Pattern Recognition and Image Analysis

Non-Linear Classifiers 3:

SVM

Dr. Manal Helal – Fall 2015

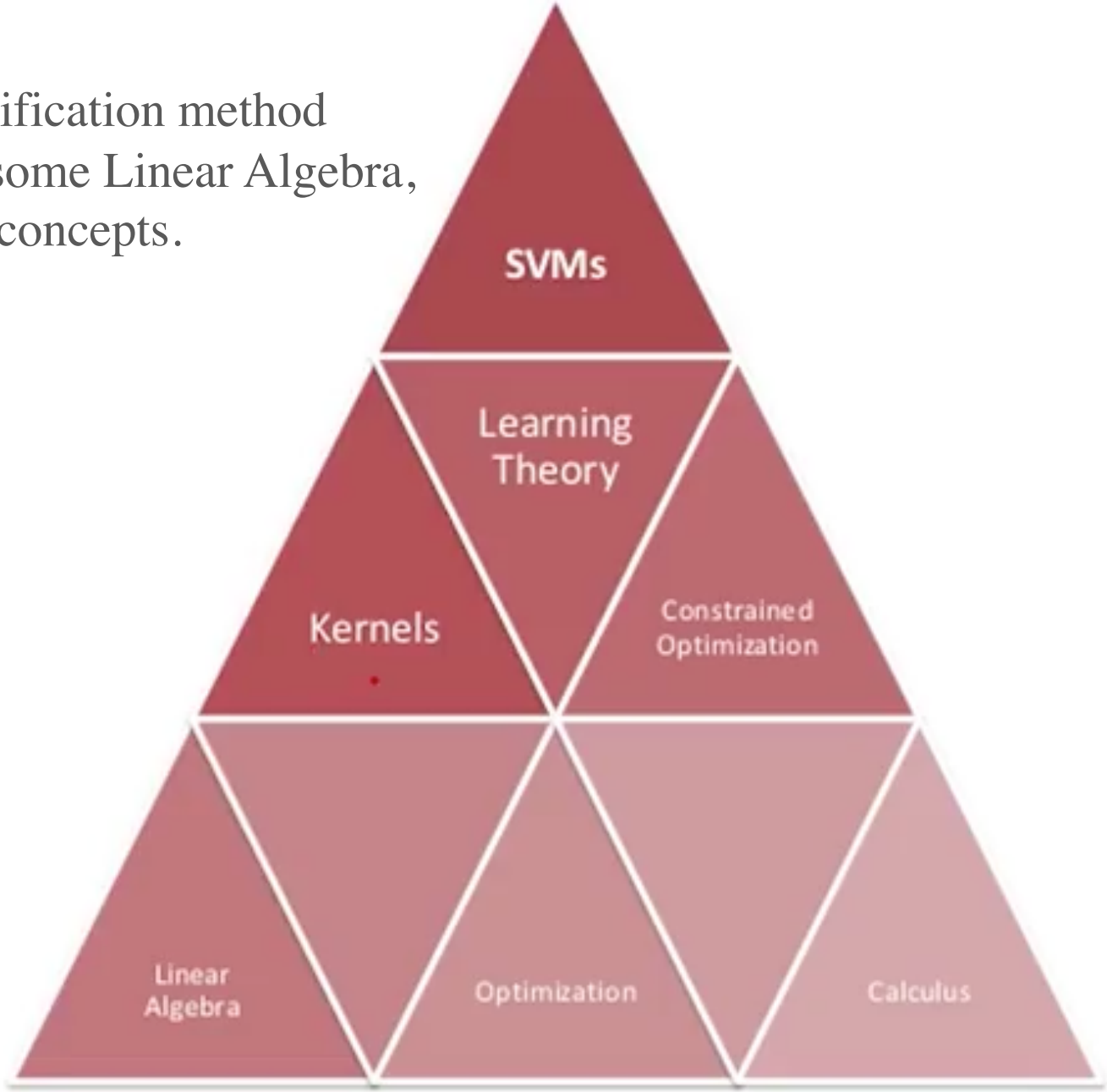
Lecture 11

Overview

- Kernel Based Methods
- RBFs
- Non Linear SVM
- Code Examples

SVM Pre-requisites

- SVM is a kernel based classification method (learning), that depends on some Linear Algebra, Optimisation, and Calculus concepts.



Popular Kernels

Most off-the-shelf classifiers allow the user to specify one of three popular kernels: the **polynomial**, **radial basis function**, and **sigmoid** kernel. You can even pass in a custom kernel.

For the following, let $\vec{x_i}, \vec{x_j} \in \mathbb{R}^N$ be rows from the dataset X.

1. **Polynomial Kernel**: $(\gamma \cdot \langle \vec{x_i}, \vec{x_j} \rangle + r)^d$
2. **Radial Basis Function (RBF) Kernel**: , $exp(-\gamma \cdot |\vec{x_i} - \vec{x_j}|^2)$ where $\gamma > 0$
3. **Sigmoid Kernel**: $tanh(\langle \vec{x_i}, \vec{x_j} \rangle + r)$

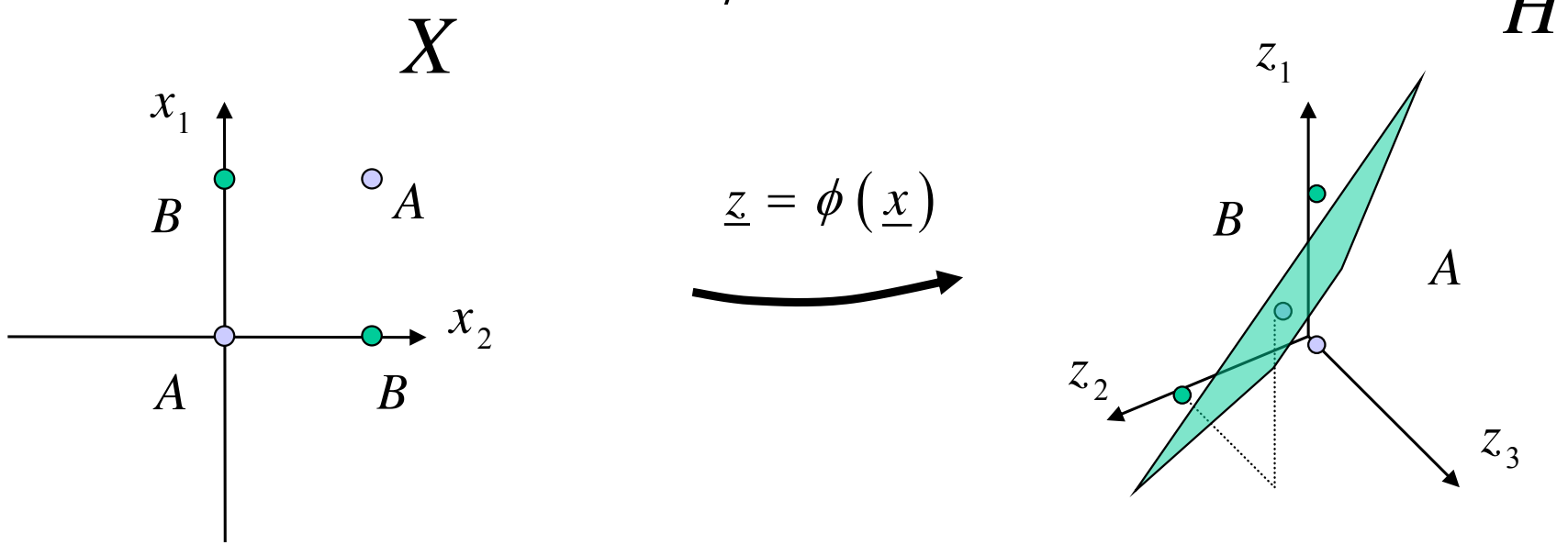
Unfortunately, choosing the 'correct' kernel is a nontrivial task, and may depend on the specific task at hand. No matter which kernel you choose, you will need to tune the kernel parameters to get good performance from your classifier. Popular parameter-tuning techniques include K-Fold Cross Validation:

https://en.wikipedia.org/wiki/Cross-validation_%28statistics%29#K-fold_cross-validation

Polynomial Classifier: XOR problem

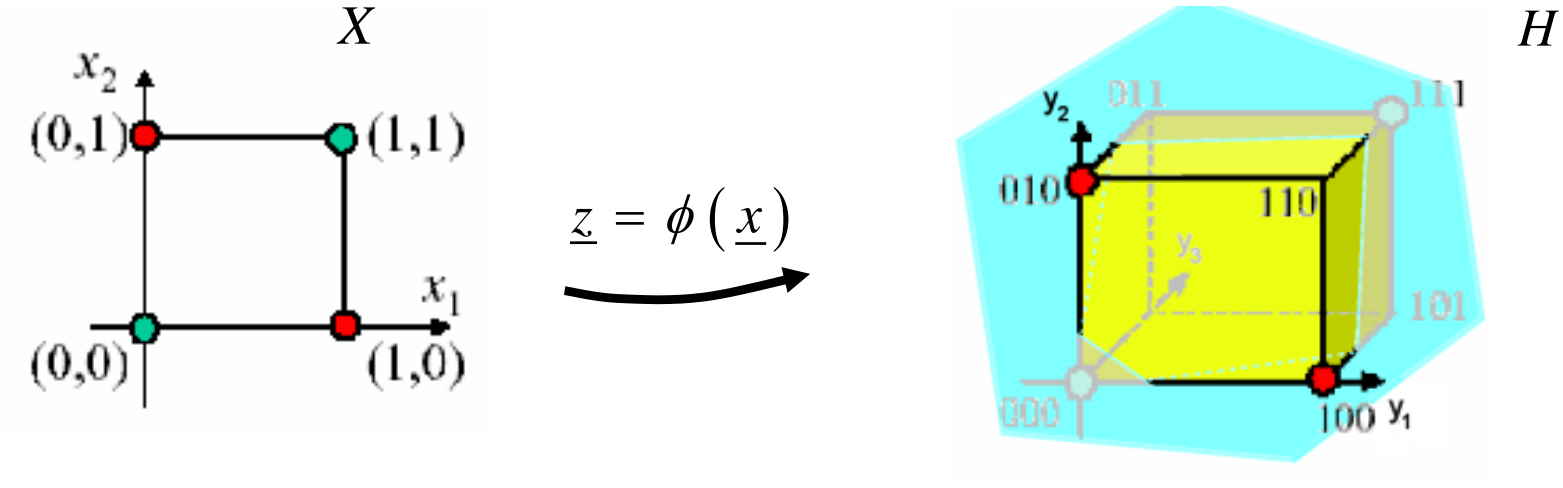
- XOR problem with polynomial function.
  - With nonlinear polynomial function classes can be classified.
  - Example XOR-Problem:
    - linear not separable!

$$\phi : X \rightarrow H$$



...but with a polynomial function!

Polynomial Classifier: XOR problem



With $\underline{z} = \begin{bmatrix} x_1 \\ x_2 \\ x_1 x_2 \end{bmatrix}$ we obtain:

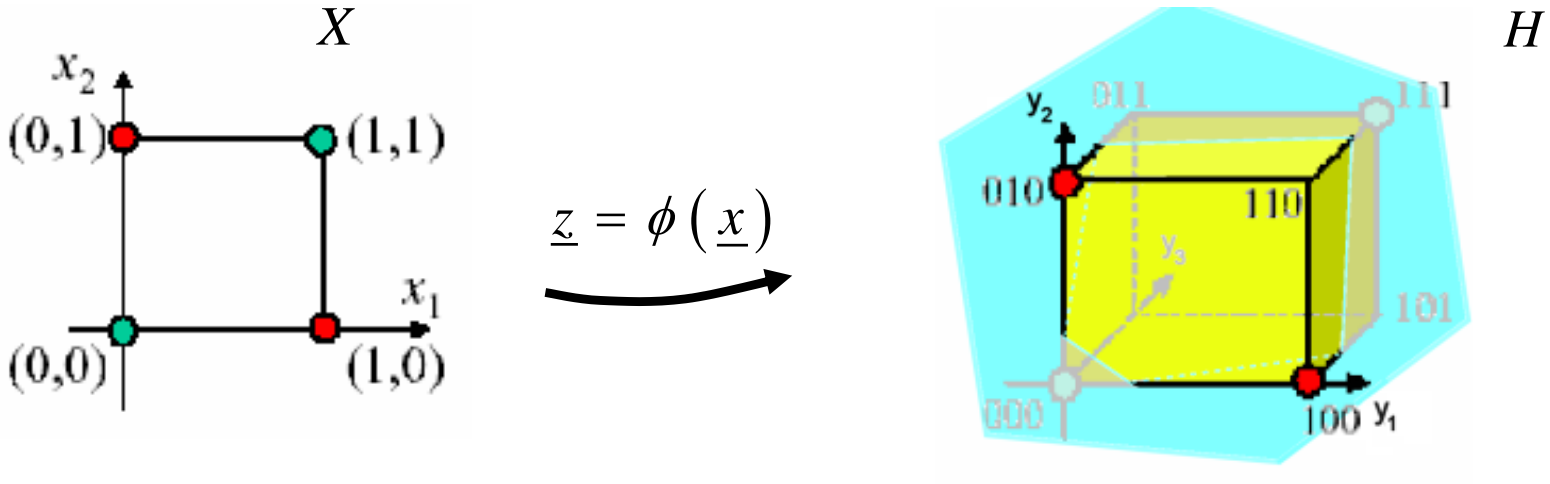
ϕ (0,0) → (0,0,0)
ϕ (0,1) → (0,1,0)
ϕ (1,0) → (1,0,0)
ϕ (1,1) → (1,1,1)

