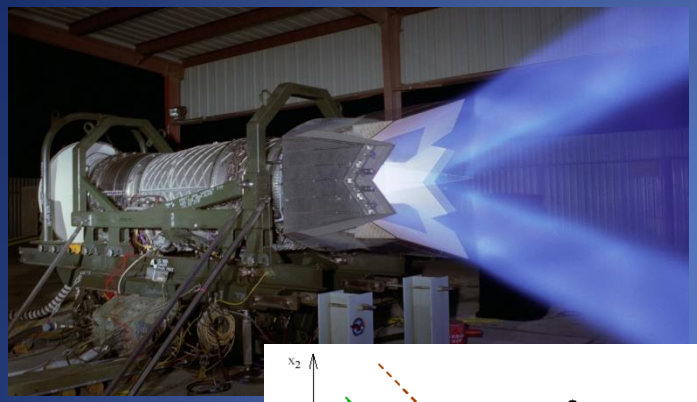
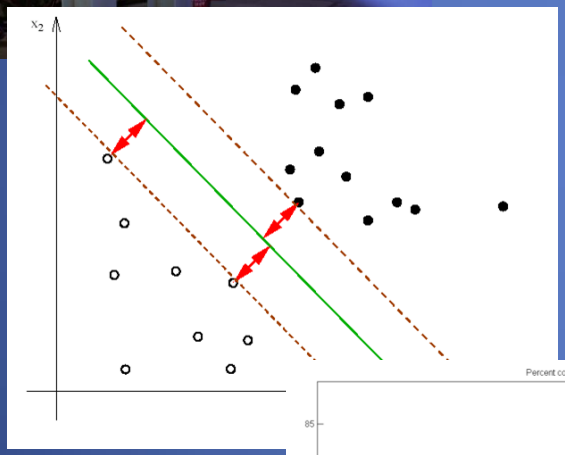
Using Support Vector Machines to determine valid turbine blade geometry

By: David Witman

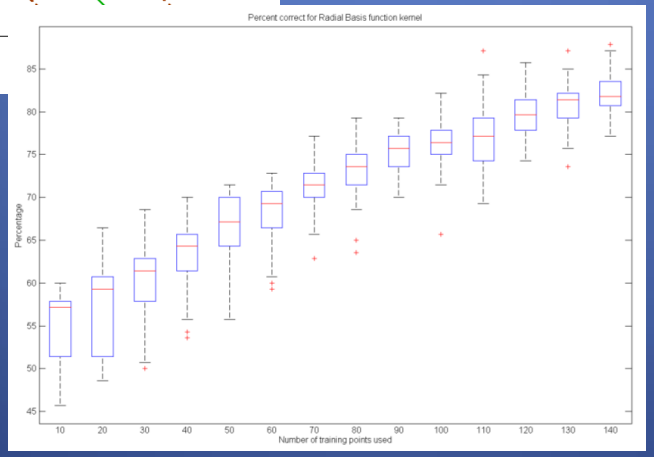
This presentation will concentrate on the use and application of Support Vector Machines SVM's



