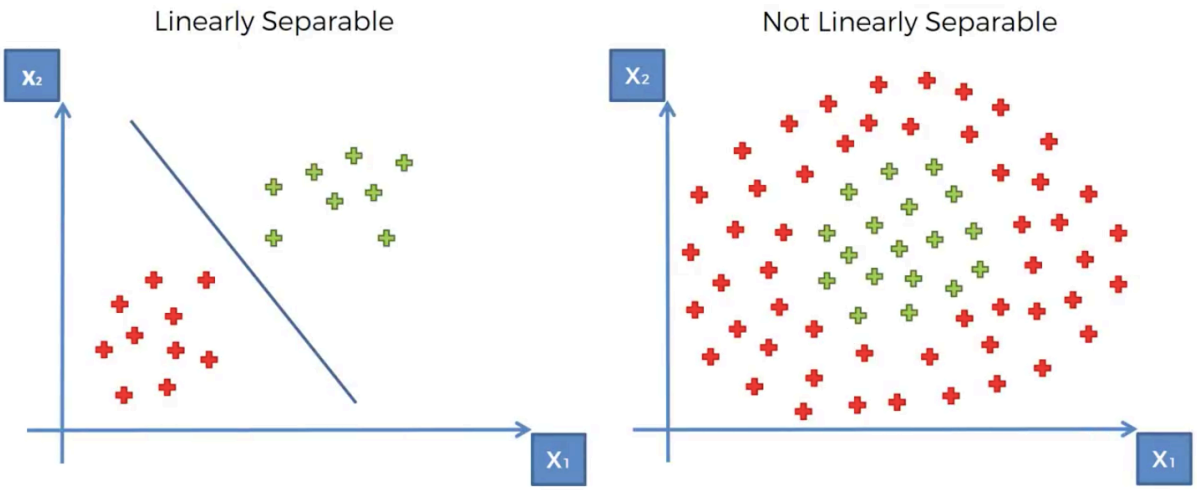





# Why ?

Because the data points are not LINEARLY SEPARABLE

## Linear Separability






A Higher-Dimensional Space

## A Higher-Dimensional Space

# Mapping to a Higher Dimension




# Mapping to a Higher Dimension

$f = x - 5$

# Mapping to a Higher Dimension

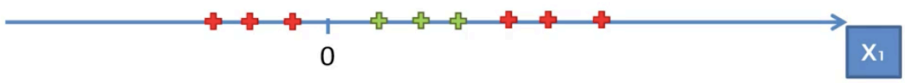
$f = x - 5$



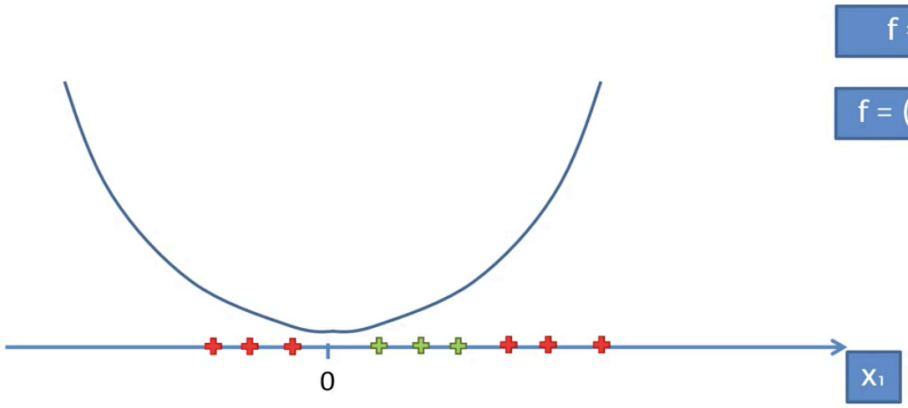

# Mapping to a Higher Dimension

$f = x - 5$

$f = (x - 5)^2$



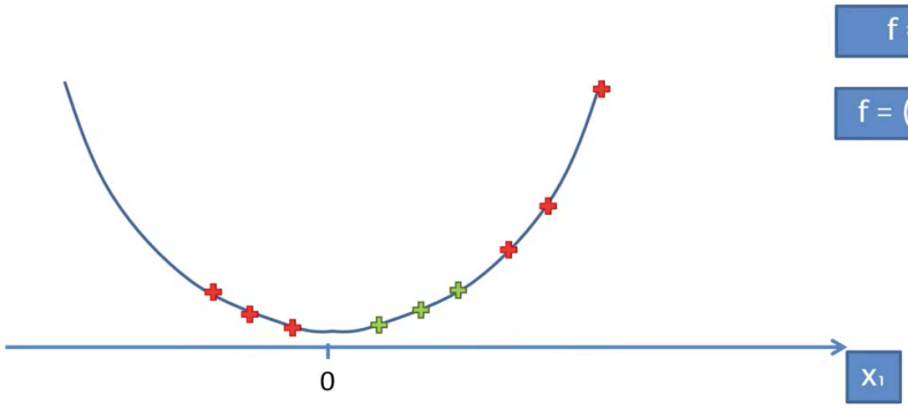