… that‘s separable in H

by the Hyperplane:

$$g(\underline{z}) = \frac{1}{4} - 1z_1 - 1z_2 + 2z_3 = 0$$

Polynomial Classifier: XOR problem



Hyperplane: $g(\underline{y}) = \underline{w}\,\underline{y} + w_0 = 0$

$$g(\underline{z}) = \frac{1}{4} - z_1 - z_2 + 2z_3 = 0$$

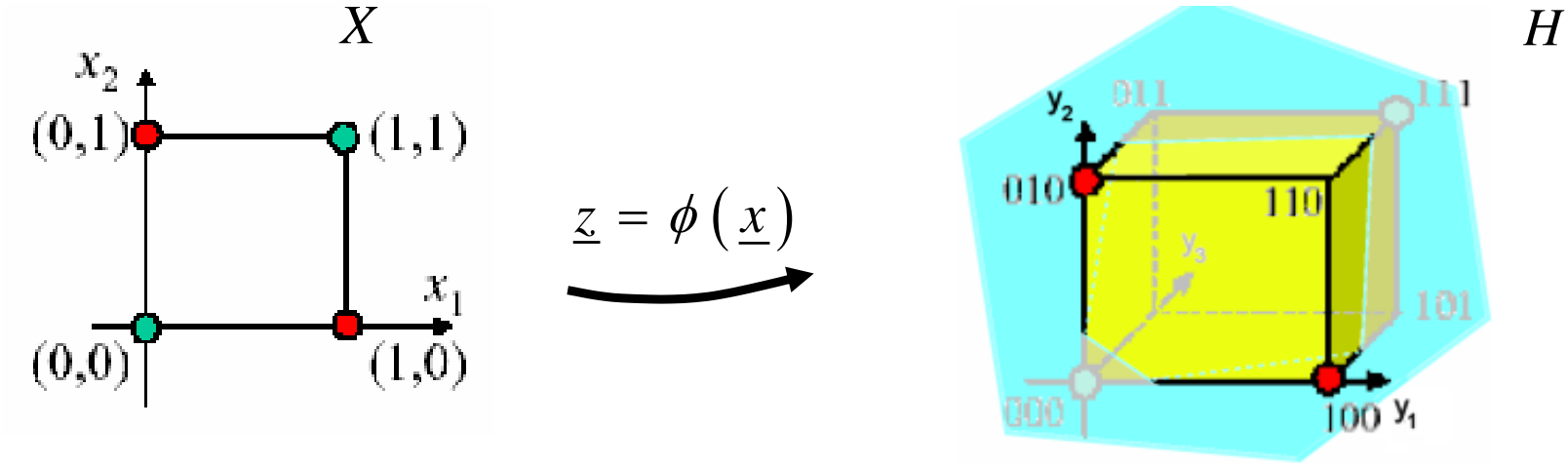
is Hyperplane in H

$$g(\underline{x}) = \frac{1}{4} - x_1 - x_2 + 2x_1x_2$$

is Polynom in X

X		H			
		z_1	z_2	z_3	
x_1	x_2	x_1	x_2	x_1x_2	$\underline{\omega}$
0	0	0	0	0	A (true)
0	1	0	1	0	B (false)
1	0	1	0	0	B (false)
1	1	1	1	1	A (true)

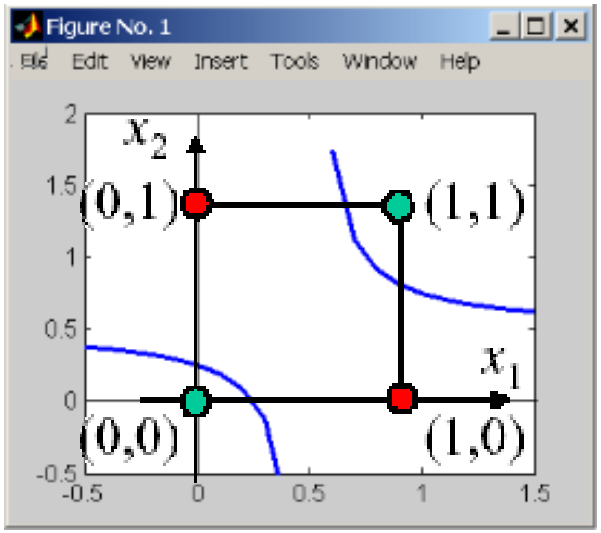
Polynomial Classifier: XOR problem



Decision Surface in X

$$g(\underline{x}) = \frac{1}{4} - 1x_1 - 1x_2 + 2x_1x_2 \quad \begin{matrix} \geq 0 & \underline{x} \in A \\ < 0 & \underline{x} \in B \end{matrix}$$

$$x_2 = (x_1 - 0.25)/(2x_1 - 1)$$



MatLab:

```
>> x1=[-0.5:0.1:1.5];
>> x2=(x1-0.25)./(2*x1-1);
>> plot(x1,x2);
```

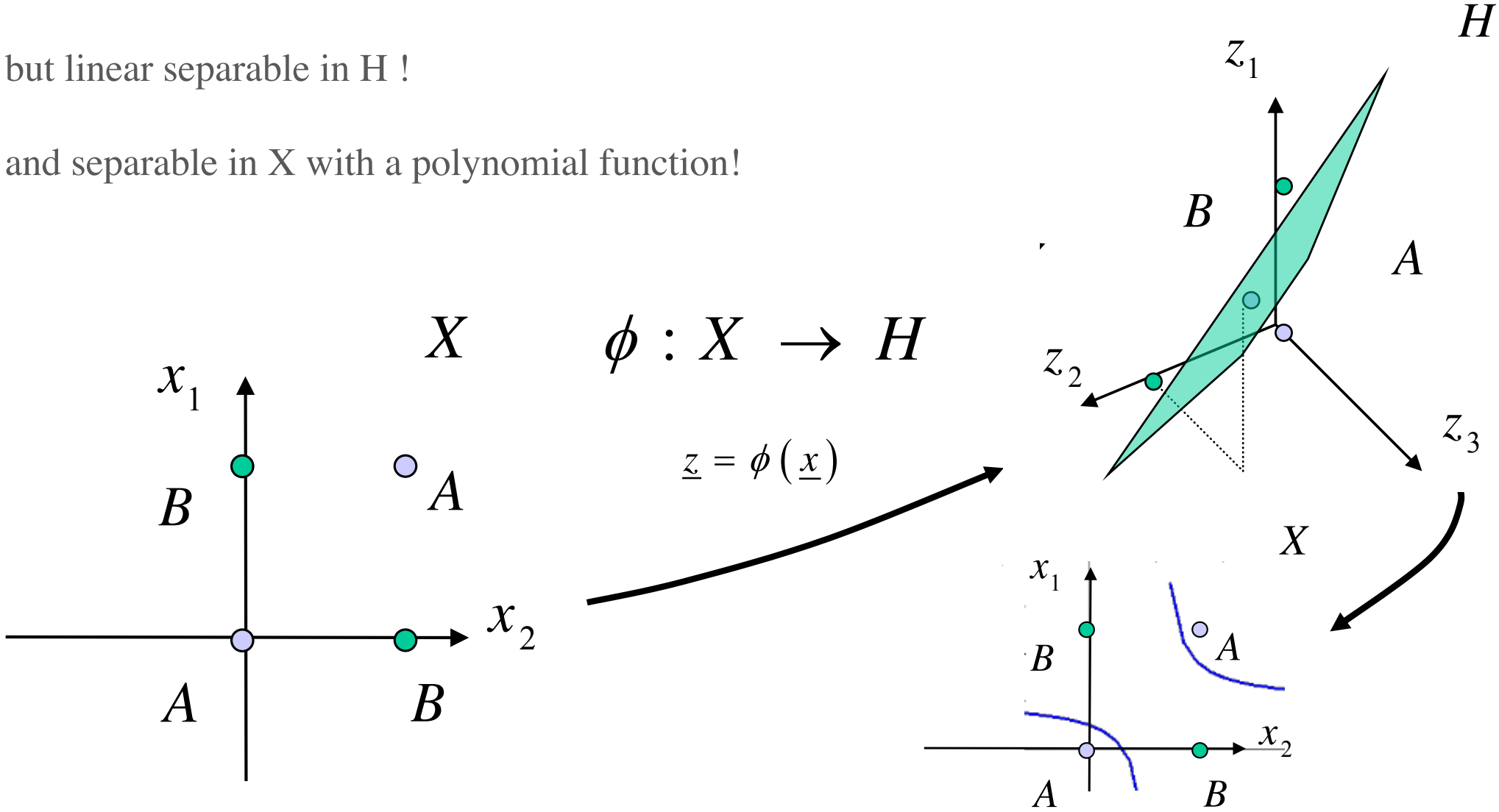
Polynomial Classifier: XOR problem

- With nonlinear polynomial functions, classes can be classified in original space X

- Example: XOR-Problem was not linearly separable!

... but linear separable in H !

... and separable in X with a polynomial function!



Polynomial Classifier

- Decision function is approximated by a polynomial function g(x) , of order p e.g. p=2:

$$g(\underline{x}) = w_0 + \sum_{i=1}^{l} w_i x_i + \sum_{i=1}^{l-1} \sum_{m=i+1}^{l} w_{im} x_i x_m + \sum_{i=1}^{l} w_{ii} x_i^2$$

$$g(\underline{x}) = \underline{w}^T \underline{z} + w_0,$$

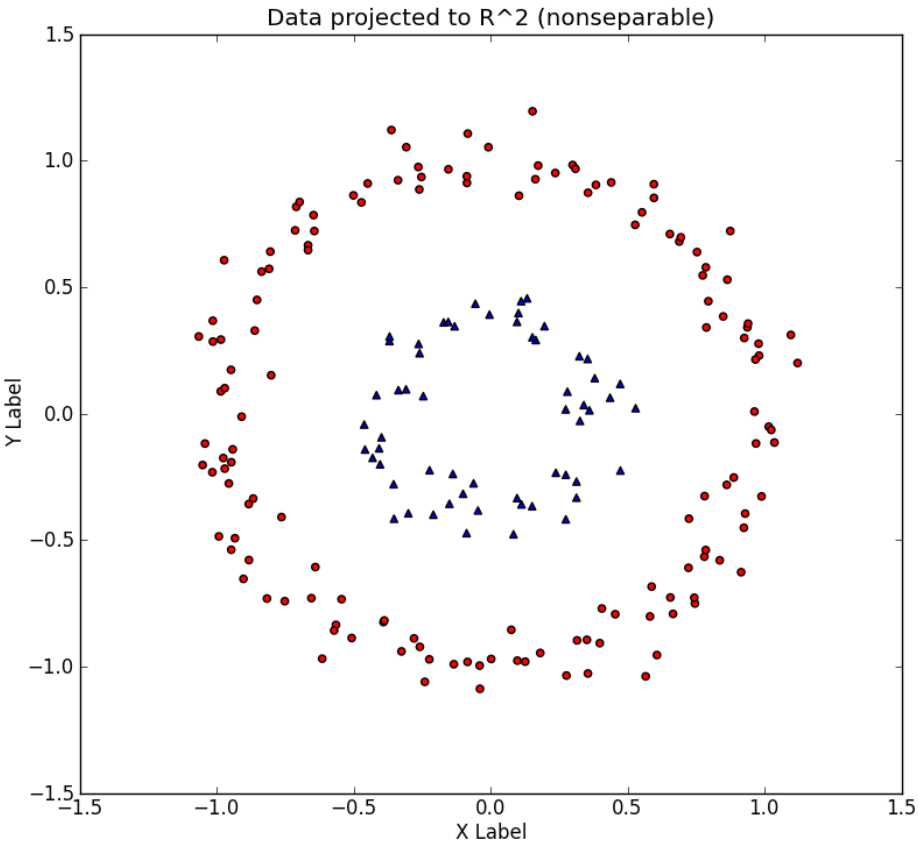
with

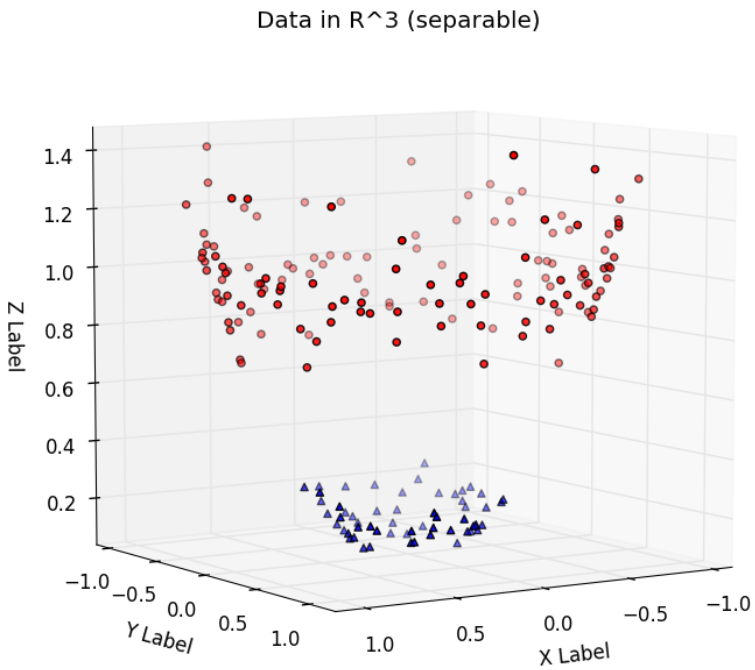
$$\underline{w}^T = [w_1, w_2, w_{12}, w_{11}, w_{22}],$$

$$\underline{z} = \left[x_1, x_2, x_1 x_2, x_1^2, x_2^2\right]^T \quad \text{and} \quad \underline{x} = [x_1, x_2]^T$$