Background on Gas Turbine Engines

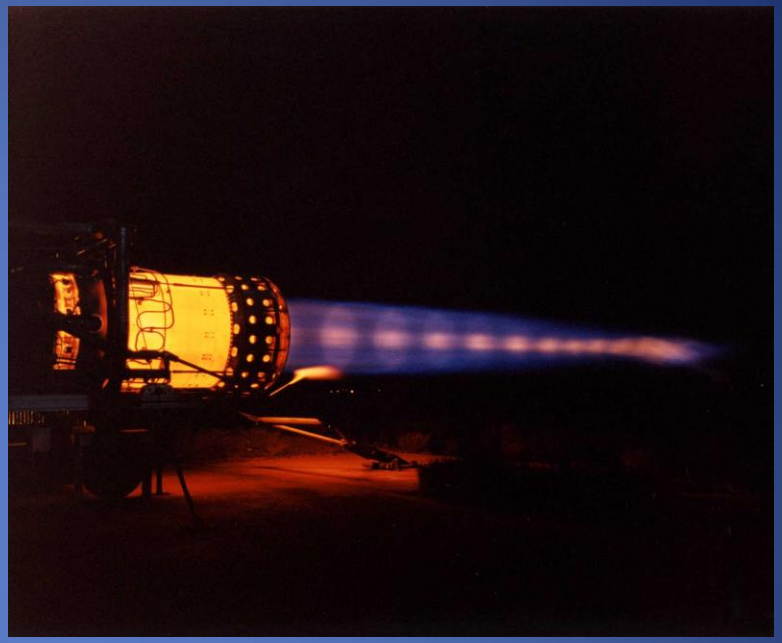


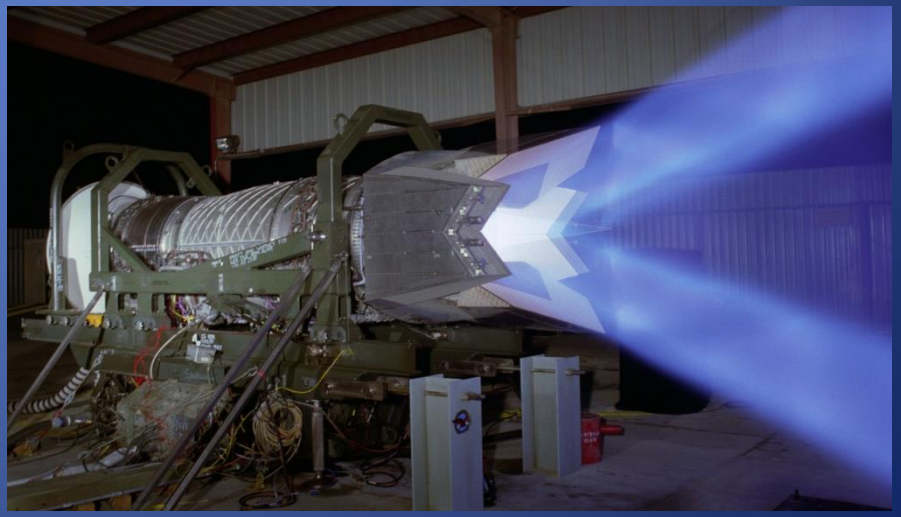
Background on SVM's



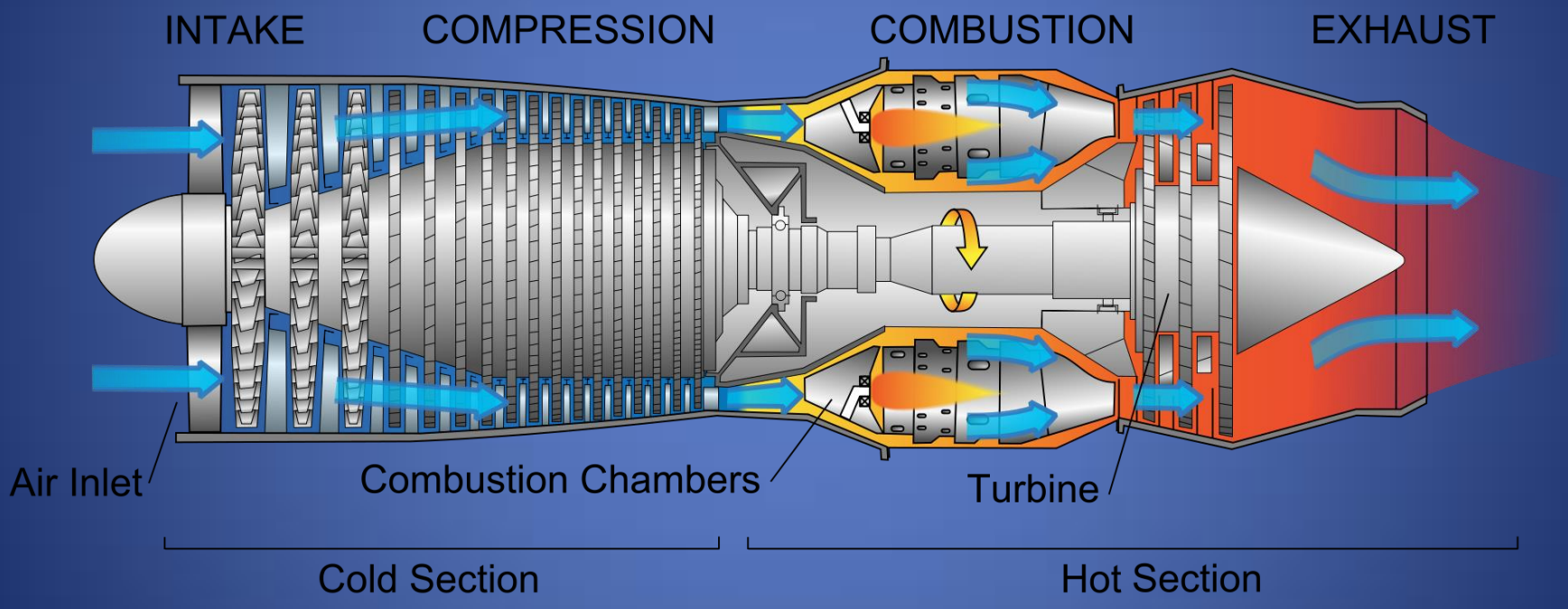
Results!

We can apply SVM's to validate acceptable turbine blade geometry. But first a little background on Gas Turbine Engines

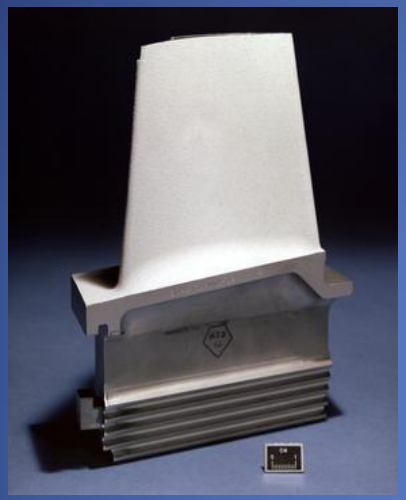


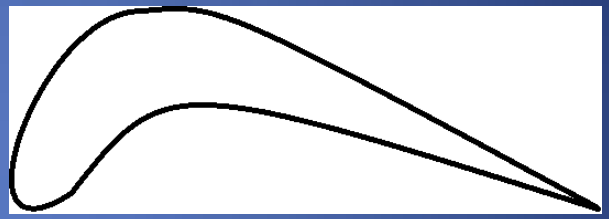


We can apply SVM's to validate acceptable turbine blade geometry. But first a little background on Gas Turbine Engines



Why is geometry so important to designing an optimal turbine blade/vane?