# Mapping to a Higher Dimension



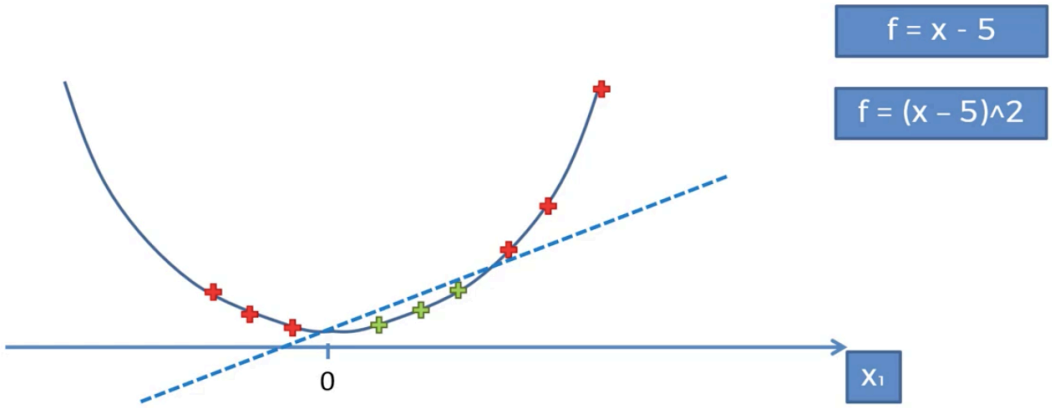




# Mapping to a Higher Dimension

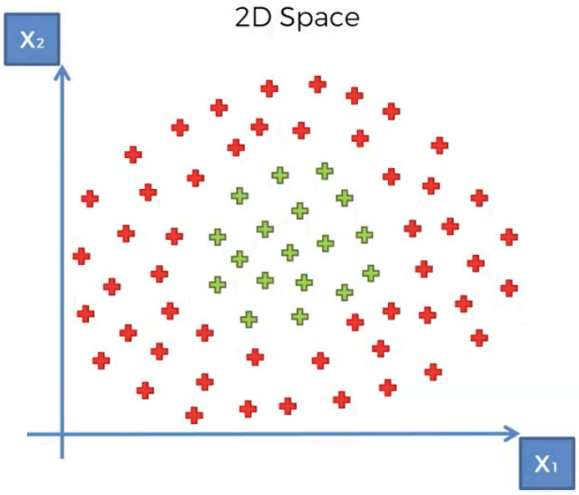


# Mapping to a Higher Dimension

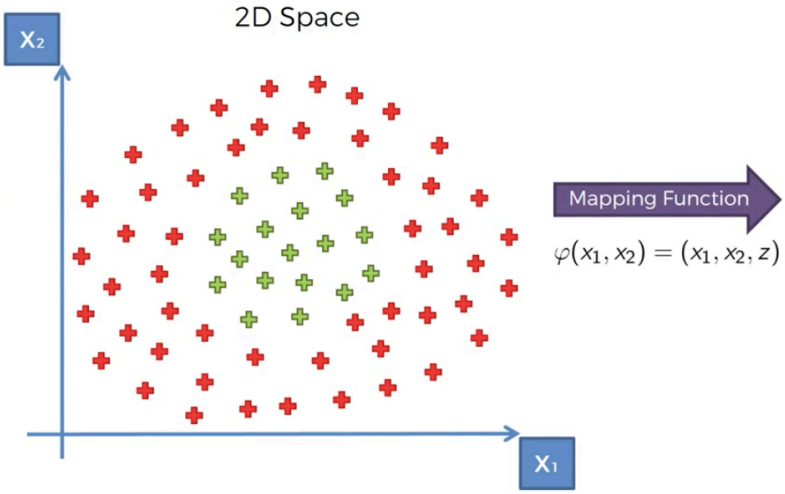





# Mapping to a Higher Dimension



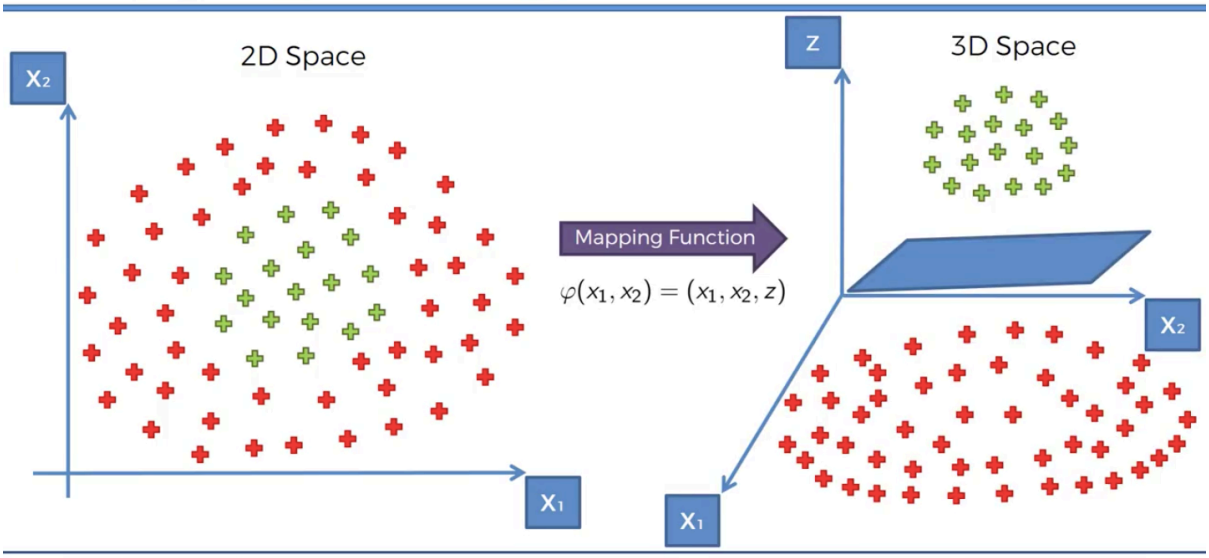
# Mapping to a Higher Dimension



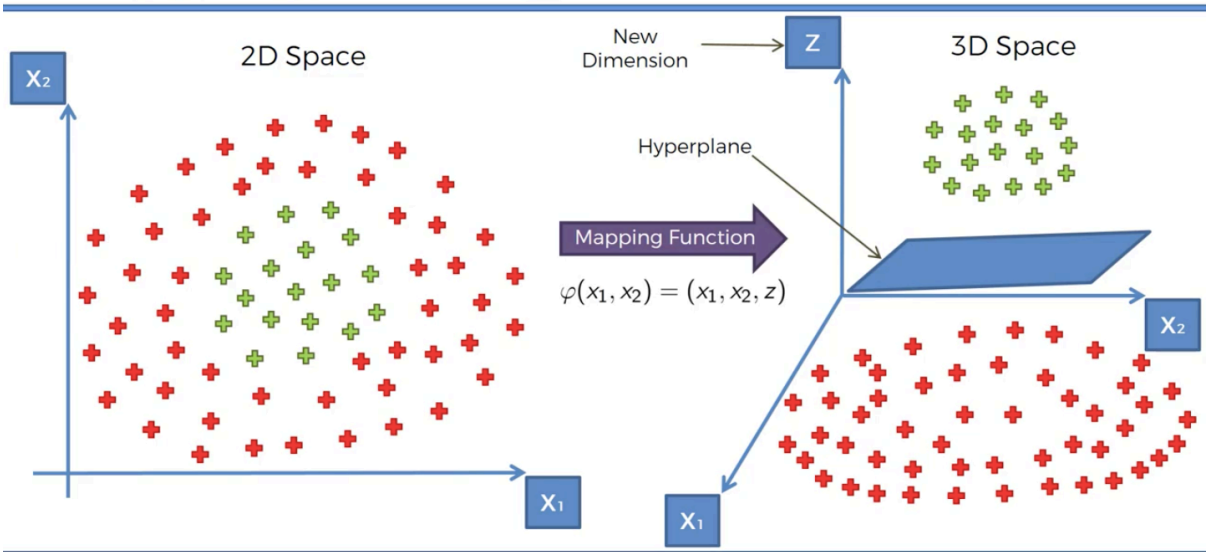




# Mapping to a Higher Dimension



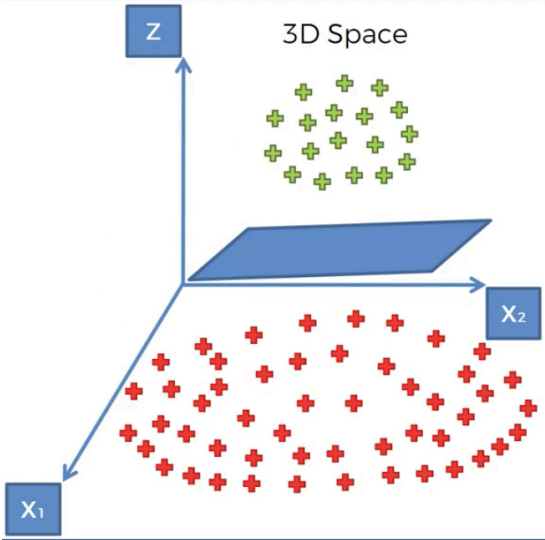