- Special case of a Two-Layer Perceptron
- Activation function with polynomial input

Similarly Map $\mathbb{R}^2 \rightarrow \mathbb{R}^3$





(Left) A dataset in $\mathbb{R}^2$, not linearly separable. (Right) The same dataset transformed to $\mathbb{R}^3$ by the transformation:

$$[x_1, x_2] \rightarrow [x_1, x_2, x_1^2 + x_2^2]$$

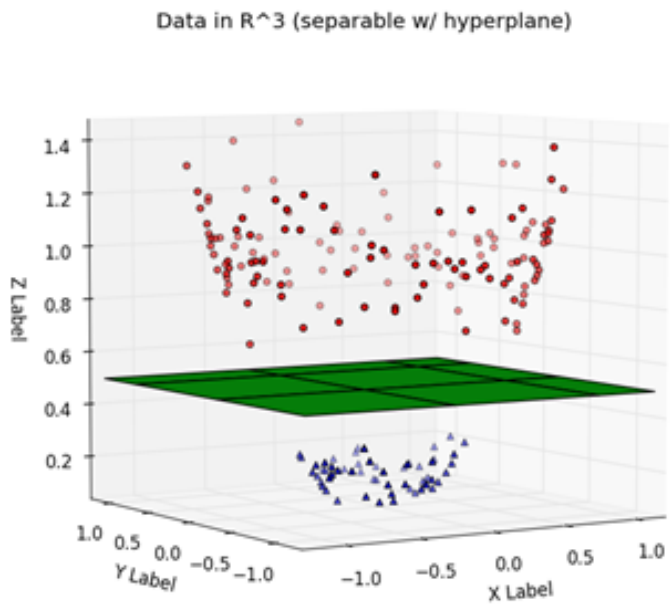
Non-Linear SVM in Higher dimensions

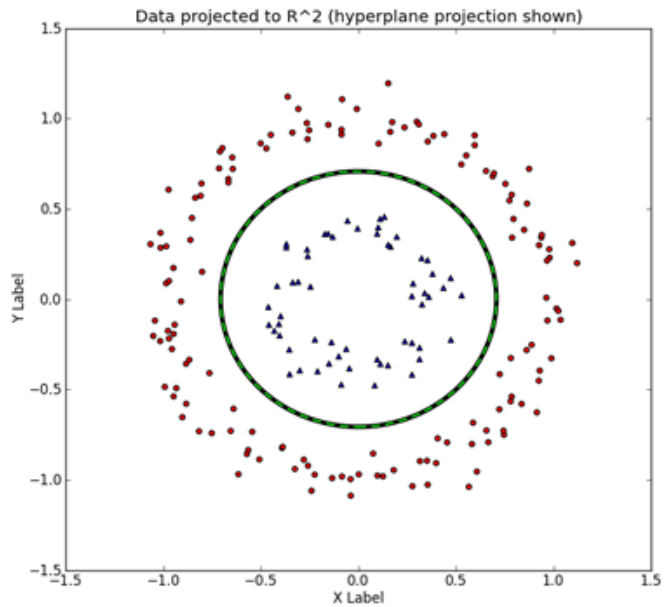
- Given a dataset that is not linearly separable in $\mathbb{R}^N$ may be linearly separable in a higher-dimensional space $\mathbb{R}^M$ (where M > N).

- Thus, if we have a transformation that lifts the dataset to a higher-dimensional <u>such that</u> is linearly separable, then we can train a linear SVM on to find a decision boundary that separates the classes in $\mathbb{R}^M$.

- Projecting the decision boundary found in back to the original space will yield a nonlinear decision boundary.

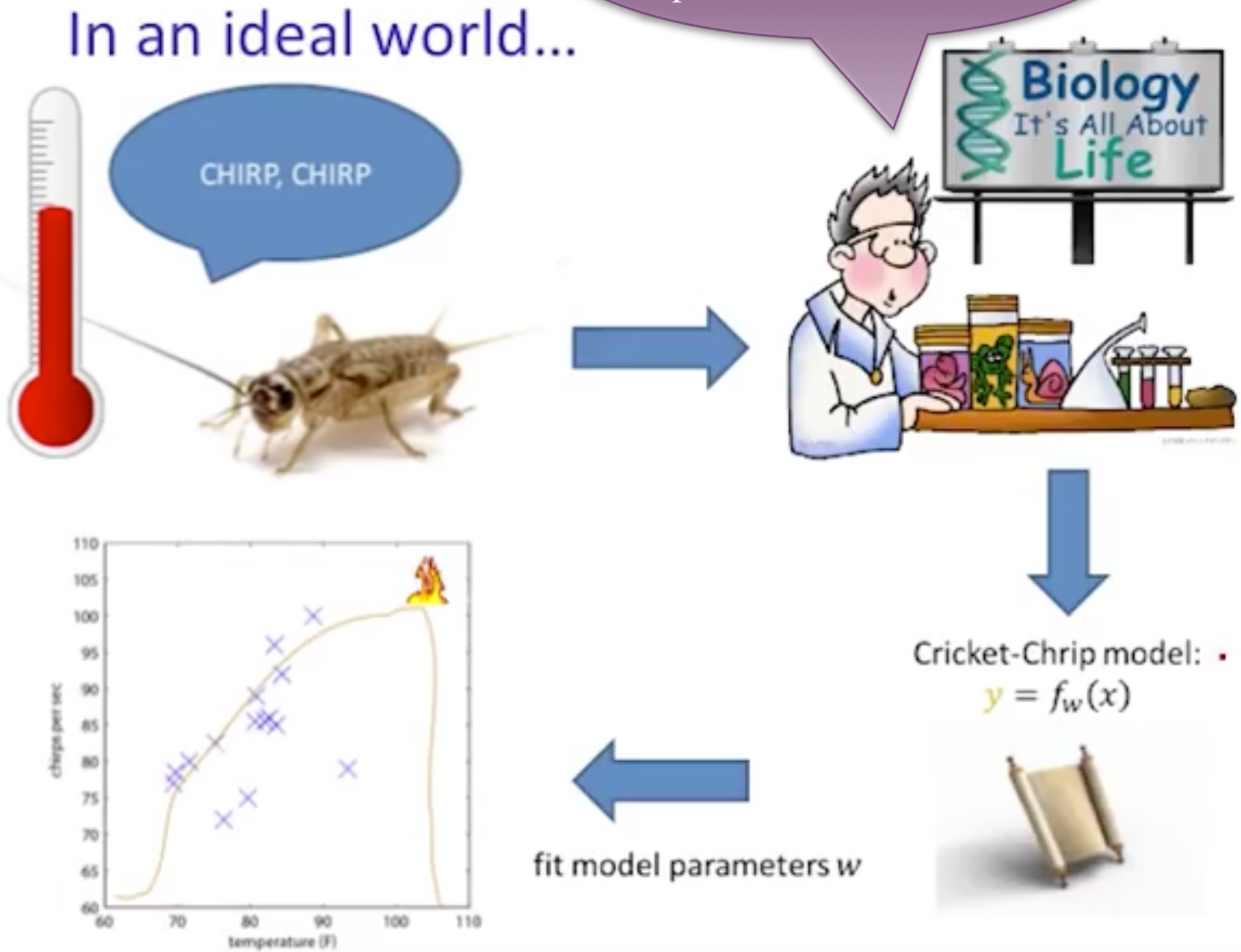
- Visualisations:
  - "SVM with polynomial kernel visualization"
  - "Performing nonlinear classification via linear separation in higher dimensional space"



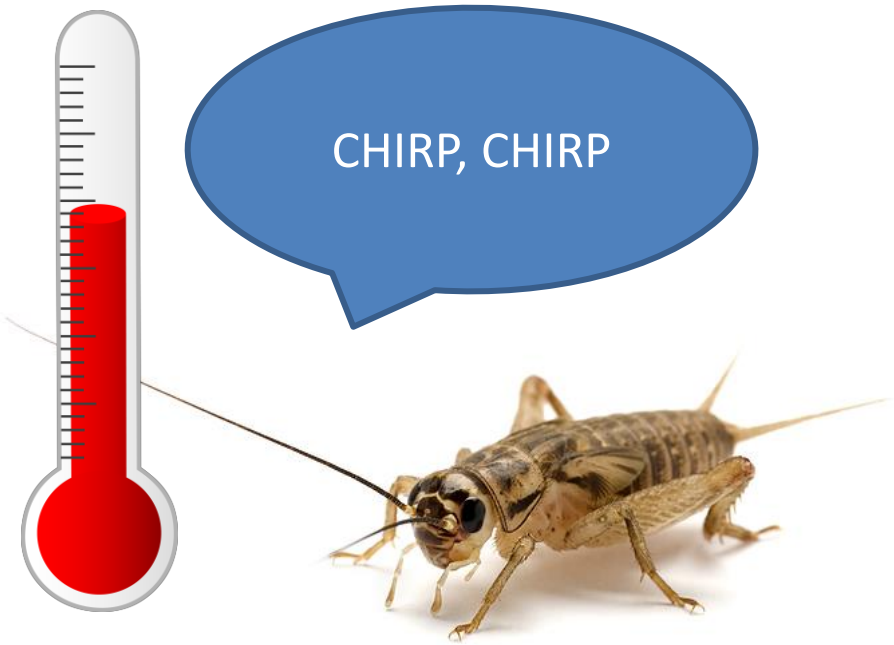


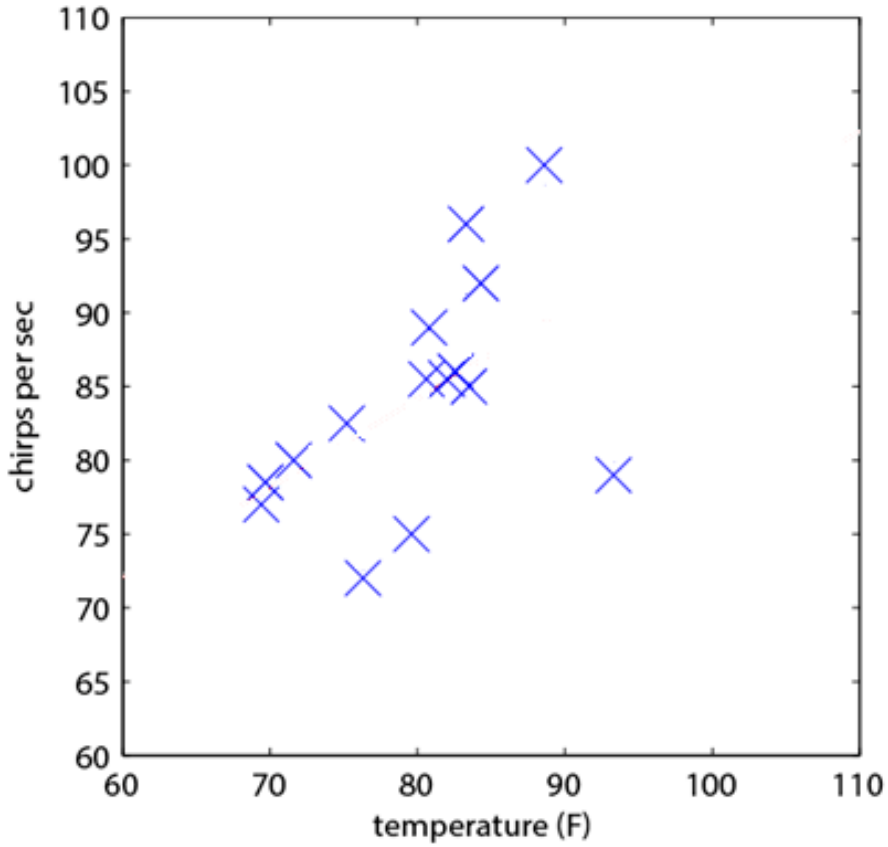
Example

We don't always have domain experts, and need to predict the model



Crickets





Chirps per second versus temperature.