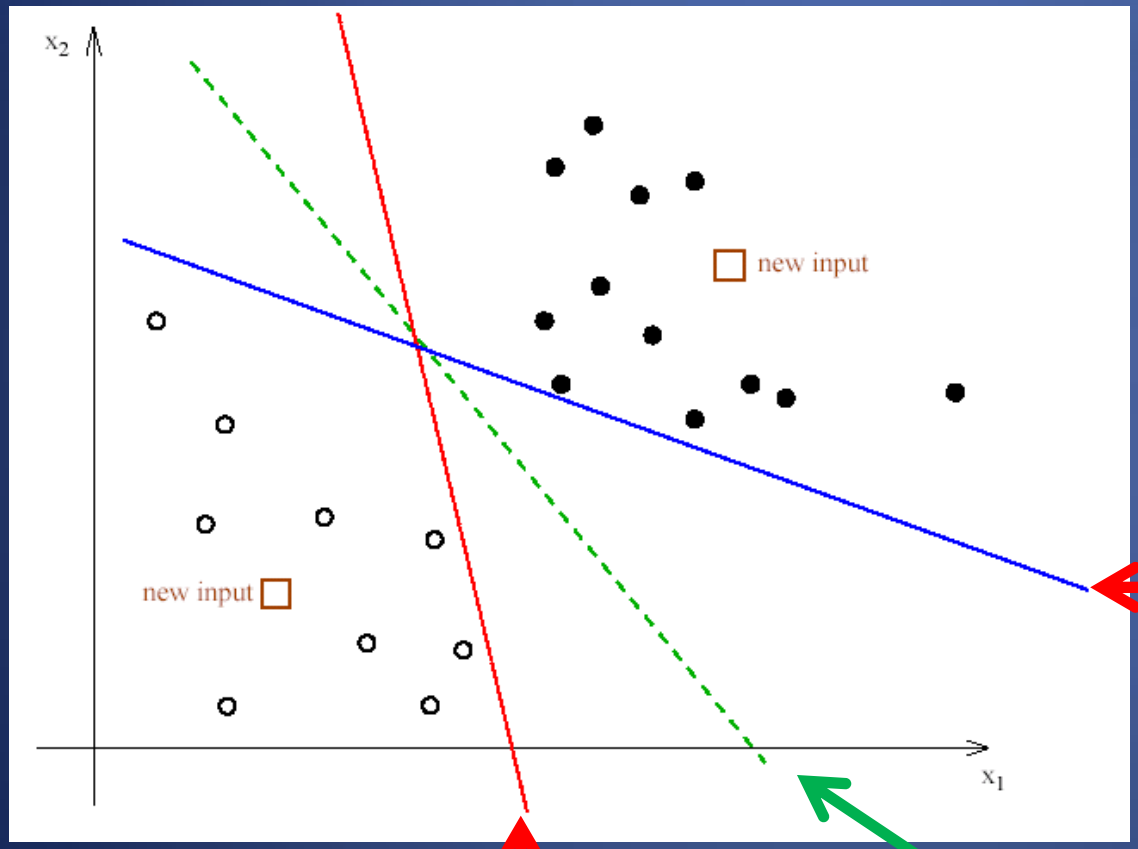




Things to consider when designing an effective turbine blade:

- Inlet temperatures exceed 1000^{o}F
- Very high centrifugal forces
- Corrosion major issue
- Weight
- Creep
- Metal Fatigue

SVM's are a classification technique designed to generate optimal decision boundaries