# Mapping to a Higher Dimension



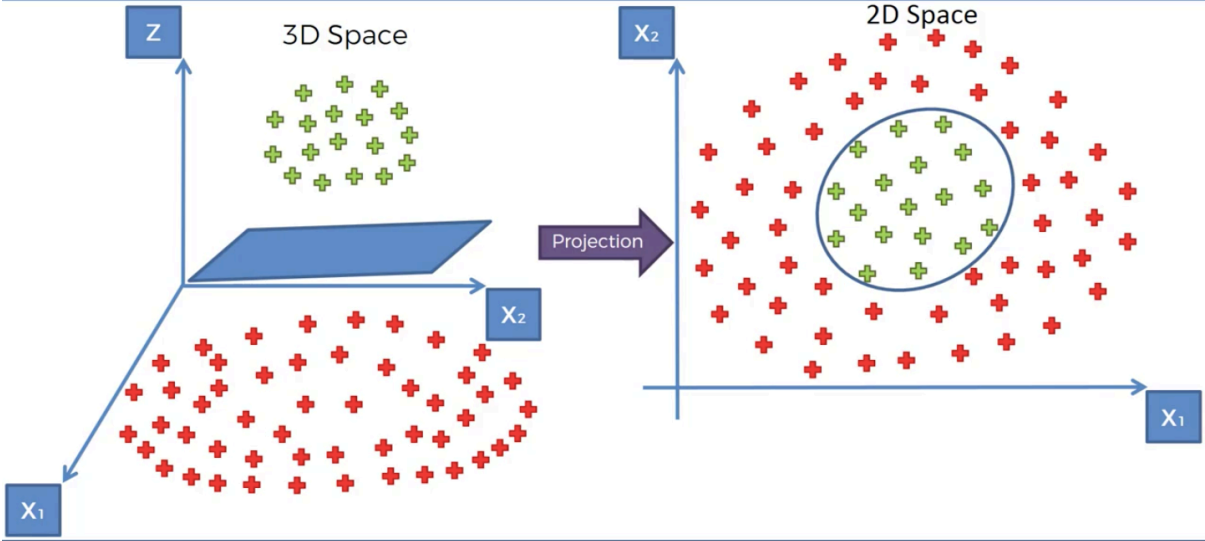




# Projecting back to 2D Space



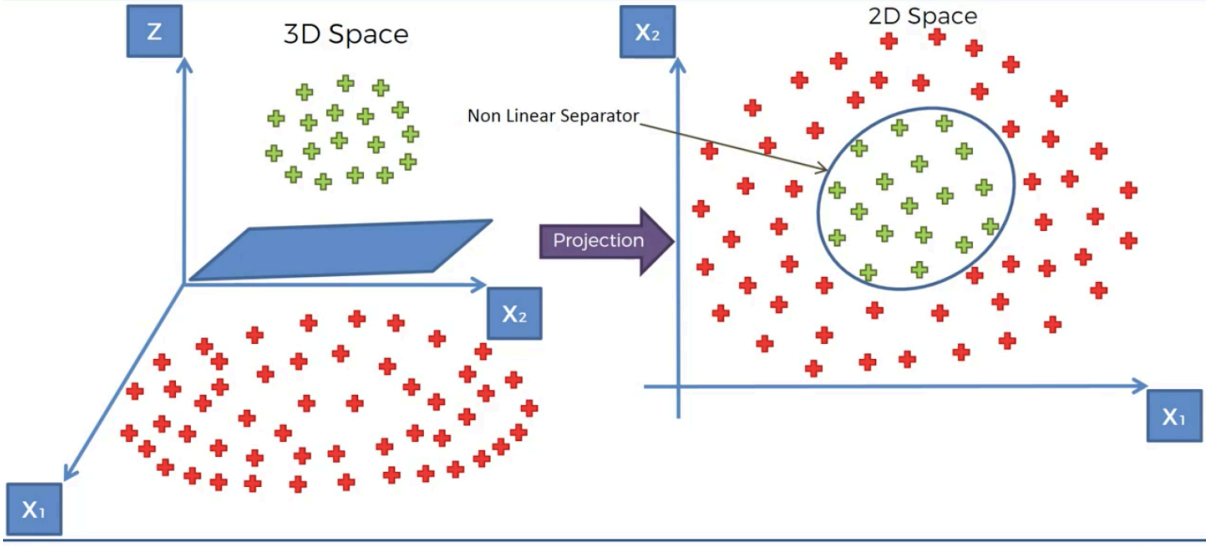
# Projecting back to 2D Space







# Projecting back to 2D Space



## But there is a catch...

Mapping to a Higher Dimensional Space can be highly compute-intensive





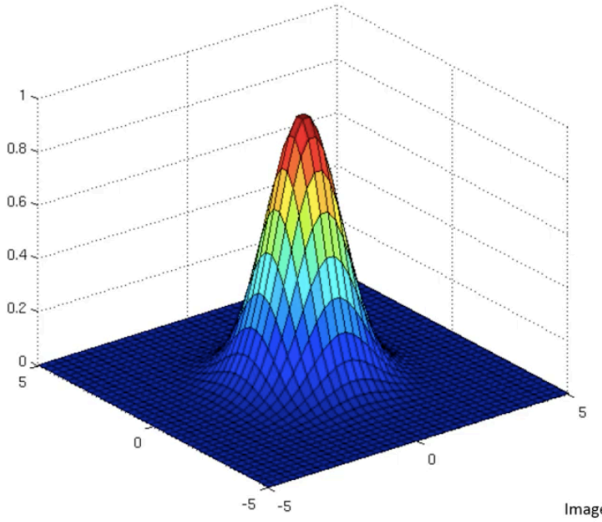
# The Kernel Trick

# The Gaussian RBF Kernel

$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x}-\vec{l}^i\|^2}{2\sigma^2}}$$