More Complex Basis functions

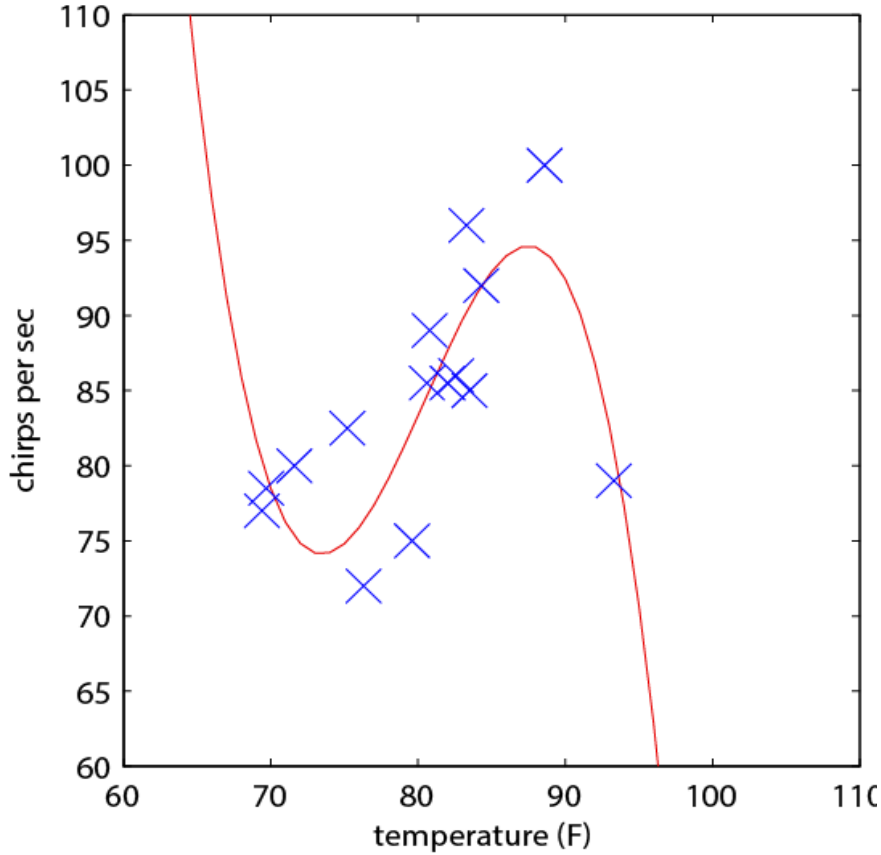
- We can also use more complex mode, for example a 3rd degree polynomial :
- $y(x) = w_1 + w_2x + w_3x^2 + w_4x^3$.
- This yields the feature map:

$$\varphi(x) = \begin{pmatrix} 1 \\ x \\ x^2 \\ x^3 \end{pmatrix}$$

Quality of Solution? Worse

Model Complexity? Higher

Why? Model does not match data



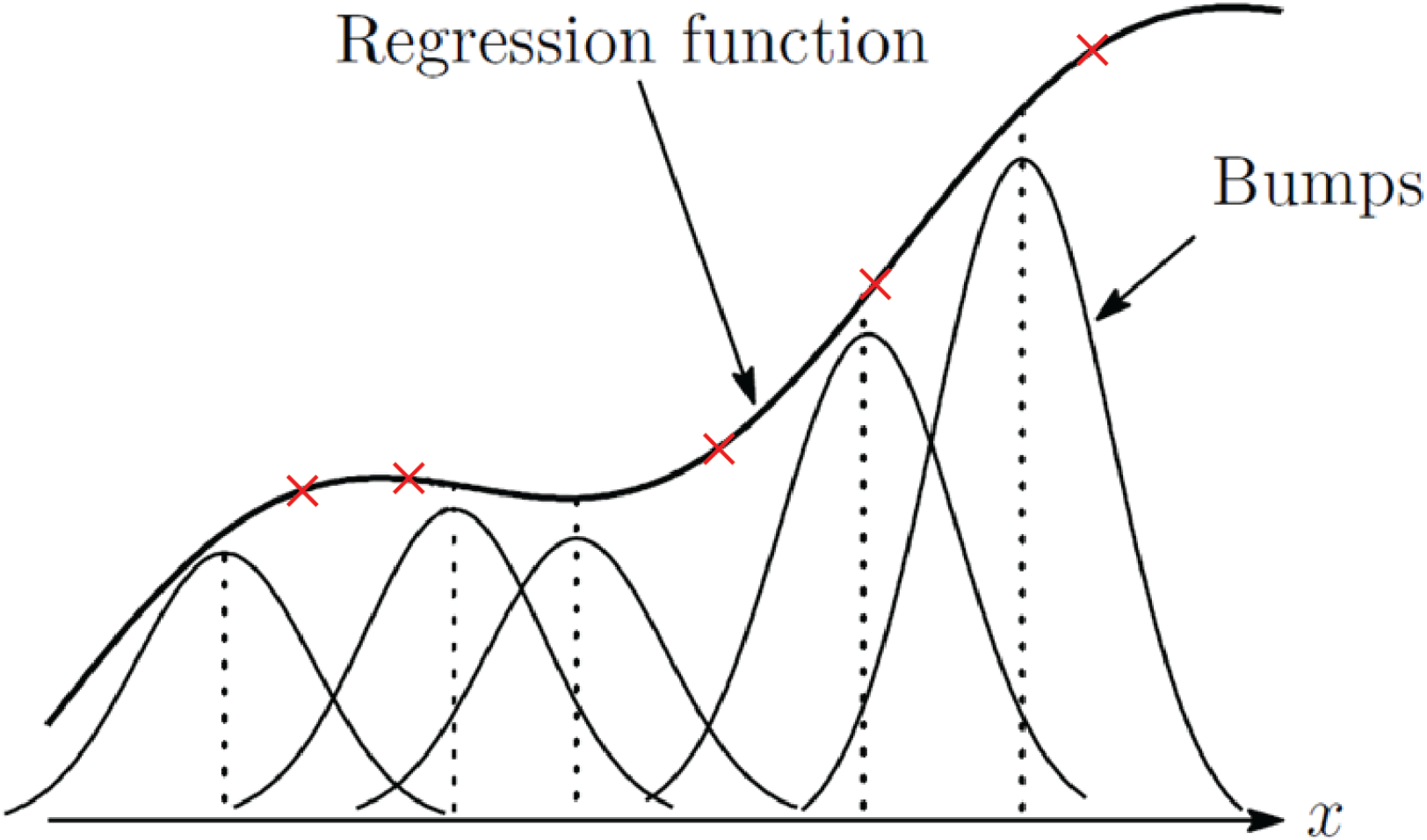
Chirps per second versus temperature.

Dealing with unknown Basis Functions

- One way to deal with over-fitting is by using regularisation.
- However, it will not help much if the basis functions do not match the data.
- So, what we should do if we do not know the model?
- Is there a universal set of basis functions that can approximate arbitrary functions reasonably well?

Linear Regression Using "bump" Functions

- Approximate the function with a sum of bump shaped functions



H. Takeda. Kernel Regression for Image Processing and Reconstruction. 2006.

Radial Basis Functions (RBFs)

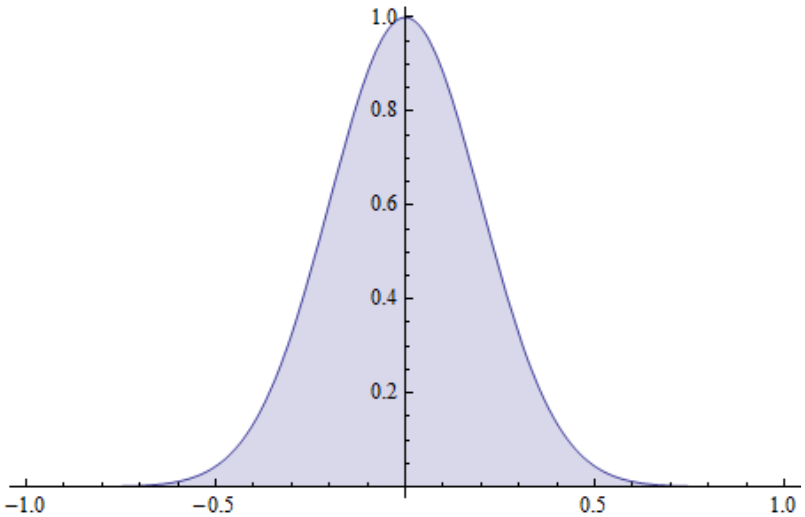
RBF a real-valued function whose value depends only on the distance from the origin, so that $\phi(\mathbf{x}) = \phi(\|\mathbf{x}\|)$; or alternatively on the distance from some other point *c*, called a *center*, so that $\phi(\mathbf{x}, \mathbf{c}) = \phi(\|\mathbf{x} - \mathbf{c}\|)$ Any function ϕ that satisfies the property $\phi(\mathbf{x}) = \phi(\|\mathbf{x}\|)$ is a radial function. The norm is usually Euclidean distance, although other distance functions are also possible.

- Gaussian Radial Basis Functions :

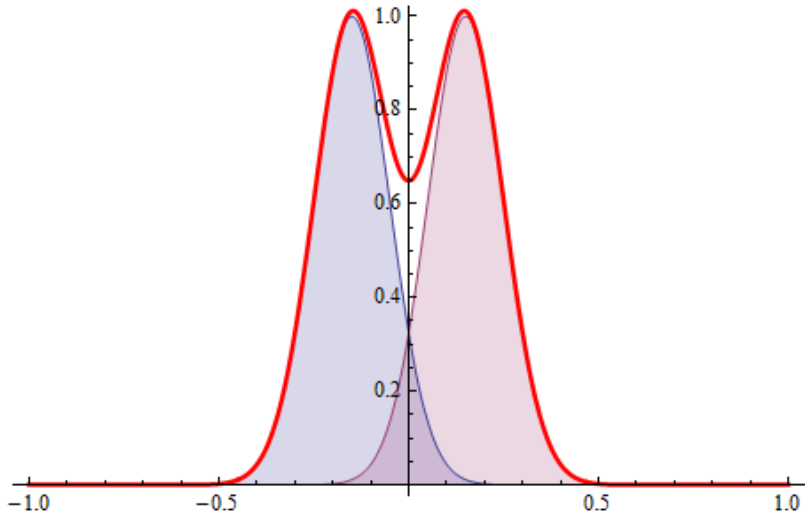
$$\phi_i(\boldsymbol{x}) = \exp\left(-\frac{\left(\boldsymbol{x} - \boldsymbol{m}^{(i)}\right)^2}{2\alpha^2}\right)$$

The centre is at $m^{(i)}$ and α determines the width of the bump.

- Advantage: They decrease smoothly to zero and do not show oscillatory behaviour unlike the higher order polynomials.



$\alpha = 0.2, m = 0$



$\alpha = 0.1, m = -0.15; m = +0.15$

Non-linear Support Vector Machines

- Recall that, the probability of having linearly separable classes increases as the dimensionality of feature vectors increases.

- Assume the mapping:

  $x \in R^l \longrightarrow z \in R^k, \qquad k > l$

- Then use linear SVM in R^k

- Recall that in this case the dual problem formulation will be :

$$\max_{\underline{\lambda}} \left(\sum_{i=1}^{N} \lambda_i - \frac{1}{2}\sum_{i,j} \lambda_i \lambda_j y_i y_j \underline{z}_i^T \underline{z}_j\right)$$

$$\text{where } \underline{z}_i \in R^k,\ y \in \{-1,1\} \text{ (class labels)}$$

- The classifier will be:

$$g(\underline{z}) = \underline{w}^T \underline{z} + w_0$$

$$= \sum_{i=1}^{N_s} \lambda_i y_i \underline{z}_i^T \underline{z} + w_0$$

Non-linear SVM

- Thus, only inner products in a high dimensional space are needed!

- Something clever (kernel trick): Compute the inner products in the high dimensional space as functions of inner products performed in the low dimensional space!!!

Non-Linear SVM

- Is this POSSIBLE?? Yes. Here is an example
- Let $x=[x_1,x_2]^T \in R^2$

$$\text{Let} \quad \underline{x} \rightarrow \underline{z} = \begin{bmatrix} x_1^2 \\ \sqrt{2}\,x_1 x_2 \\ x_2^2 \end{bmatrix} \in R^3$$

It is easy to show that $\boxed{\underline{z}_i^T \underline{z}_j = (\underline{x}_i^T \underline{x}_j)^2}$

$$(\underline{x}_i^T \underline{x}_j)^2 = \left(x_{i1}x_{j1} + x_{i2}x_{j2}\right)^2$$

$$= x_{i1}^2 x_{j1}^2 + 2\,x_{i1}x_{j1}x_{i2}x_{j2} + x_{i2}^2 x_{j2}^2$$

$$= \left(x_{i1}^2, \sqrt{2}\,x_{i1}x_{i2}, x_{i2}^2\right) \begin{pmatrix} x_{j1}^2 \\ \sqrt{2}\,x_{j1}x_{j2} \\ x_{j2}^2 \end{pmatrix} = \underline{z}_i^T \underline{z}_j$$

What about if M is too large?