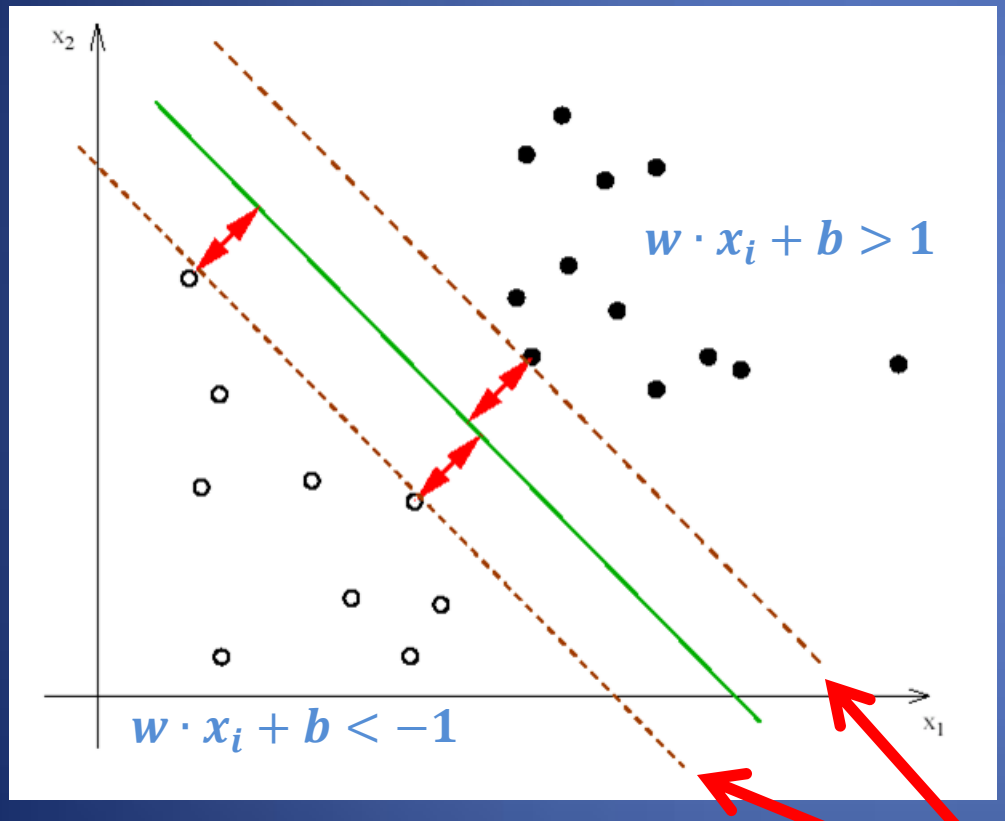


Decision Boundary

Decision Boundary

Optimal Decision Boundary aka Hyperplane

SVM's are a classification technique designed to generate optimal decision boundaries



To define the Hyperplane:

$$w \cdot x_i + b = 0$$

So that when we use a test point (x_i) the value is either:

$$w \cdot x_i + b < 0$$

or

$$w \cdot x_i + b > 0$$

Support Vectors

SVM's are a classification technique designed to generate optimal decision boundaries

For the linear case, we can form an optimization based on the Lagrange multipliers

Maximize:

$$L_D = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j \, y_i y_j \mathbf{x}_i \mathbf{x}_j$$

Subject to the constraints:

$$\lambda_i \geq 0$$

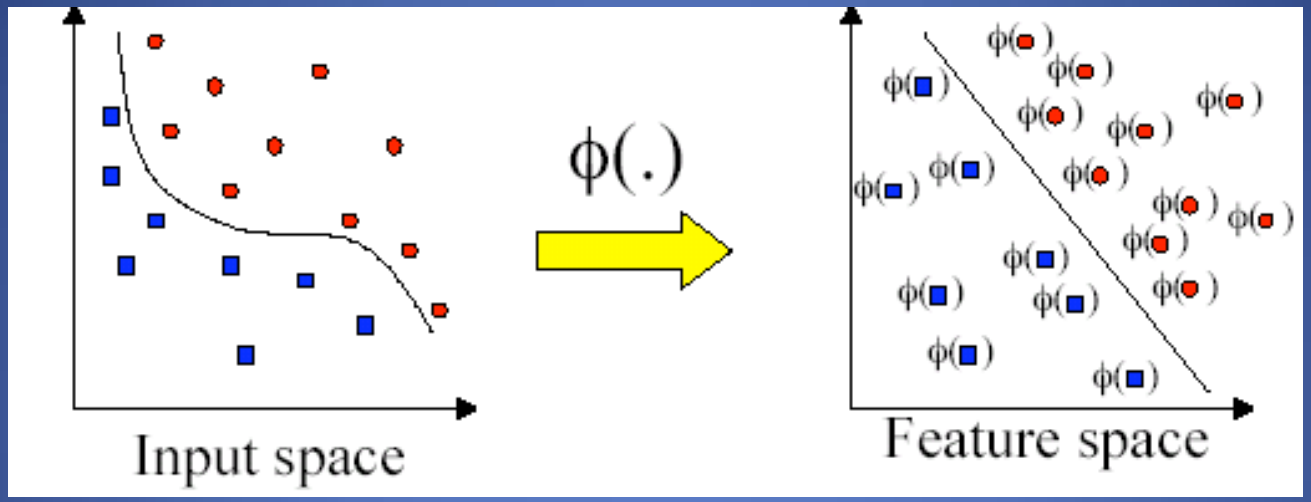
$$\lambda_i [y_i (\mathbf{w} \cdot \mathbf{x}_i + \mathrm{b}) - 1] = 0$$

L_D is known as a dual problem that can be solved using numerical techniques like quadratic programming

$$f(x) = \frac{1}{2} x^T Q x + y^T x$$

Subject to the same constraints as above

But what if the data is non-linear? We can use the Kernel Trick



Common types of Kernels:

Polynomial Kernel

$$k(x,y) = (\alpha x^T y + c)^d$$

Gaussian Kernel:

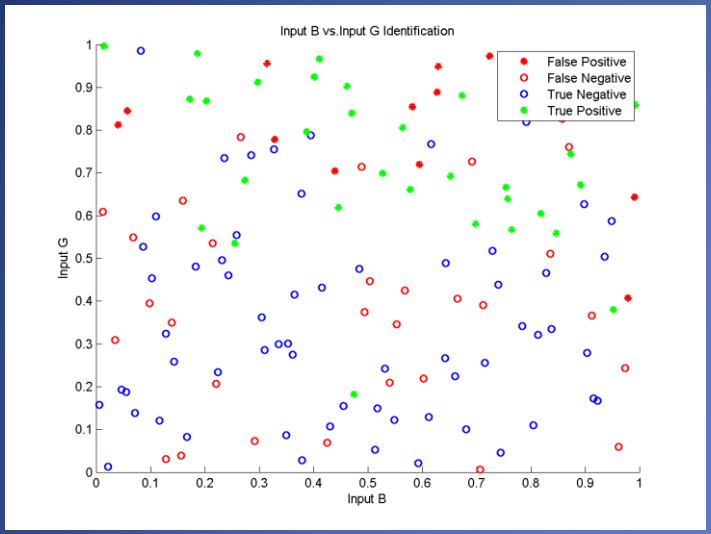
$$k(x,y) = exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$$