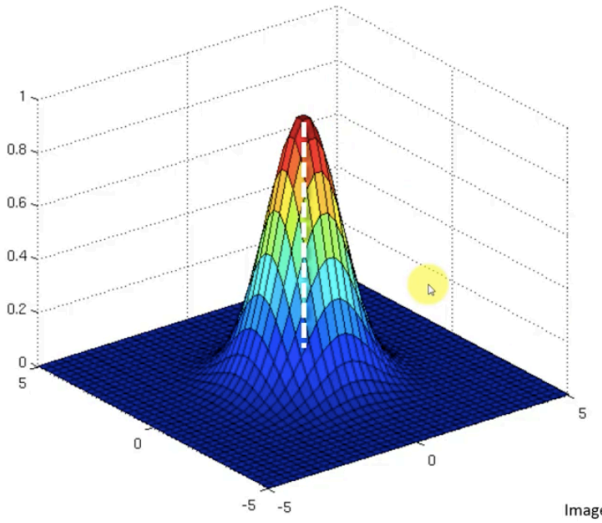
# The Gaussian RBF Kernel




$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x}-\vec{l}^i\|^2}{2\sigma^2}}$$

Image source: http://www.cs.toronto.edu/~duvenaud/cookbook/index.html

# The Gaussian RBF Kernel

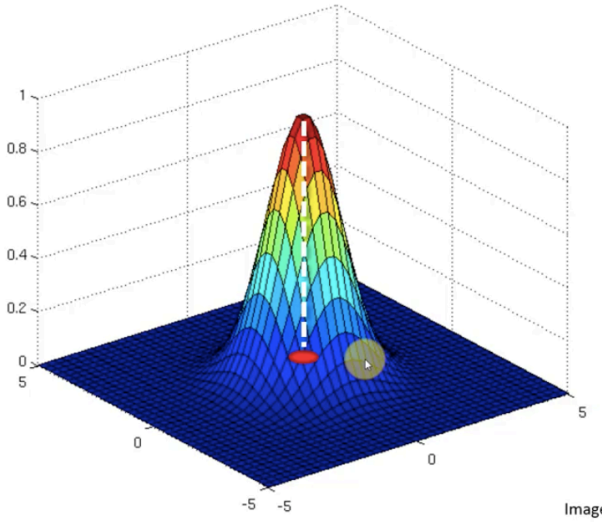



$$K(\vec{x}, \vec{l}^{i}) = e^{-\frac{\|\vec{x}-\vec{l}^{i}\|^2}{2\sigma^2}}$$

Image source: http://www.cs.toronto.edu/~duvenaud/cookbook/index.html




# The Gaussian RBF Kernel



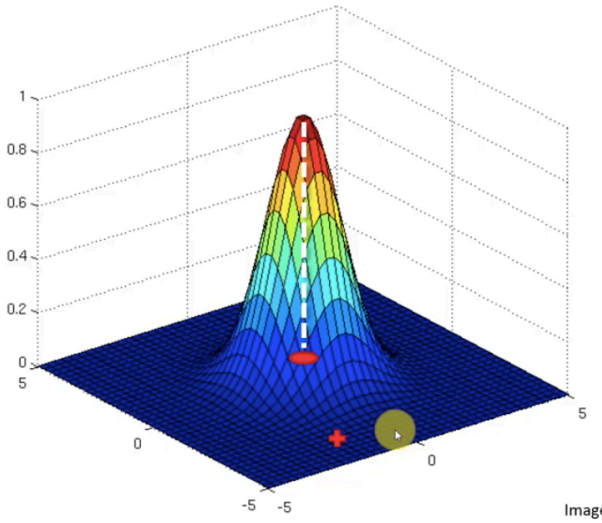

$$K(\vec{x}, \vec{l}^{i}) = e^{-\frac{\|\vec{x}-\vec{l}^{i}\|^2}{2\sigma^2}}$$

Image source: http://www.cs.toronto.edu/~duvenaud/cookbook/index.html

Machine Learning A-Z

© SuperDataScience

# The Gaussian RBF Kernel

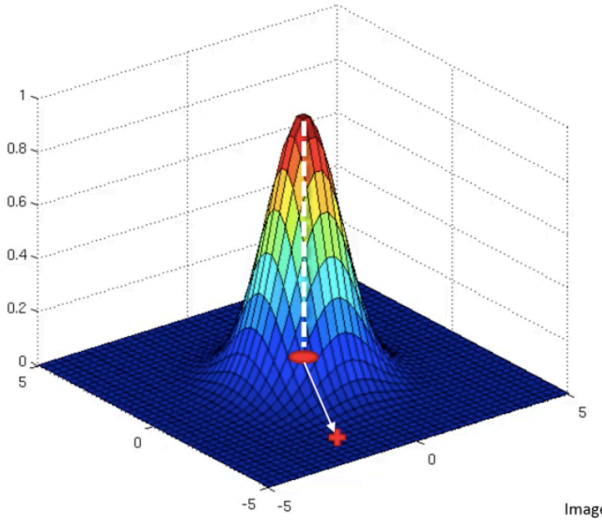



$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x}-\vec{l}^i\|^2}{2\sigma^2}}$$

