Lecture 3: SVM dual, kernels and regression

IMAGE

C19 Machine Learning Hilary 2015 A. Zisserman

IMAGE

• Primal and dual forms

• Linear separability revisted

• Feature maps

• Kernels for SVMs

• Regression

• Ridge regression

• Basis functions

SVM – review

IMAGE

• We have seen that for an SVM learning a linear classifier f(x) = w>x + b

is formulated as solving an optimization problem over w : XN wmin ∈Rd||w|| 2+ C max (0, 1 − yif(xi)) i

• This quadratic optimization problem is known as the primal problem.

• Instead, the SVM can be formulated to learn a linear classifier XN f(x) = αiyi(xi>x) + b

i

by solving an optimization problem over αi.

• This is know as the dual problem, and we will look at the advantages of this formulation.

Sketch derivation of dual form

IMAGE

The Representer Theorem states that the solution w can always be written as a linear combination of the training data:

XN j=1

w=

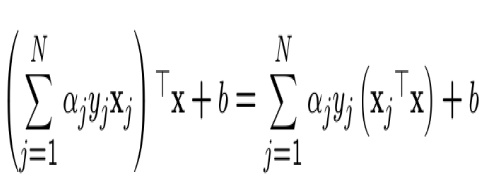
αjyjxj

Proof: see example sheet .

Now, substitute for w in f(x) = w>x + b

f(x) = ⎝X ´ and for w in the cost function minw ||w||> subject to yi w xi + b ´≥ 1, ∀i

³



||w|| 2= ⎩⎧⎨X αjyjxj ⎫ ⎬>⎧⎨Xαkykxk⎬⎫ = j ⎭⎩ k ⎭ X αjαkyjyk(xj>xk) jk

Hence, an equivalent optimization problem is over αj X ⎛XN min αjαkyjyk(xj>xk) subject to yi ⎝

αj jkαjyj(xj>xi) + b⎠ ⎞≥ 1, ∀i j=1

and a few more steps are required to complete the derivation.

Primal and dual formulations

IMAGE

N is number of training points, and d is dimension of feature vector x. Primal problem: for w ∈ Rd

XN wmin ∈Rd||w|| 2+ C max (0, 1 − yif(xi)) i

Dual problem: for α ∈ RN (stated without proof): maxX αi −1X αjαkyjyk(xj >xk) subject to 0 ≤ αi ≤ C for ∀i, andX αi≥0 2 i jk i

α i yi = 0

• Need to learn d parameters for primal, and N for dual

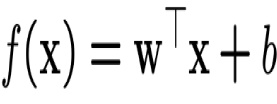
• If N << d then more eﬃcient to solve for α than w

• Dual form only involves (xj>xk). We will return to why this is an advantage when we look at kernels.

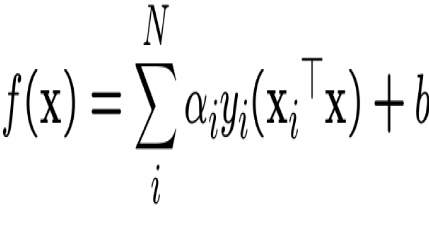
Primal and dual formulations

IMAGE

Primal version of classifier:

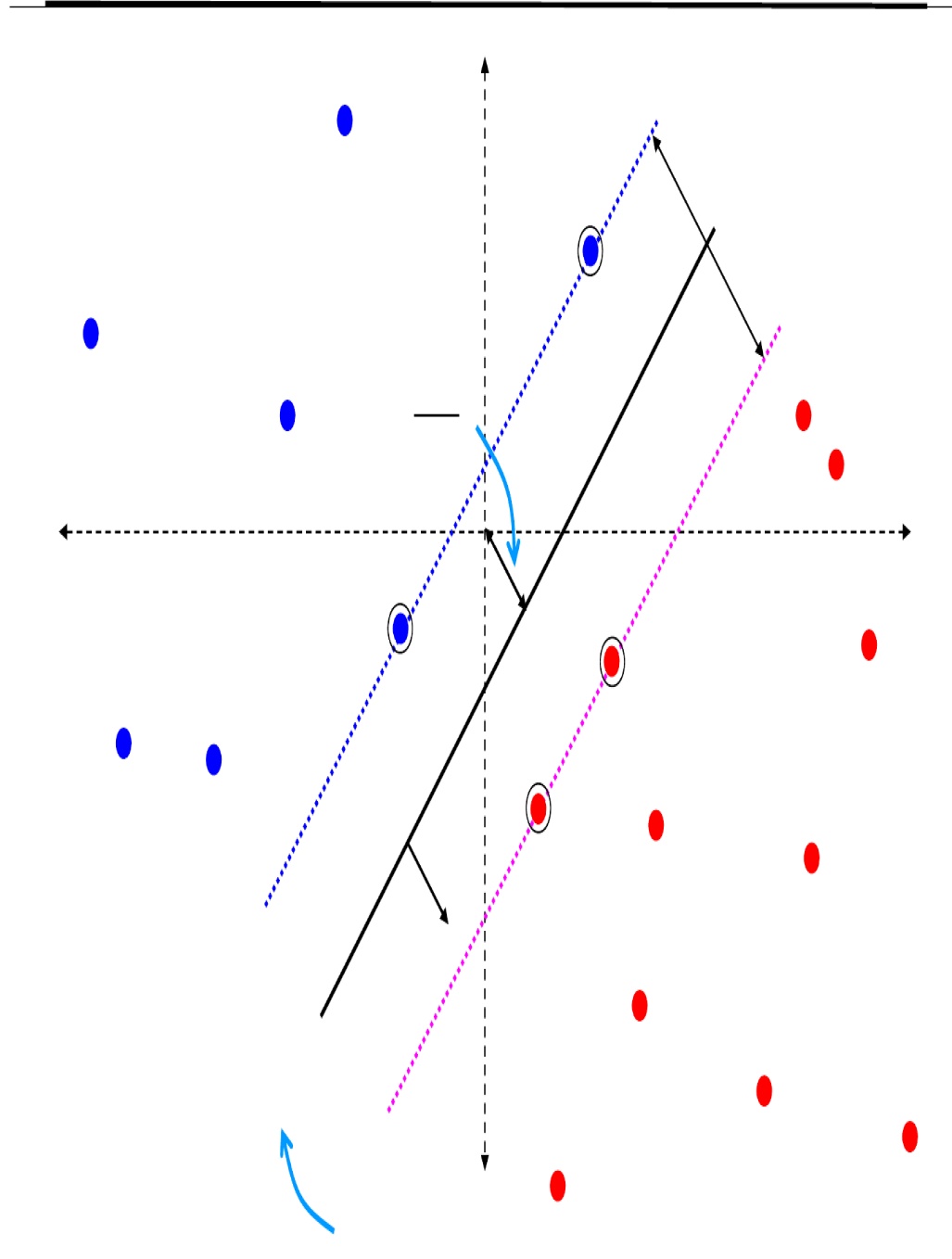


Dual version of classifier:



At first sight the dual form appears to have the disadvantage of a K-NN classifier — it requires the training data points xi. However, many of the αi’s are zero. The ones that are non-zero define the support vectors xi.