Exponential Kernel

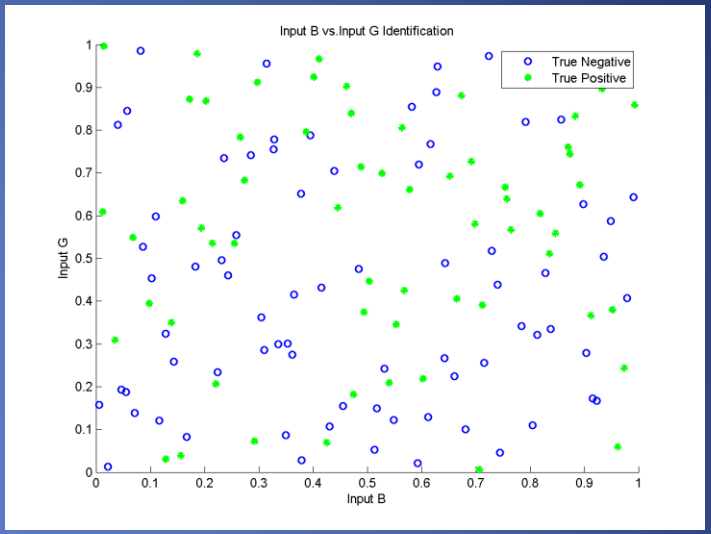
$$k(x,y) = exp\left(-\frac{\|x-y\|}{2\sigma^2}\right)$$

Examining some 2D plots shows how we can classify the different points in 2D space