Image source: http://www.cs.toronto.edu/~duvenaud/cookbook/index.html



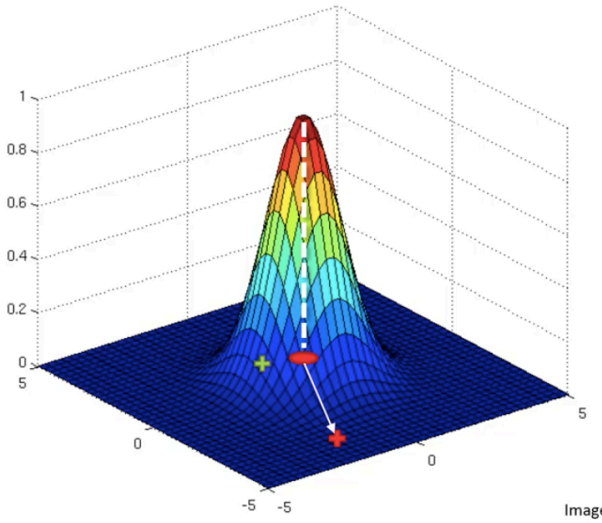
# The Gaussian RBF Kernel




$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x}-\vec{l}^i\|^2}{2\sigma^2}}$$

Image source: http://www.cs.toronto.edu/~duvenaud/cookbook/index.html

# The Gaussian RBF Kernel

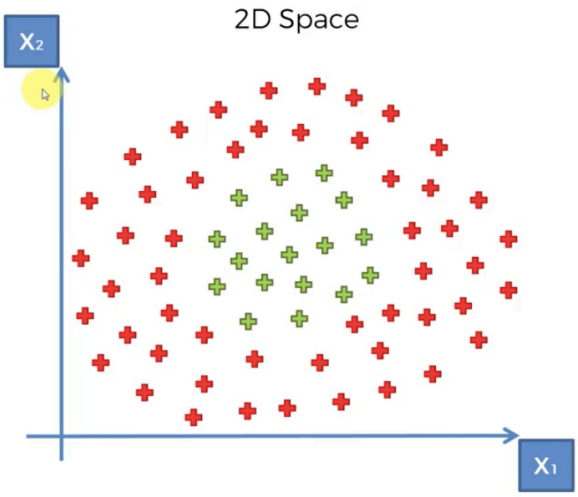


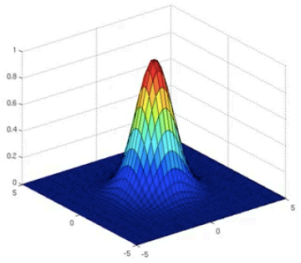

$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x}-\vec{l}^i\|^2}{2\sigma^2}}$$

Image source: http://www.cs.toronto.edu/~duvenaud/cookbook/index.html



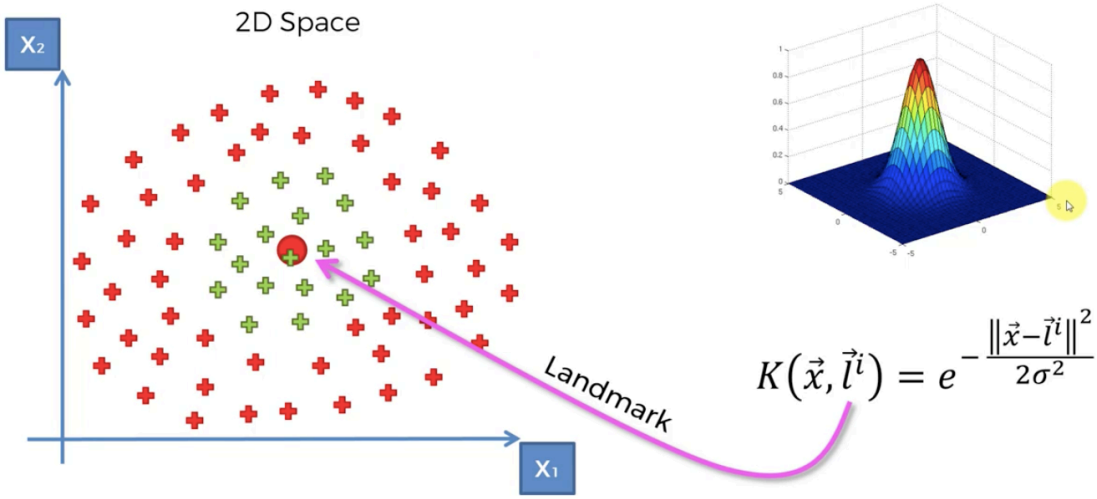

# The Gaussian RBF Kernel



$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x}-\vec{l}^i\|^2}{2\sigma^2}}$$