- The computational consequences of increasing the dimensionality of R^n to R^m, (with $M > N$), if M grows very quickly with respect to ($M \in O(2^N)$), then learning SVMs via dataset transformations, will incur serious computational and memory problems.

- Example Polynomial Kernel from R2 to R5:

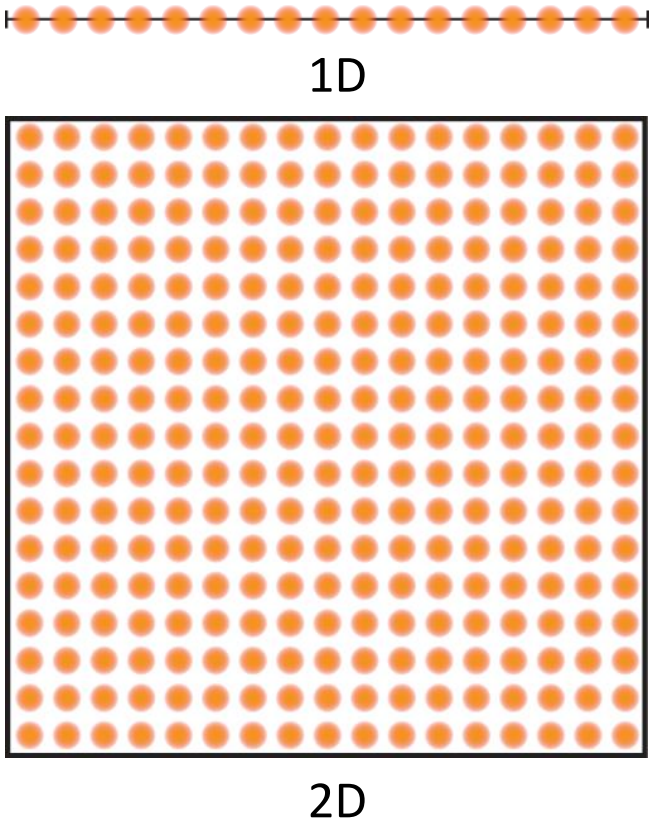
$[x_1, x_2] = [x_1^2,\ \ x_2^2,\ \sqrt{2}.x_1.\ x_2\ ,\ \sqrt{2}.c.x_1\ , \sqrt{2}.c.x_2.c]$

- A d-dimensional polynomial kernel maps from Rn to an $\binom{N+d}{d}$-dimensional space

- Thus for datasets with large dimensionality, naively performing such transformation will quickly become a performance problem.

Curse of Dimensionality

- To cover the data with a constant discretisation level (number of RBFs per unit volume) the number of basis functions and weights will grows exponentially with the number of dimensions:



Dimensions	# of basis functions / weights
1	16
2	$16^2 = 256$
3	$16^3 = 4096$
4	$16^4 = 65536$
5	$16^5 = 1048576$
:	:
10	$16^{10} \approx 10^{12}$

- Is there a way to reduce the number of required weights?

The Kernel Trick (We only need dot products)

- Give me the numeric result of $(x + y)^{10}$ for some values of x and y.
- By The Binomial Theorem:

$$x^{10}+10x^9y+45x^8y^2+120x^7y^3+210x^6y^4+252x^5y^5+210x^4y^6+120x^3y^7+45x^2y^8+10xy^9+y^{10}$$

- The trick: we can evaluate the dot product between degree 10 polynomials without explicitly forming them. Valid kernels are dot products where we can compute the numeric result between two points without having to form their explicit feature values.
- $k(x, y) = \langle\varphi(x), \varphi(y)\rangle$ where $\langle . , . \rangle$ is the inner product in the new space. We don't even need to know what φ is

The "Kernel Trick"

- **The linear classifier relies on dot product between vectors $K(x_i,x_j)=x_i^T x_j$**
- **If every data point is mapped into high-dimensional space via some transformation Φ: $x \rightarrow \varphi(x)$, the dot product becomes:**

$$K(x_i,x_j)= \varphi(x_i)^T \varphi(x_j)$$

- **A *kernel function* is some function that corresponds to an inner product in some expanded feature space.**
- **Example:**

2-dimensional vectors $x=[x_1 \;\; x_2]$; let $K(x_i,x_j)=(1 + x_i^T x_j)^2$,

Need to show that $K(x_i,x_j)= \varphi(x_i)^T \varphi(x_j)$:

$$K(x_i,x_j)=(1 + x_i^T x_j)^2,$$

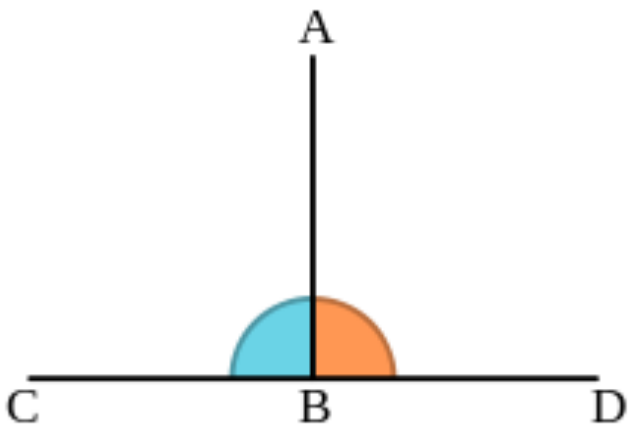
$$= 1+ x_{i1}^2 x_{j1}^2 + 2\, x_{i1}x_{j1}\, x_{i2}x_{j2} + x_{i2}^2 x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}$$

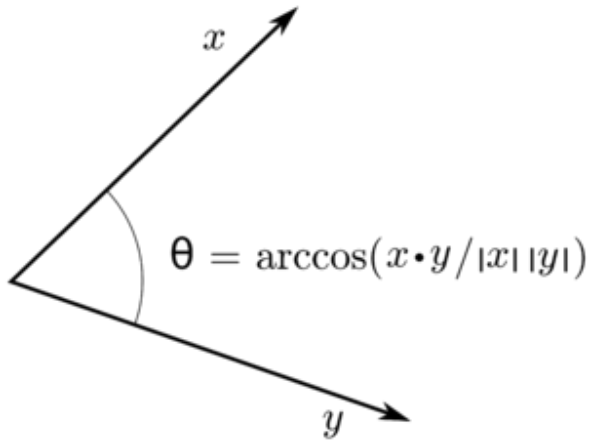
$$= [1 \;\; x_{i1}^2 \;\; \sqrt{2}\, x_{i1}x_{i2} \;\; x_{i2}^2 \;\; \sqrt{2}x_{i1} \;\; \sqrt{2}x_{i2}]^T\, [1 \;\; x_{j1}^2 \;\; \sqrt{2}\, x_{j1}x_{j2} \;\; x_{j2}^2 \;\; \sqrt{2}x_{j1} \;\; \sqrt{2}x_{j2}]$$

$$= \varphi(x_i)^T \varphi(x_j), \quad \text{where } \varphi(x) = [1 \;\; x_1^2 \;\; \sqrt{2}\, x_1x_2 \;\; x_2^2 \;\; \sqrt{2}x_1 \;\; \sqrt{2}x_2]$$

Orthogonality explains the trick

- It is the relation of two lines at right angles to one another (perpendicularity).
- It describes non-overlapping, uncorrelated, or independent objects of some kind.
- The dot product of two vectors measures the angle between them. Two vectors are orthogonal, when their dot product is equal to zero. (arccos(0) = 90°).





Non-Linear SVM

- Mercer's Theorem

$$\text{Let } x \longrightarrow \varnothing(x) \in H$$

- To guarantee that the symmetric function $K(x_i, x_j)$ (kernel) can be represented as

$$\sum_r \varnothing_r(x_i)\varnothing_r(x_j) = K(x_i, x_j)$$

- that is an inner product in H, it is necessary and sufficient that

$\int K(x_i, x_j)\, g(x_i) g(x_j)\, dx_i\, dx_j \geq 0$ (1) *K* is *non-negative definite (Positive semidefinite)*

for any $g(x)$: $\int g^2(x)\, dx < +\infty$ (2) square-integrable real-valued function

Non Linear SVM

- Kernel Function
  – So, any kernel K(x,y) satisfying (1) & (2), corresponds to an inner product in SOME space!!!

  – Kernel trick: We do not have to know the mapping function, but for some kernel functions we try to linearly separate pattern sets in a high dimensional space only using a function of the inner product in the original space.

Non Linear SVM

- Polynomial:

$$K(x_i, x_j) = (x_i^T x_j + 1)^q, \quad q > 0$$

- Radial Basis Functions:

$$K(\underline{x}_i, \underline{x}_j) = \exp\left(-\frac{\left\| \underline{x}_i - \underline{x}_j \right\|^2}{\sigma^2} \right)$$

- Hyperbolic Tangent:

$$K(x_i, x_j) = \tanh(\beta x_i^T x_j + \gamma)$$

for appropriate values of β, γ (e.g. $\beta = 2$ and $\gamma = 1$).

Non-linear SVM Formulation

- Support Vector Machines Formulation
- **Step 1:** Choose appropriate kernel. This implicitly assumes a mapping to a higher dimensional (yet, not known) space.
- **Step 2:**

$$\lambda = \text{argmax}_{\lambda} \left(\sum_i \lambda_i - 1/2 \sum_{ij} \lambda_i \lambda_j y_i y_j K(x_i, x_j)\right)$$

subject to: $0 \leq \lambda_i \leq C$, $i = 1,2,\ldots,N$, $\sum_i \lambda_i y_i = 0$

This results to an implicit combination
$w = \sum_i^{N_s} \lambda_i y_i \varnothing(x_i)$