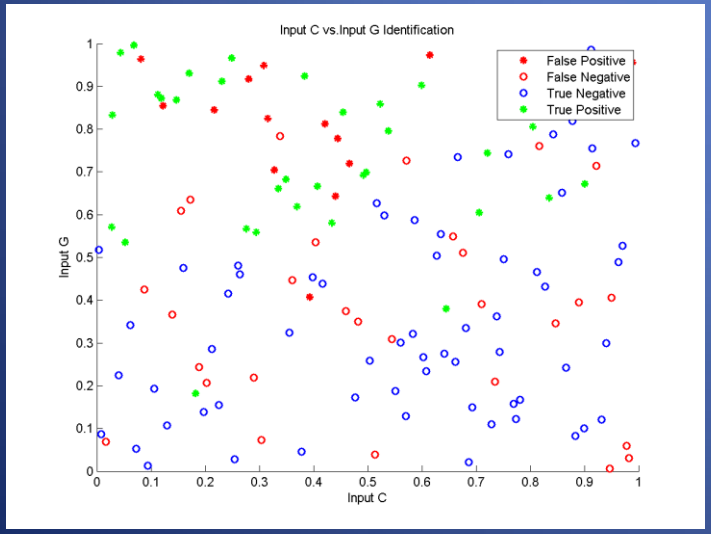
Linear



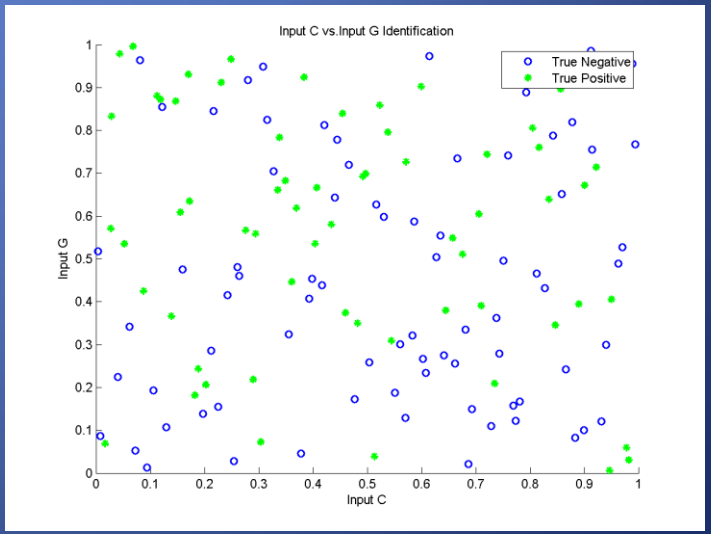
Polynomial



Linear



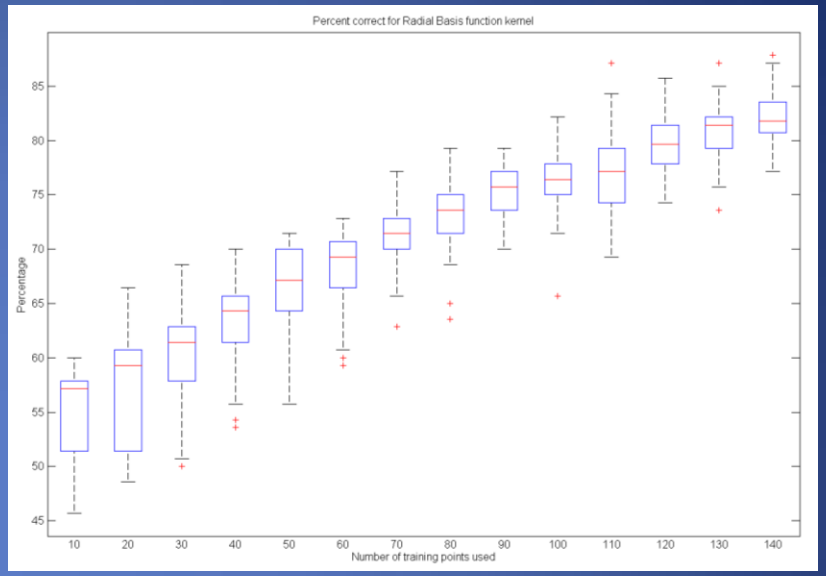
Radial Basis Function



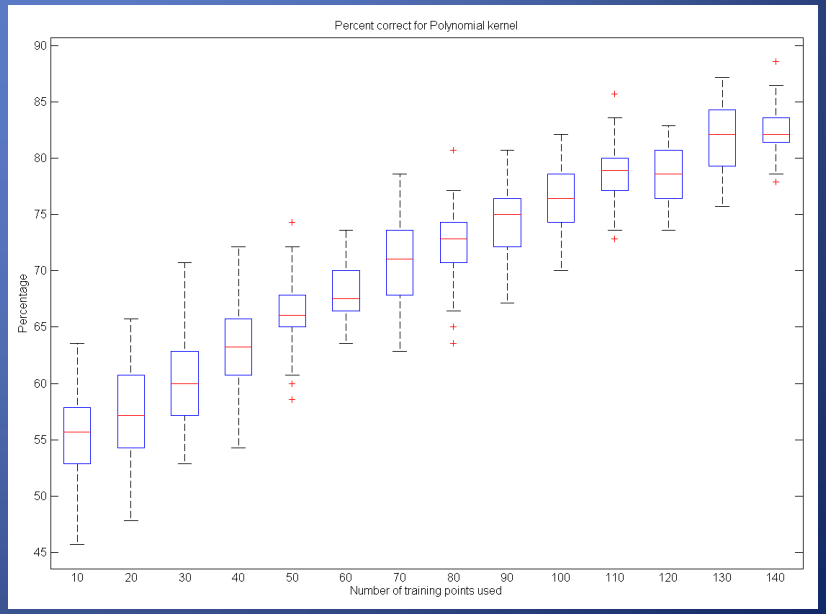
The Polynomial and the Gaussian Radial Basis function appear to predict the training data the best.

"Perfect" conditions	
Kernel	Percent Correct
Linear	67.86%
Quadratic	71.43%
Polynomial	100.00%
Gaussian Radial Basis function	100.00%