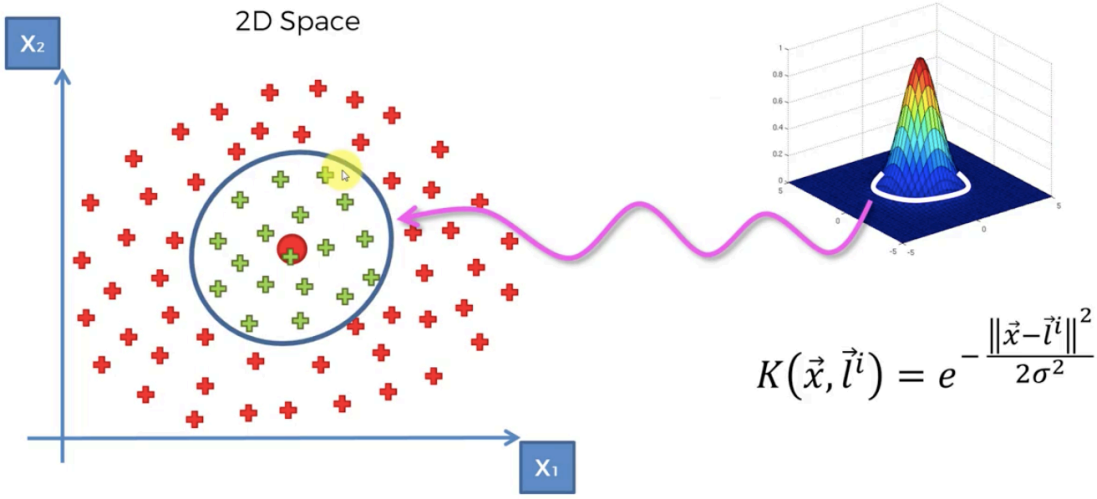
# The Gaussian RBF Kernel



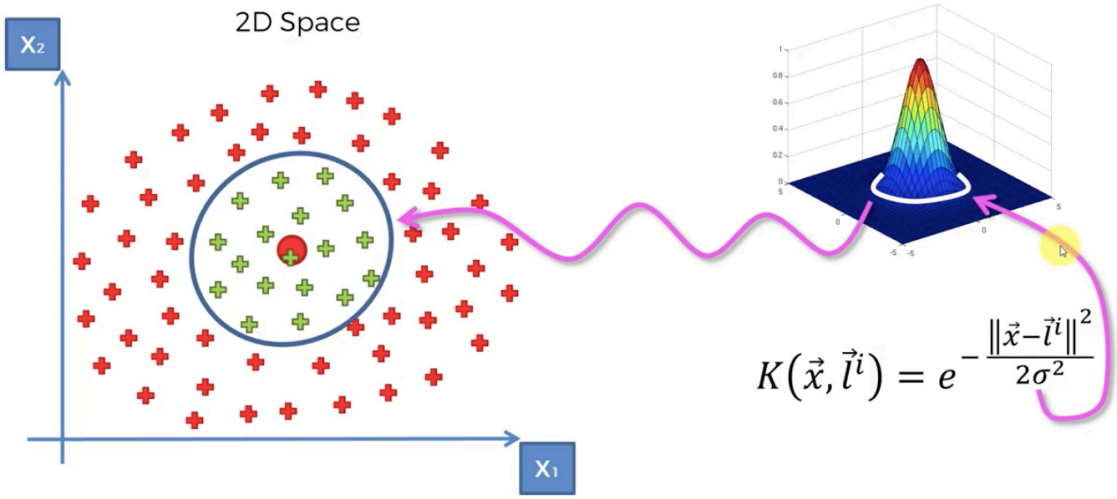




# The Gaussian RBF Kernel



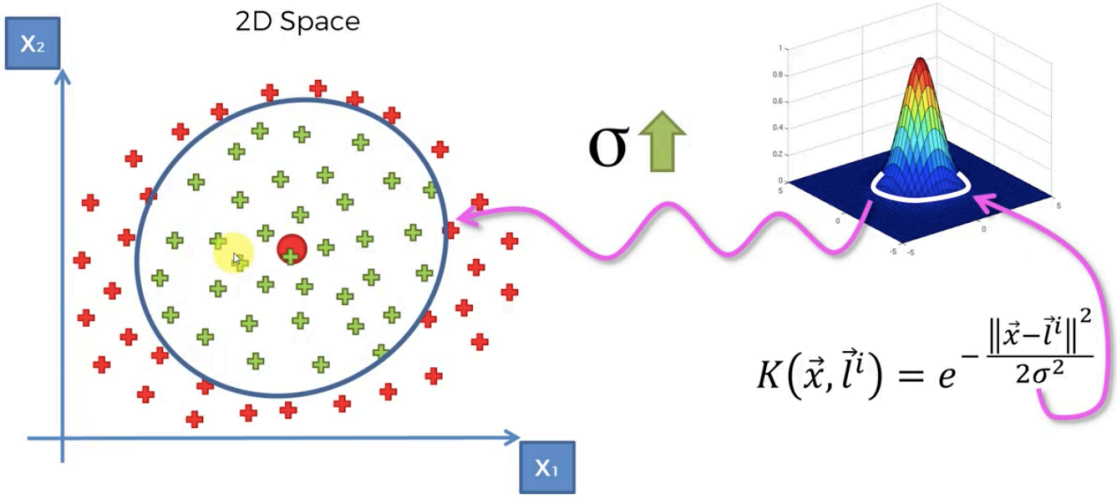
# The Gaussian RBF Kernel



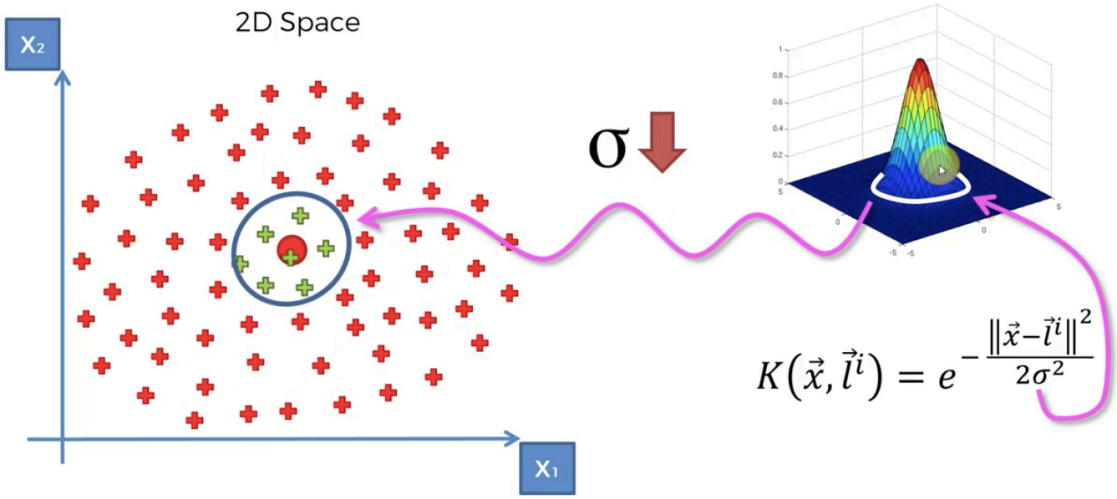




# The Gaussian RBF Kernel



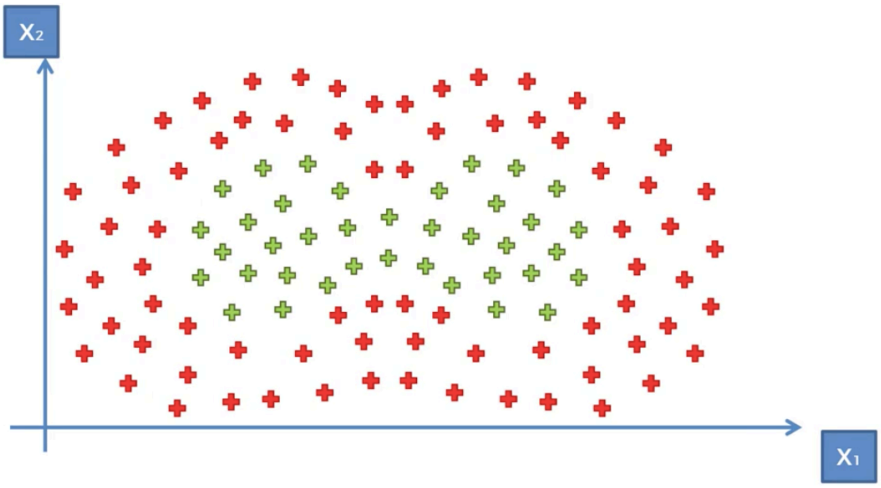
# The Gaussian RBF Kernel



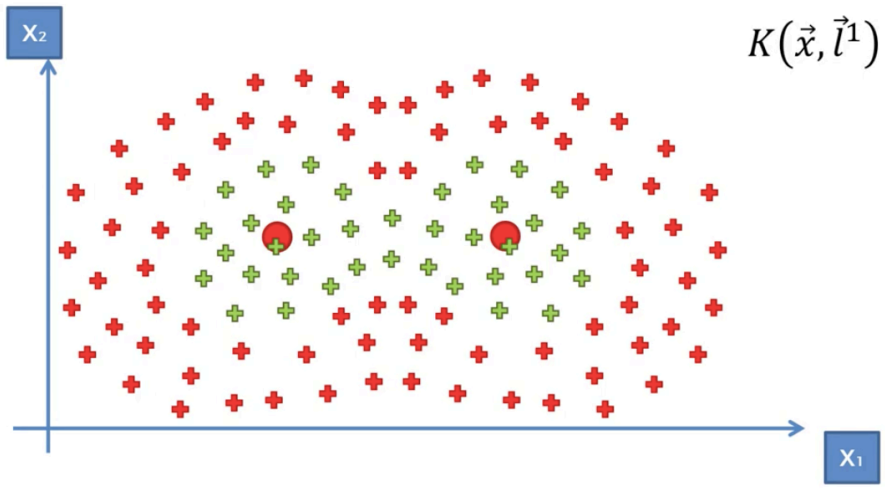




# The Gaussian RBF Kernel



# The Gaussian RBF Kernel

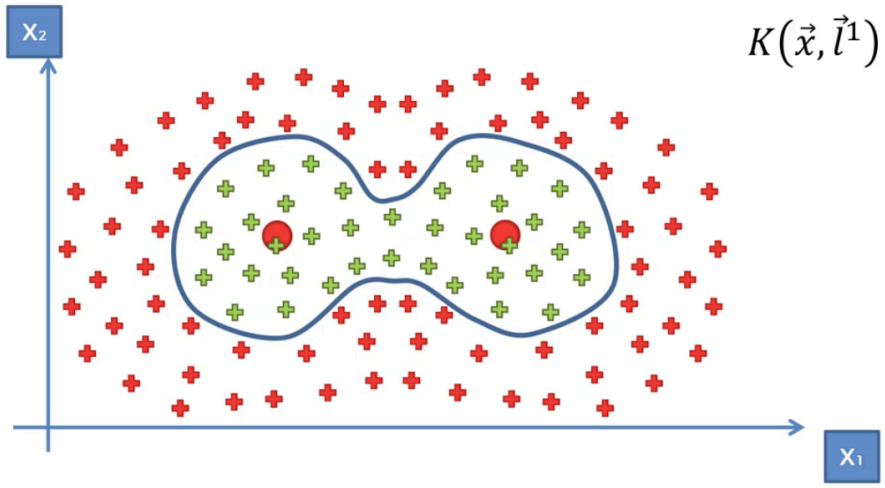



$$K(\vec{x}, \vec{l}^1) + K(\vec{x}, \vec{l}^2)$$

*(Simplified Formula)*




# The Gaussian RBF Kernel

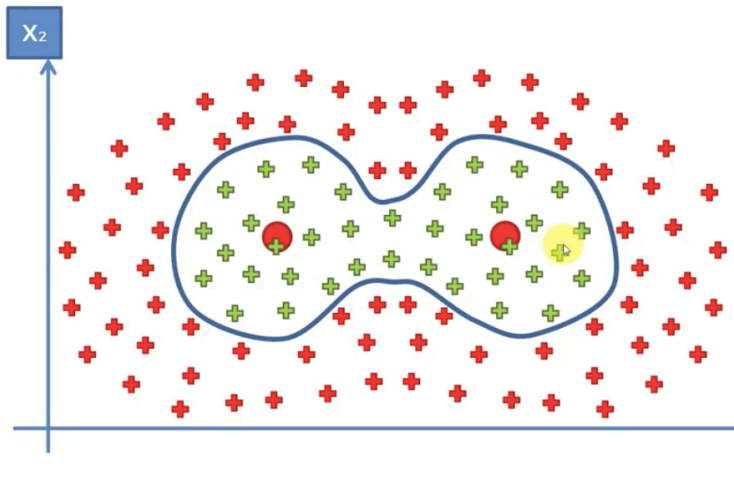



$$K(\vec{x}, \vec{l}^1) + K(\vec{x}, \vec{l}^2)$$

*(Simplified Formula)*

Machine Learning A-Z
© SuperDataScience