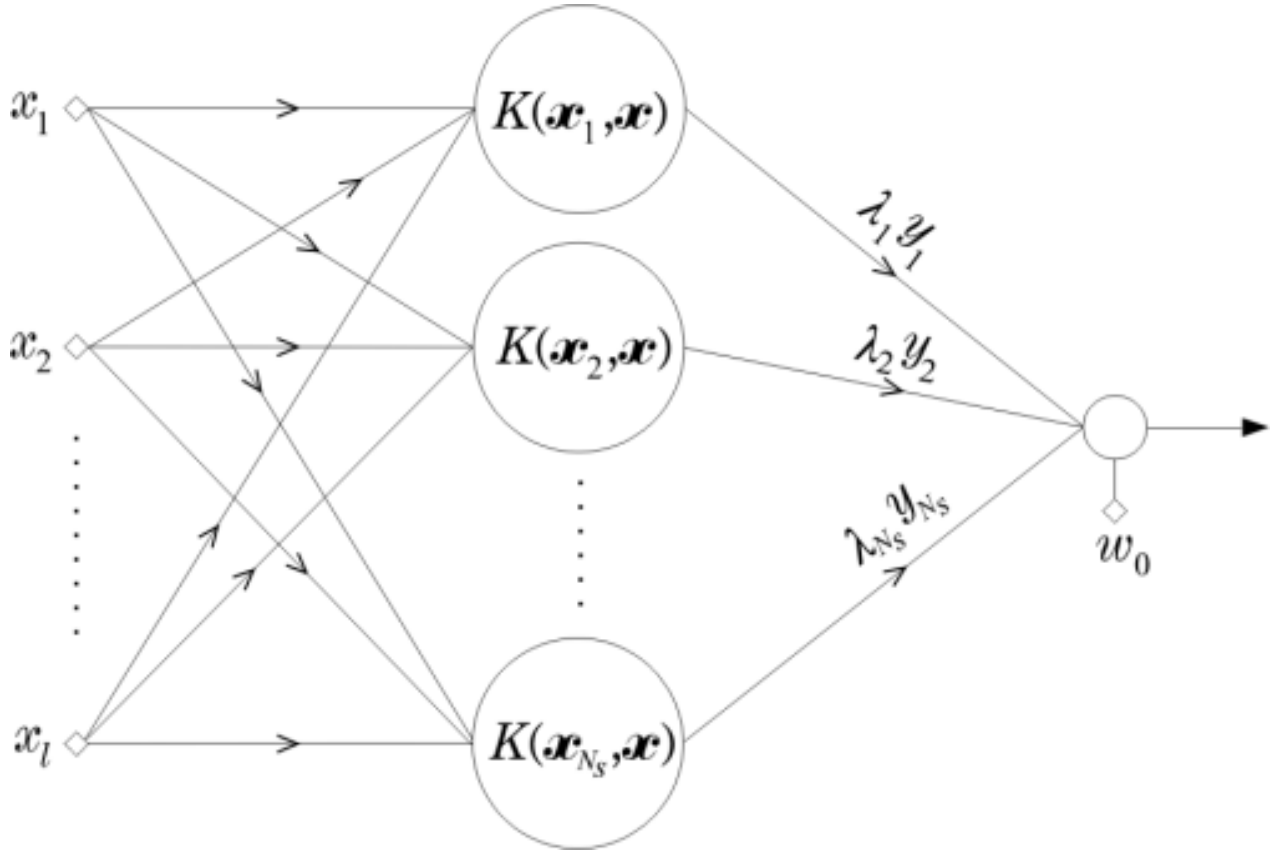
- **Step 3:** Assign x to

w_1 if $g(x) = \sum_i^{N_s} \lambda_i y_i K(x_i, x) \geq 0$

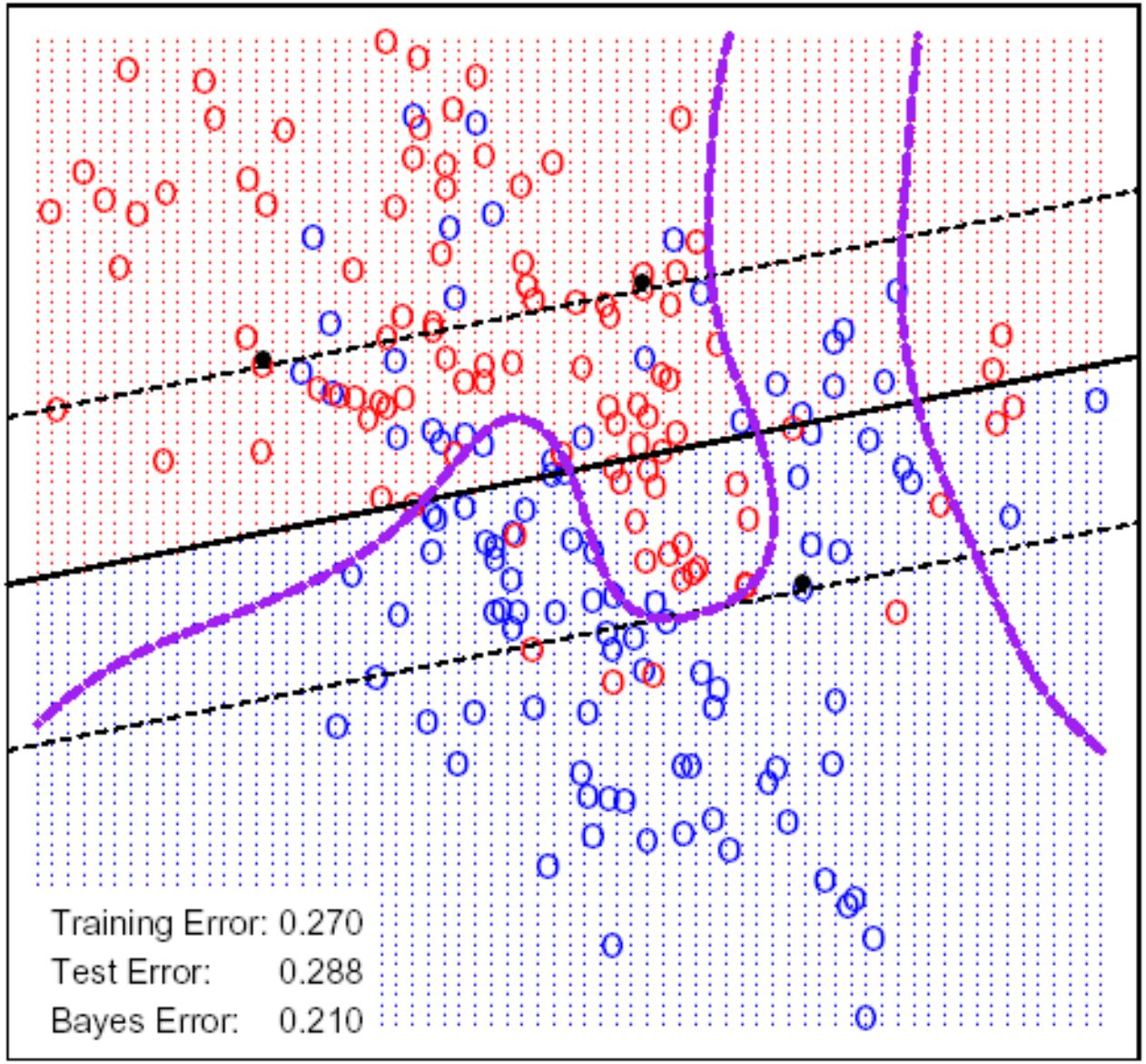
w_2 if $g(x) = \sum_i^{N_s} \lambda_i y_i K(x_i, x) < 0$

Non-linear SVM Formulation

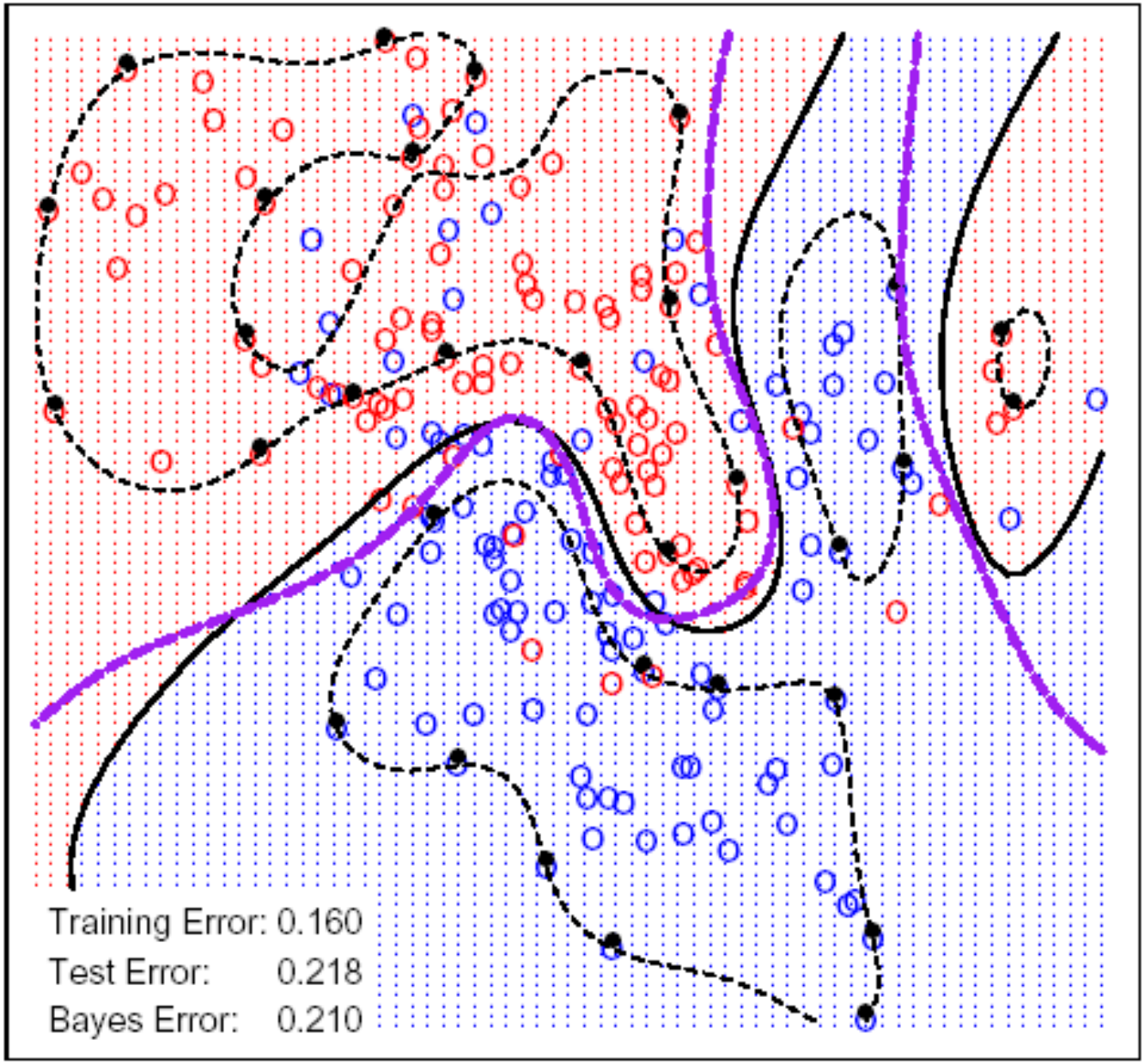
- The SVM Architecture
- SVM special case of a two-layer neural network with special activation function and a different learning method.
- Their attractiveness comes from their good generalisation properties and simple learning.



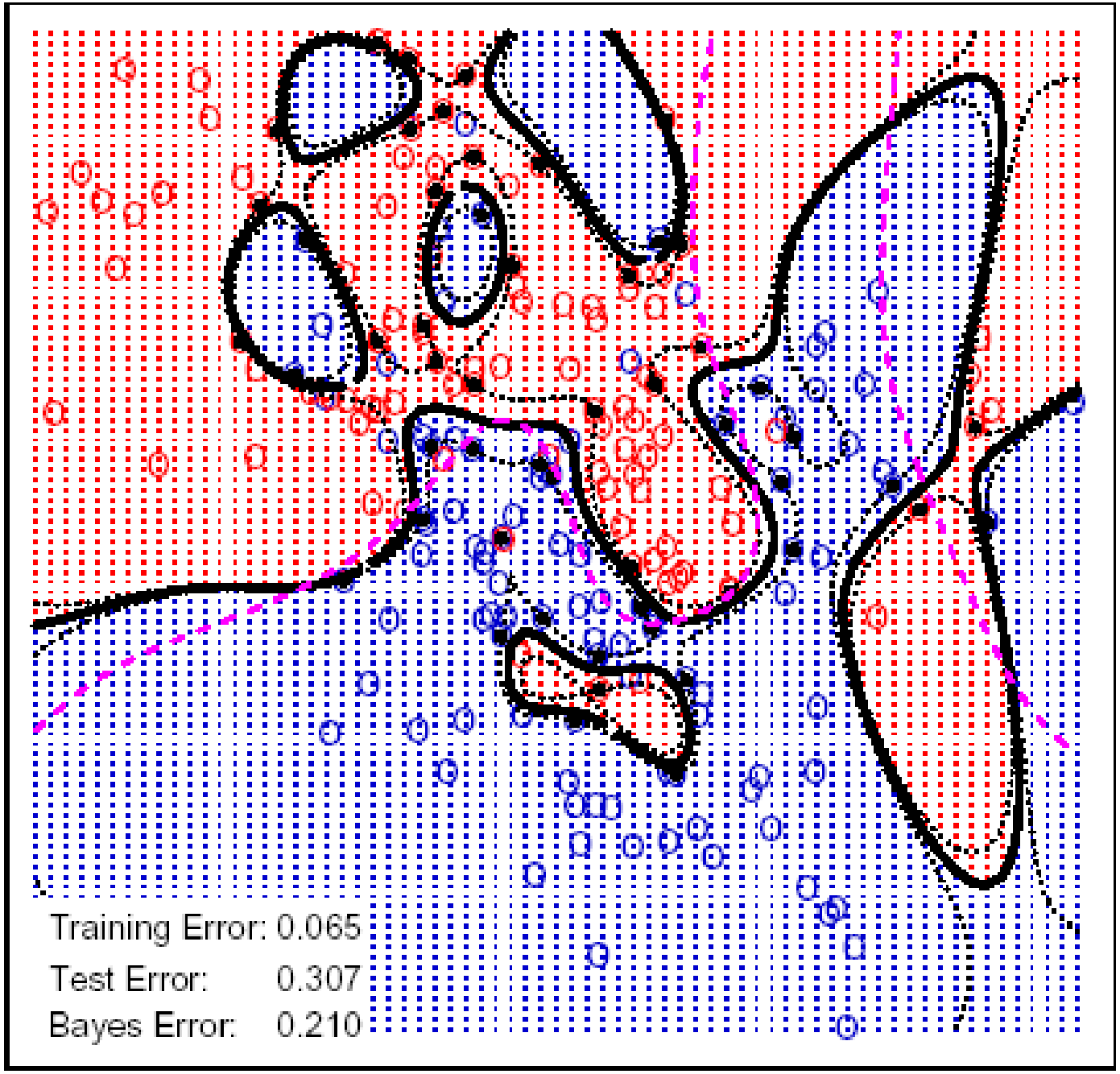
Linear SVM – Pol SVM in the input Space X