RBF

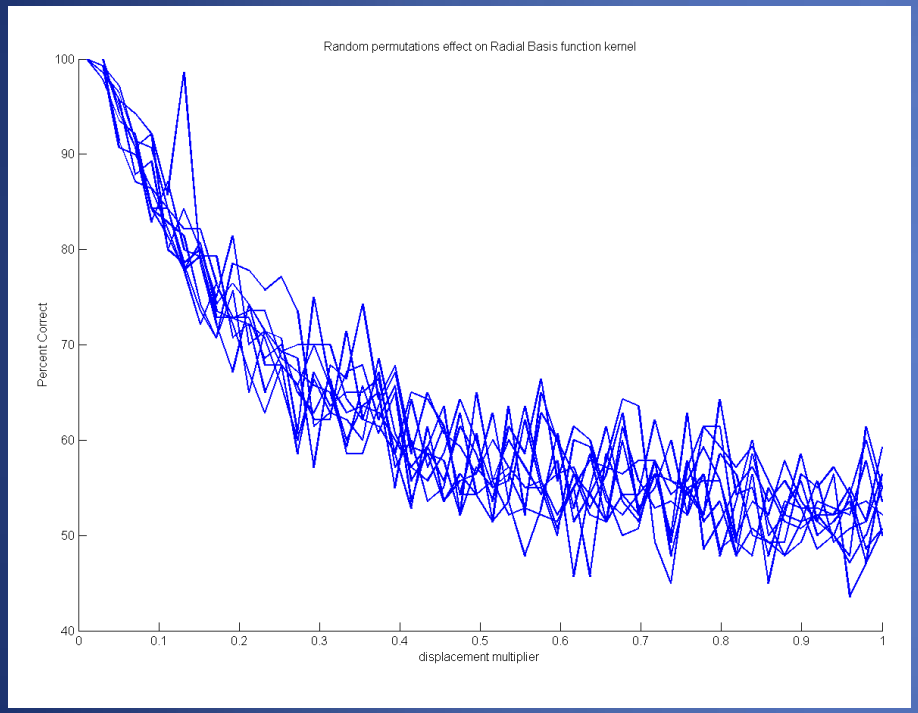


Polynomial

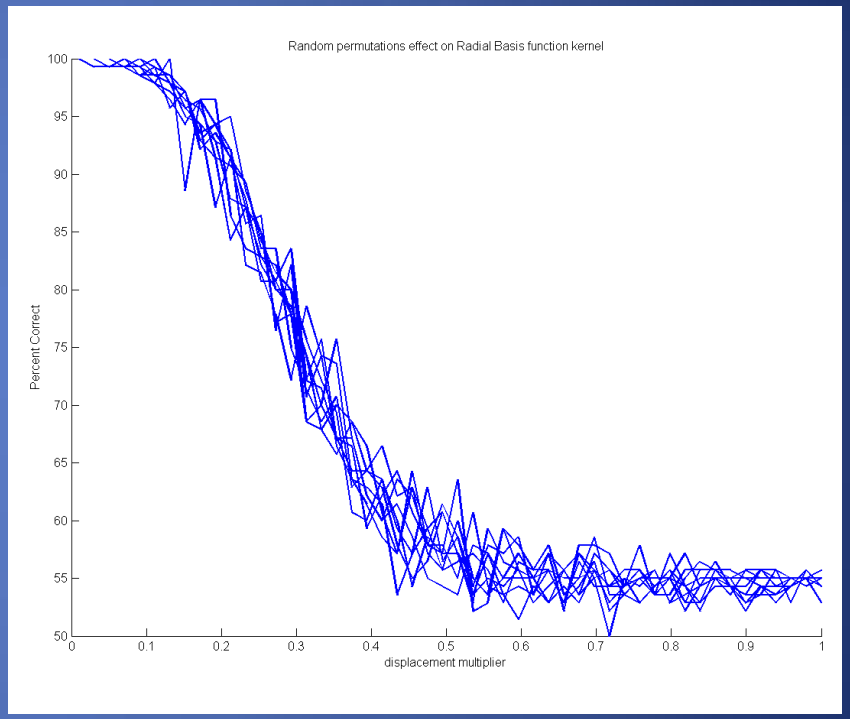


Permute the initial training data to determine the sensitivity of our SVM

Polynomial



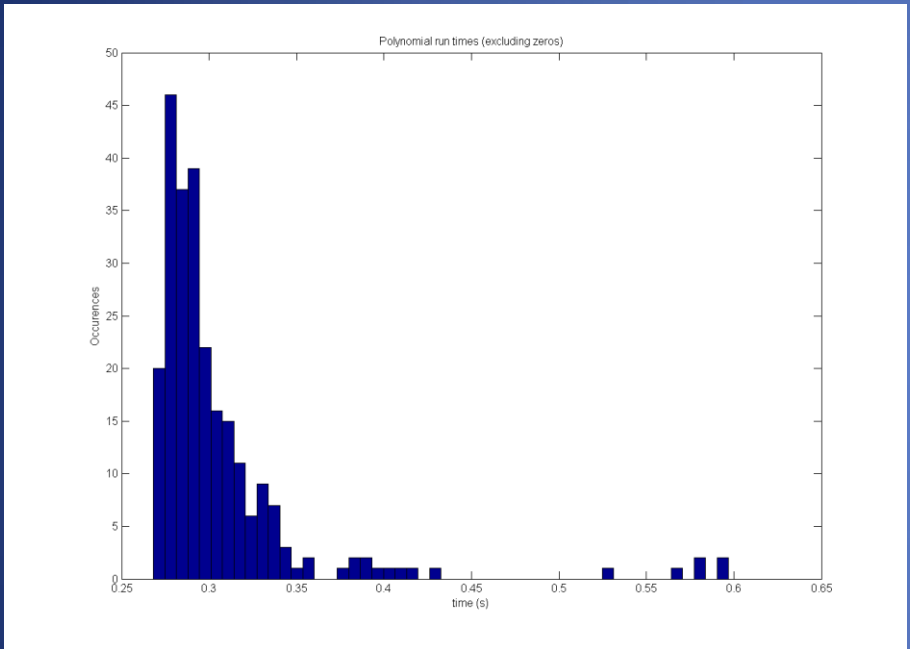
Radial Basis Function



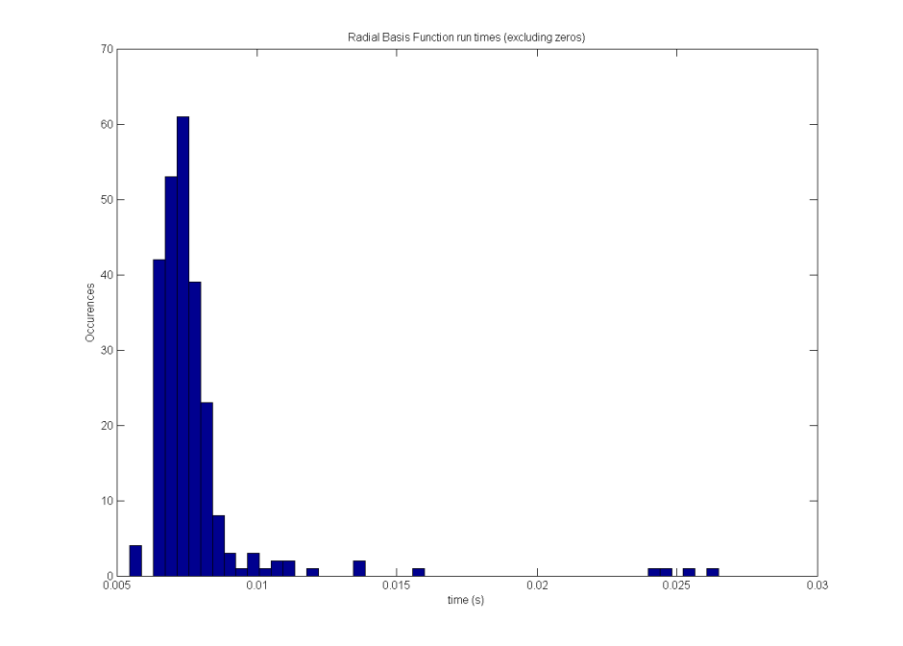
Appears that the RBF kernel is more sensitive up to a point but then quickly drops off!