# The Gaussian RBF Kernel




$$K(\vec{x}, \vec{l}^1) + K(\vec{x}, \vec{l}^2)$$

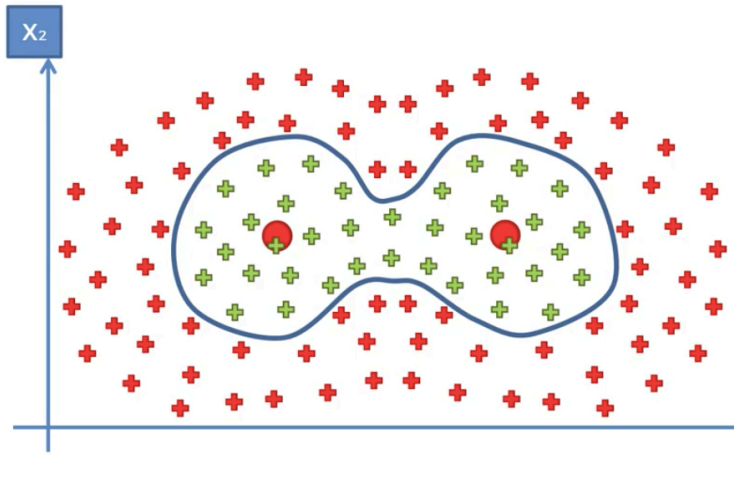
*(Simplified Formula)*

Green when:

$$K(\vec{x}, \vec{l}^1) + K(\vec{x}, \vec{l}^2) > 0$$




# The Gaussian RBF Kernel




$$K(\vec{x}, \vec{l}^1) + K(\vec{x}, \vec{l}^2)$$

*(Simplified Formula)*

Green when:

$$K(\vec{x}, \vec{l}^1) + K(\vec{x}, \vec{l}^2) > 0$$

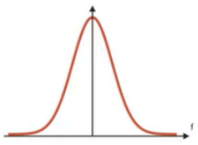
Red when:

$$K(\vec{x}, \vec{l}^1) + K(\vec{x}, \vec{l}^2) = 0$$

# Types of Kernel Functions

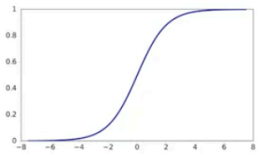


## Types of Kernel Functions



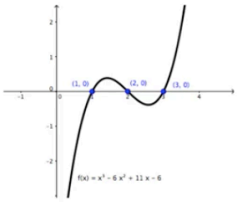
Gaussian RBF Kernel

$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x}-\vec{l}^i\|^2}{2\sigma^2}}$$