Pol. SVM – RBF SVM in the input space X

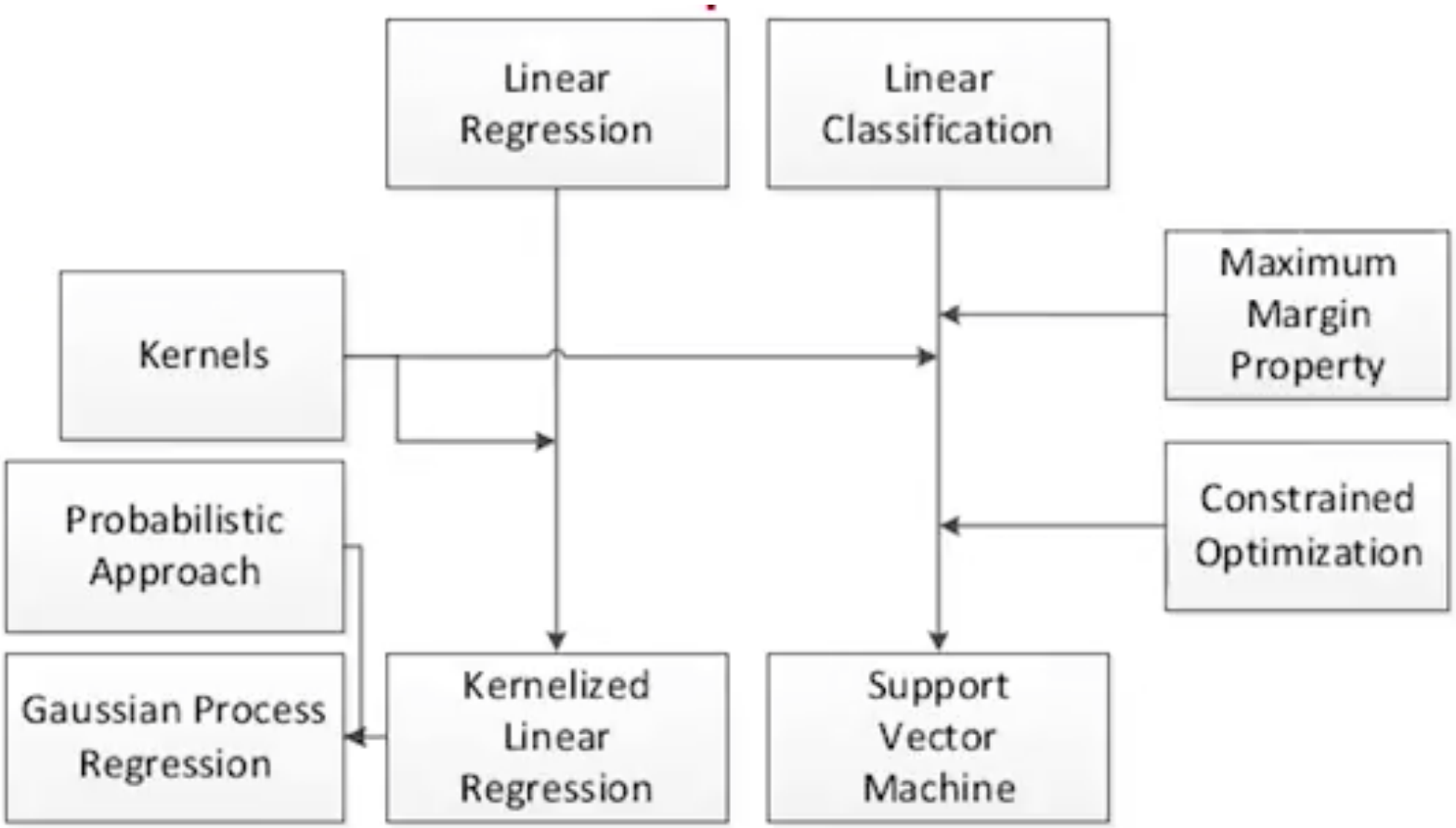


Pol. SVM – RBF SVM in the input space X



Kernels Summary

- Kernels compute the value of a scalar product in high dimensional feature space without explicitly computing the features.
- They can be used in any model that depends (or can be rewritten so that it depends) on the scalar product of the training samples.
- Not every function is a kernel.
- Use Kernel construction rules to prove that a function is a valid kernel.


Linear Regression
Linear Classification
Kernels
Maximum Margin Property
Probabilistic Approach
Constrained Optimization
Gaussian Process Regression
Kernelized Linear Regression
Support Vector Machine

SVM vs. Neural Networks

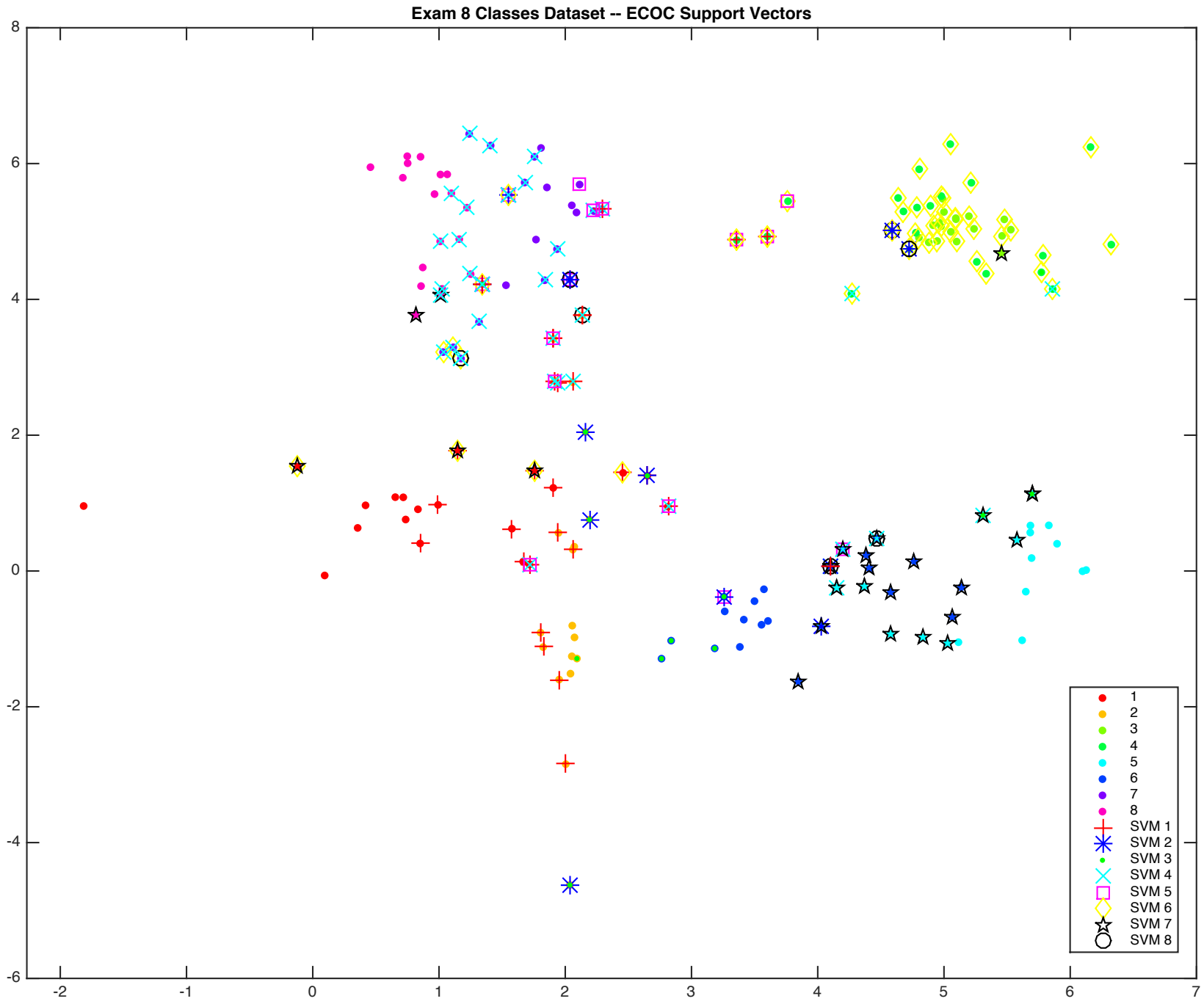
- Training a Support Vector Machine (SVM) involves optimisation of a *concave* function: there is a unique solution
- Training a neural network learning model is generally non-convex: there are potentially different solutions depending on the starting values for the model parameters.

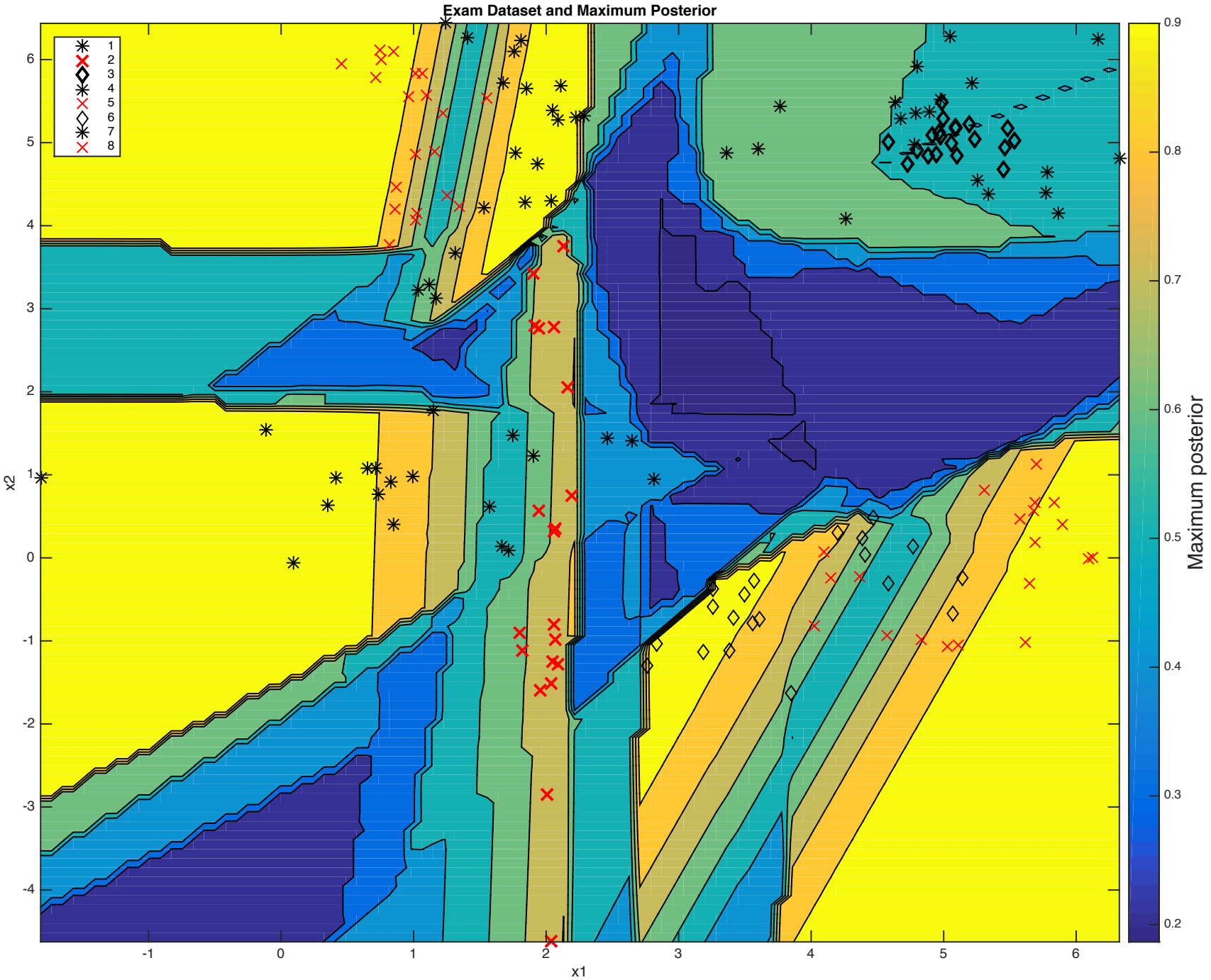
SVM Software

- SVM^{light}: Thorsten Joachims - free software in C, known for quality and speed.
- *LIB SVM*: free software based on Platt's SMO algorithm and Joachims code, written by Chih-Chung Chang and Chih-Jen Lin.
- *Equbits*: Commercial software package which automates the tuning and model selection with SVMs