What about the speed of each of the different kernels?

Polynomial



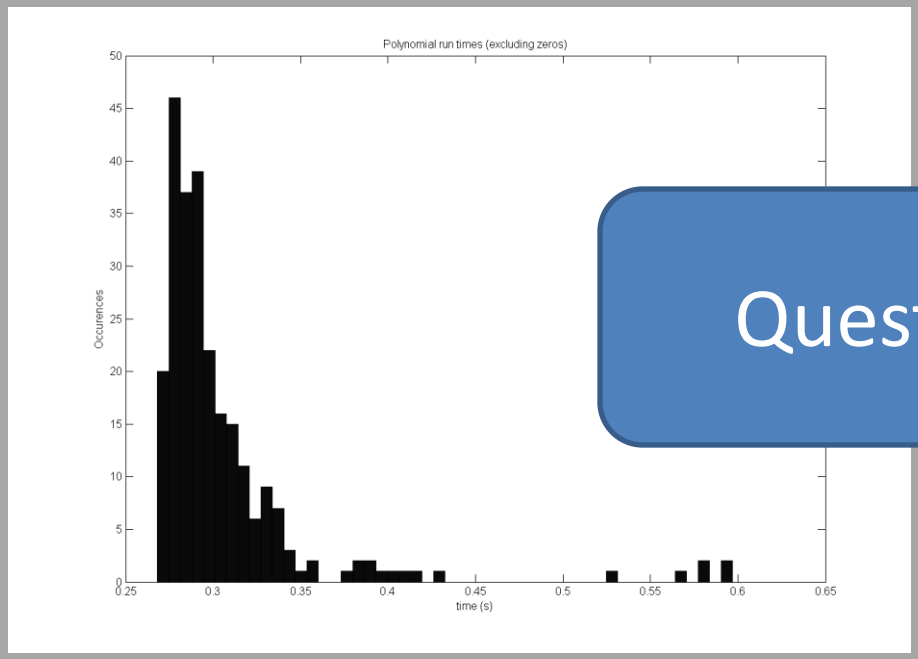
Radial Basis Function



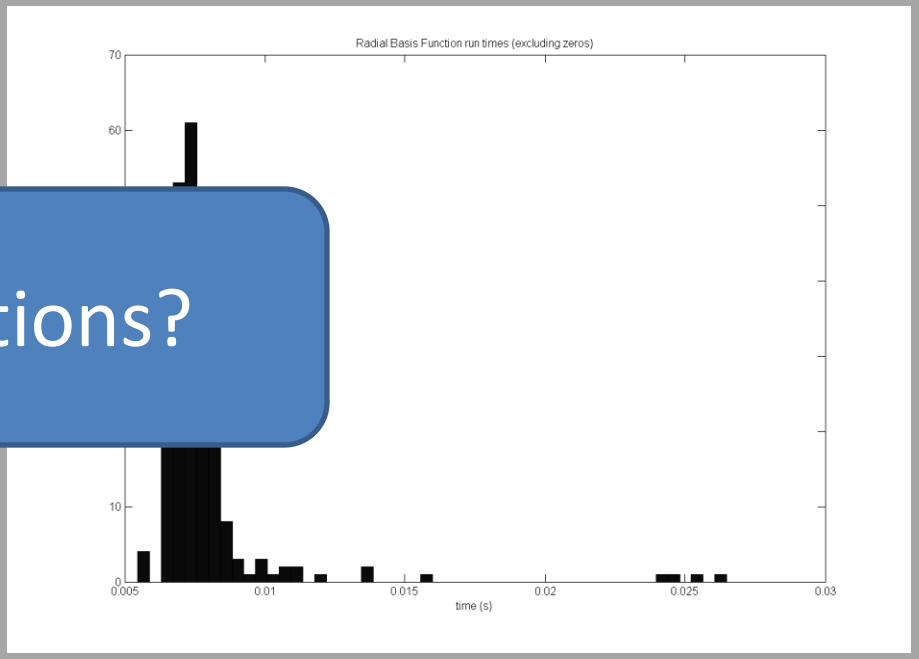
The RBF kernel is faster by over an order of magnitude!

What about the speed of each of the different kernels?

Polynomial



Radial Basis Function



Questions?

The RBF kernel is faster by over an order of magnitude!