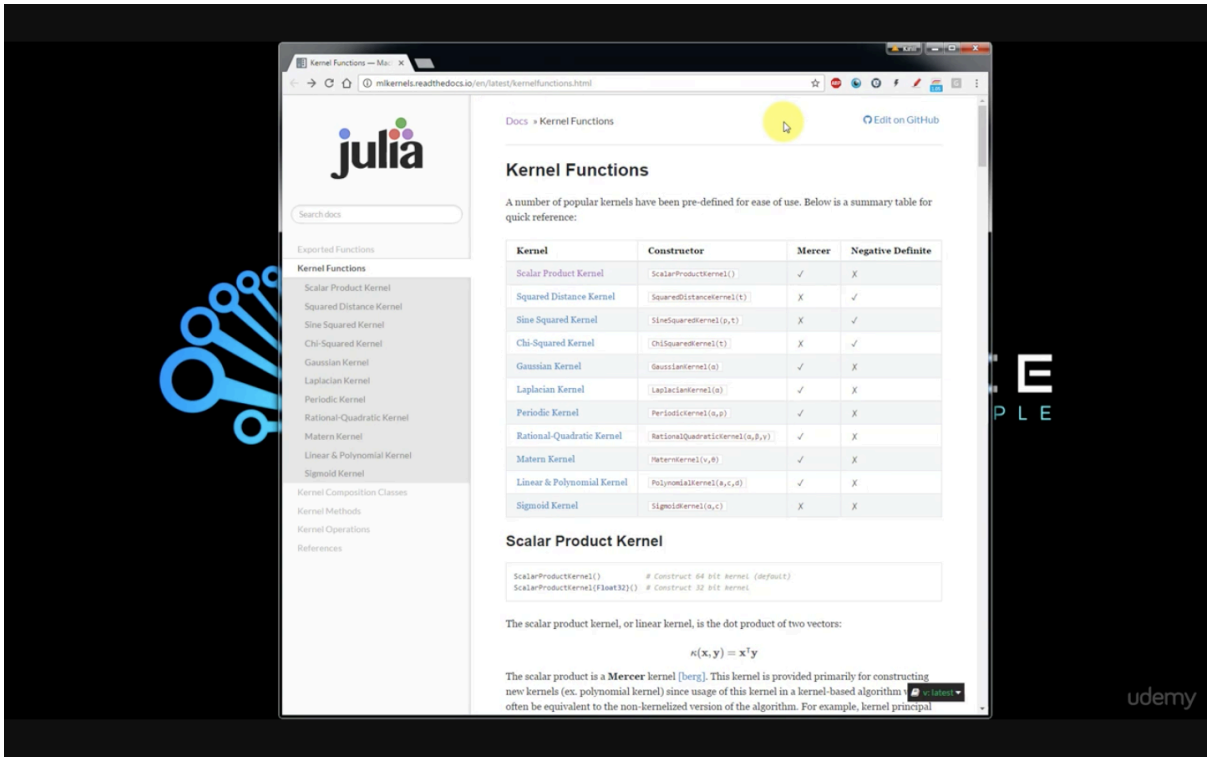
Sigmoid Kernel

$$K(X, Y) = \tanh(\gamma \cdot X^T Y + r)$$



Polynomial Kernel

$$K(X, Y) = (\gamma \cdot X^T Y + r)^d, \gamma > 0$$


Kernel Functions — Ma...
mlkernels.readthedocs.io/en/latest/kernelfunctions.html
julia
Search docs
Exported Functions
Kernel Functions
Scalar Product Kernel
Squared Distance Kernel
Sine Squared Kernel
Chi-Squared Kernel
Gaussian Kernel
Laplacian Kernel
Periodic Kernel
Rational-Quadratic Kernel
Matern Kernel
Linear & Polynomial Kernel
Sigmoid Kernel
Kernel Composition Classes
Kernel Methods
Kernel Operations
References
Docs » Kernel Functions
Edit on GitHub
Kernel Functions
A number of popular kernels have been pre-defined for ease of use. Below is a summary table for quick reference:
Kernel | Constructor | Mercer | Negative Definite
Scalar Product Kernel | ScalarProductKernel() | ✓ | ✗
Squared Distance Kernel | SquaredDistanceKernel(t) | ✗ | ✓
Sine Squared Kernel | SineSquaredKernel(p,t) | ✗ | ✓
Chi-Squared Kernel | ChiSquaredKernel(t) | ✗ | ✓
Gaussian Kernel | GaussianKernel(α) | ✓ | ✗
Laplacian Kernel | LaplacianKernel(α) | ✓ | ✗
Periodic Kernel | PeriodicKernel(α,p) | ✓ | ✗
Rational-Quadratic Kernel | RationalQuadraticKernel(α,β,γ) | ✓ | ✗
Matern Kernel | MaternKernel(ν,θ) | ✓ | ✗
Linear & Polynomial Kernel | PolynomialKernel(a,c,d) | ✓ | ✗
Sigmoid Kernel | SigmoidKernel(a,c) | ✗ | ✗
Scalar Product Kernel
ScalarProductKernel() # Construct 64 bit kernel (default)
ScalarProductKernel(Float32)() # Construct 32 bit kernel
The scalar product kernel, or linear kernel, is the dot product of two vectors:
$$\kappa(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{y}$$
The scalar product is a Mercer kernel [berg]. This kernel is provided primarily for constructing new kernels (ex. polynomial kernel) since usage of this kernel in a kernel-based algorithm
latest
often be equivalent to the non-kernelized version of the algorithm. For example, kernel principal
udemy