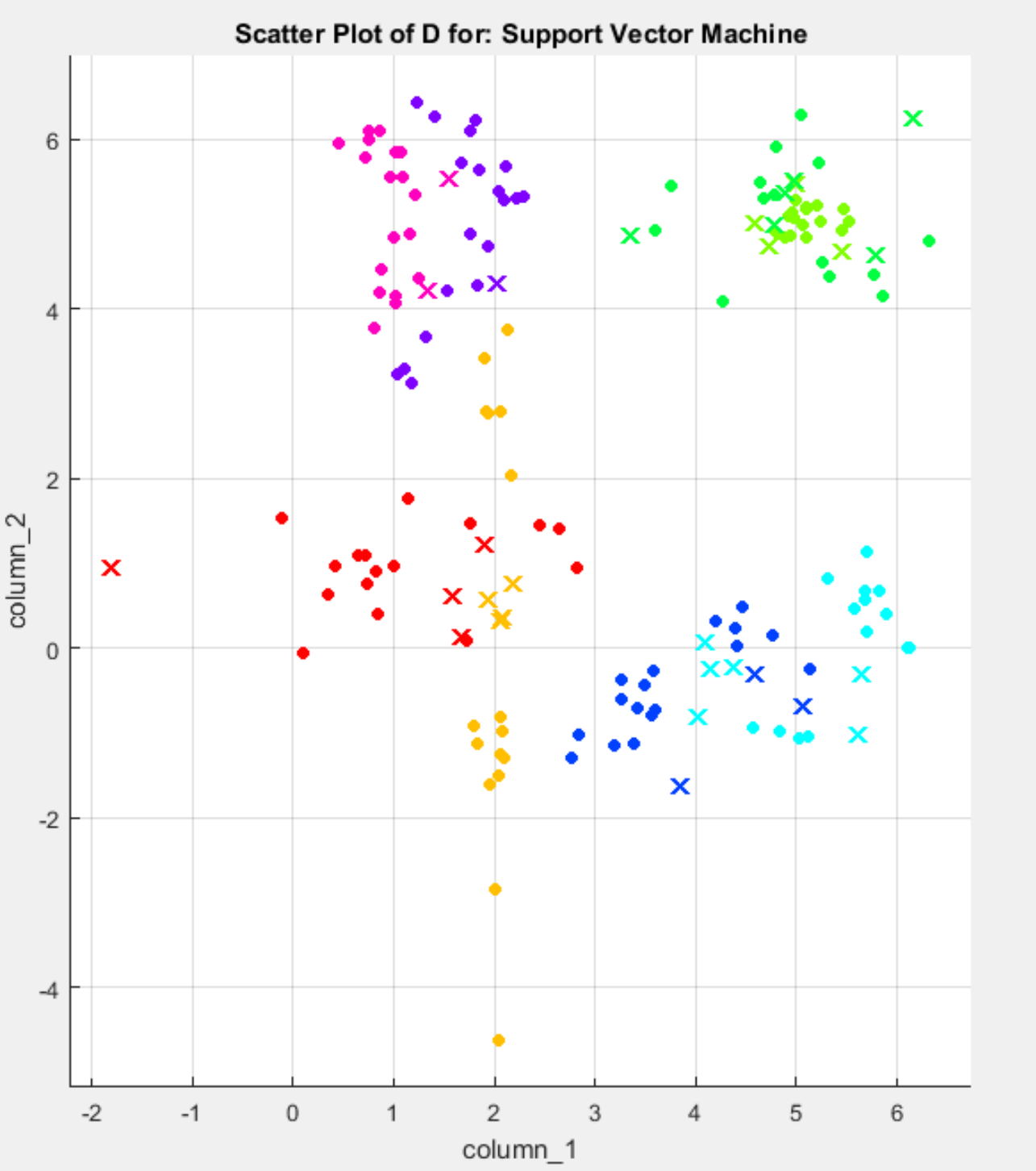
Matlab SVM Multiclass Function (fitcecoc)


```
X = D(:, 1:2)
Y = D(:, 3)
[n,p] = size(X)
isLabels = unique(Y);
nLabels = numel(isLabels)
tabulate(categorical(Y))

t = templateSVM('Standardize',1,'SaveSupportVectors',true);
predictorNames = {'x1','x2'};
classNames = {'1','2','3','4','5','6','7','8'};
Mdl = fitcecoc(X,Y,'Learners',t,'FitPosterior',1,'PredictorNames',predictorNames,'ClassNames',classNames)
[label,~,~,Posterior] = resubPredict(Mdl,'Verbose',1);  % takes long time, but eventually converges
Mdl.BinaryLoss

idx = randsample(size(X,1),10,1);
Mdl.ClassNames
table(Y(idx),label(idx),Posterior(idx,:),'VariableNames',{'TrueLabel','PredLabel','Posterior'})

xMax = max(X);xMin = min(X);
x1Pts = linspace(xMin(1),xMax(1));x2Pts = linspace(xMin(2),xMax(2));
[x1Grid,x2Grid] = meshgrid(x1Pts,x2Pts);

[~,~,~,PosteriorRegion] = predict(Mdl,[x1Grid(:),x2Grid(:)]);   % takes long time,, but eventually converges

figure;
contourf(x1Grid,x2Grid,reshape(max(PosteriorRegion,[],2),size(x1Grid,1),size(x1Grid,2)));
h = colorbar;
h.YLabel.String = 'Maximum posterior';
h.YLabel.FontSize = 15;
hold on
gh = gscatter(X(:,1),X(:,2),Y,'krk','*xd',8);
gh(2).LineWidth = 2; gh(3).LineWidth = 2;

title 'Exam Dataset and Maximum Posterior';
xlabel 'x1';ylabel 'x2';
axis tight
legend(gh,'Location','NorthWest')
hold off

sv = cell(L,1); % Preallocate for support vector indices
for j = 1:L;
    SVM = Mdl.BinaryLearners{j};
    sv{j} = SVM.SupportVectors;
    sv{j} = bsxfun(@plus,bsxfun(@times,sv{j},SVM.Sigma),SVM.Mu);
end

figure;
h = zeros(8 + L,1); % Preallocate for handles
h(1:8) = gscatter(D(:,1),D(:,2),D(:,3));
hold on;
markers = {'ko','r+','b*', 'g.','cx','ms','yd','kp'};
for j = 1:L;
    svs = sv{j};
    h(j+8) = plot(svs(:,1),svs(:,2),markers{mod(j, length(markers))+1}, 'MarkerSize',10);
end
title 'Exam 8 Classes Dataset -- ECOC Support Vectors';
legend(h,[classNames,{'SVM 1','SVM 2','SVM 3', 'SVM 4', 'SVM 5', 'SVM 6', 'SVM 7', 'SVM 8'}],'Location','Best')
hold off

CVMdl = crossval(Mdl);
oosLoss = kfoldLoss(CVMdl)

oofLabel = kfoldPredict(CVMdl);
ConfMat = confusionmat(Y,oofLabel);

% Convert the integer label vector to a class-identifier matrix.
[~,grp] = ismember(oofLabel,isLabels);
oofLabelMat = zeros(nLabels,n);
idxLinear = sub2ind([nLabels n],grp,(1:n)');
oofLabelMat(idxLinear) = 1; % Flags the row corresponding to the class
YMat = zeros(nLabels,n);
idxLinearY = sub2ind([nLabels n],grp,(1:n)');
YMat(idxLinearY) = 1;

figure;
plotconfusion(YMat,oofLabelMat);
h = gca;
h.XTickLabel = [num2cell(isLabels); {''}];
h.YTickLabel = [num2cell(isLabels); {''}];
```

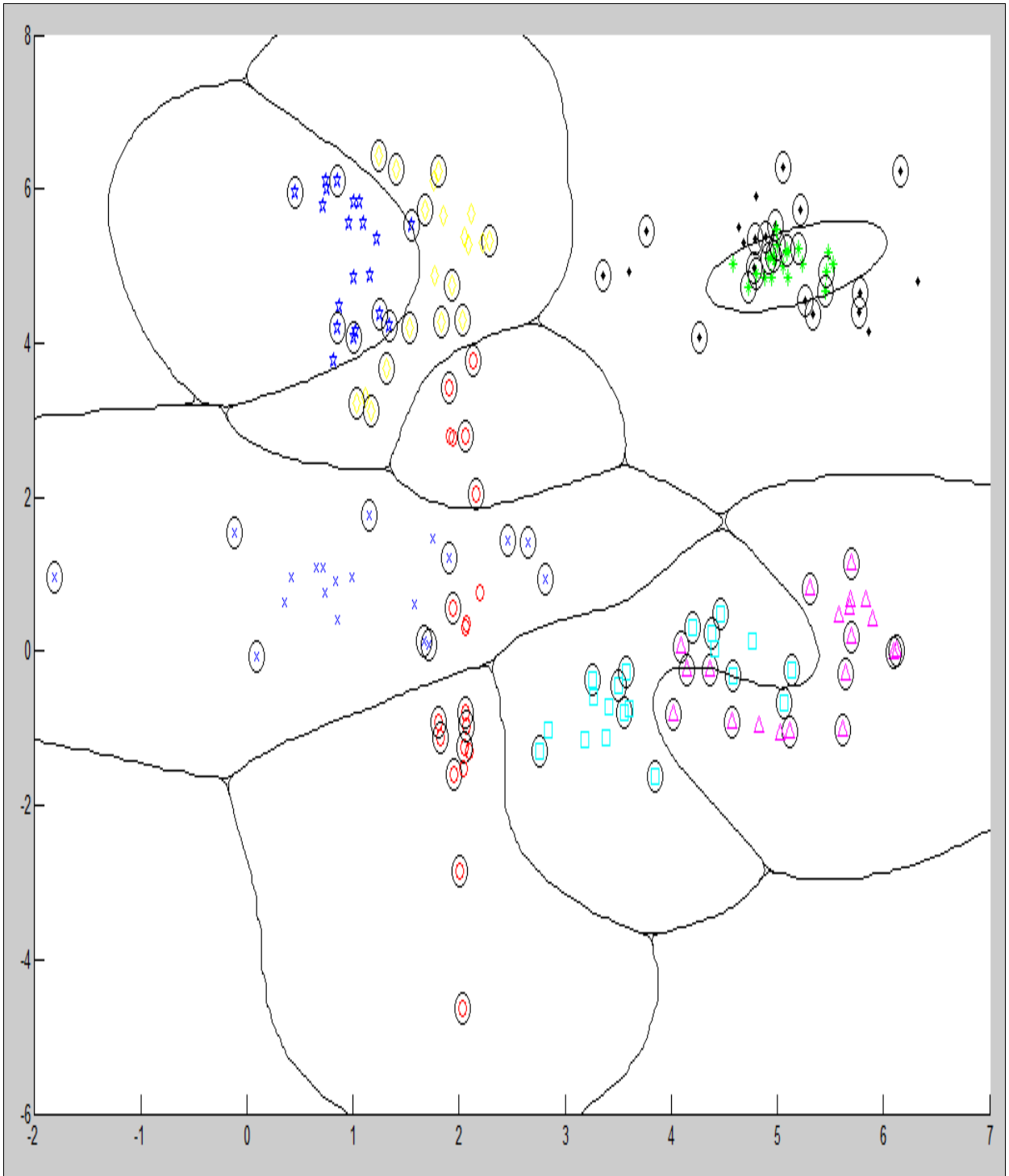

Exam 8 Classes Dataset -- ECOC Support Vectors
8
6
4
2
0
-2
-4
-6
-2
-1
0
1
2
3
4
5
6
7
1
2
3
4
5
6
7
8
SVM 1
SVM 2
SVM 3
SVM 4
SVM 5
SVM 6
SVM 7
SVM 8


Exam Dataset and Maximum Posterior
1
2
3
4
5
6
7
8
x1
x2
Maximum posterior
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2

Using LibSVM for the Fall 2015 Exam Dataset



Using LibSVM for the Fall 2015 Exam Dataset



Matlab SVM Functions

- For Fall 2014 exam dataset, the polynomial kernel function in SVM was the best scoring:

```
xtrain = Dataset(1:80,1:2)

ytrain = Dataset(1:80, 3)

xtest = Dataset(81:100,1:2)

%The Polynomial Kernel SVM classifier:

SVMStruct = svmtrain(xtrain,ytrain, 'kernel_function','polynomial’)

ytest  = svmclassify(SVMStruct , xtest )

bad = ~strcmp(svmclassify(SVMStruct , xtrain),ytrain)

P_SVMResubErr = sum(bad) / 80

cp = cvpartition(ytrain,'k',10)

SVMClassFun = @(xtrain,ytrain,xtest (svmclassify(SVMStruct ,xtest))

P_SVMCVErr = crossval('mcr',xtrain,ytrain,'predfun',SVMClassFun,'partition',cp)
```

Other Matlab SVM functions

```
SVMModel1 = fitcsvm(xtrain,ytrain,'KernelFunction','polynomial','Standardize',true);
% Compute the scores over a grid
d = 0.02; % Step size of the grid
[x1Grid,x2Grid] = meshgrid(min(xtrain(:,1)):d:max(xtrain(:,1)),...
    min(xtrain(:,2)):d:max(xtrain(:,2)));
xGrid = [x1Grid(:),x2Grid(:)];        % The grid
[~,scores1] = predict(SVMModel1,xGrid);  % The scores
figure;
h(1:2) = gscatter(xtrain(:,1),xtrain(:,2),ytrain)
hold on
h(3) = plot(xtrain(SVMModel1.IsSupportVector,1),...
    xtrain(SVMModel1.IsSupportVector,2),'ko','MarkerSize',10); % Support vectors
contour(x1Grid,x2Grid,reshape(scores1(:,2),size(x1Grid)),[0 0],'k');
% Decision boundary
title('Scatter Diagram with the Decision Boundary')
legend({'-1','1','Support Vectors'},'Location','Best');
hold off
```