Support Vector Machine



**Support Vector**

**Support Vector**

f(x) =

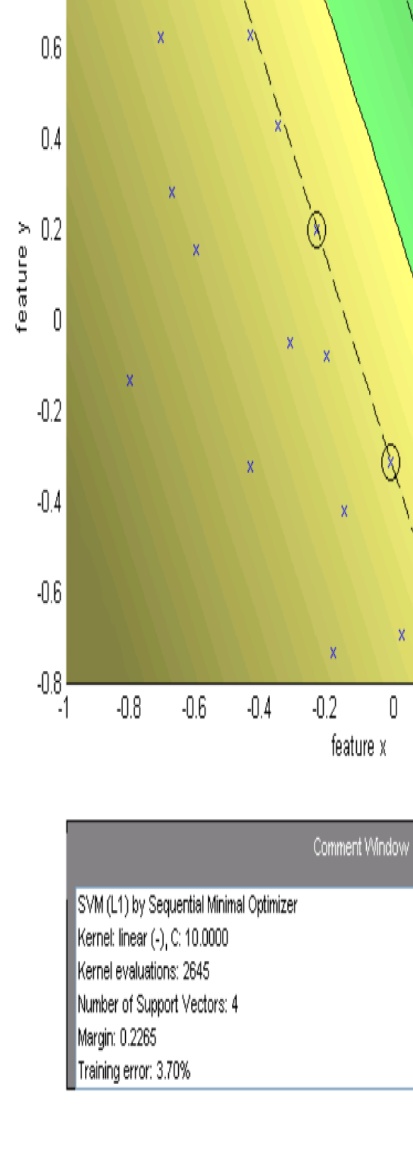
support vectors

**w**

X   
αiyi(xi>x) + b  
i

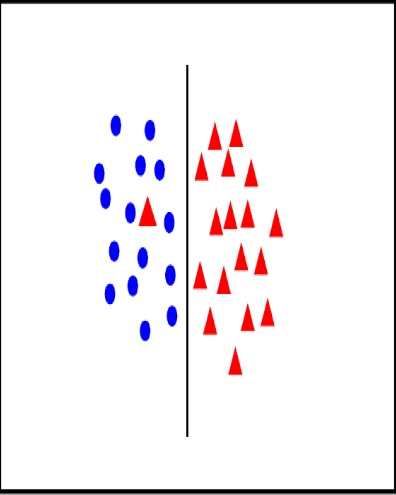
soft margin

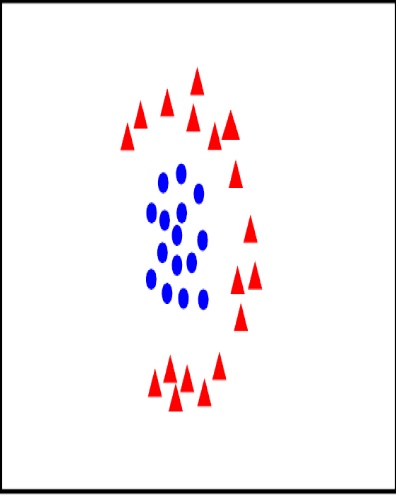
C = 10



Handling data that is not linearly separable

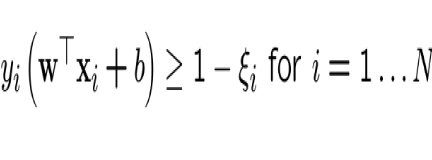
IMAGE





• introduce slack variables XN min||w|| 2+ C ξi w∈Rd,ξ i∈R+ i

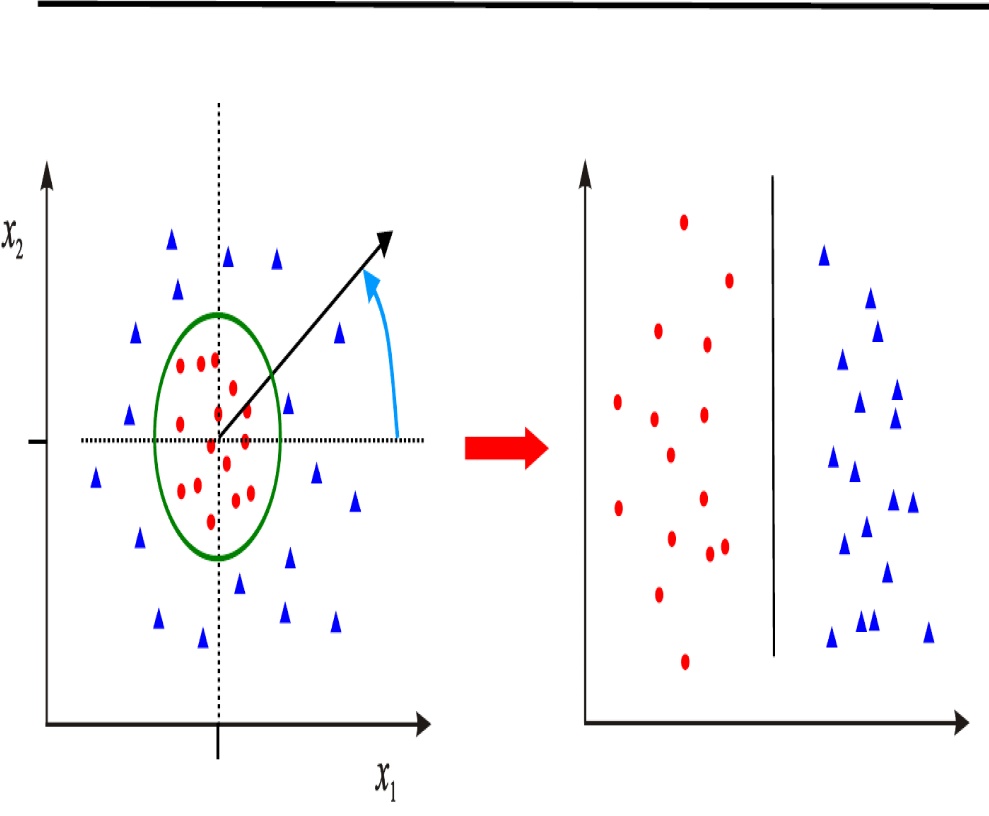
subject to



• linear classifier not appropriate

??

Solution 1: use polar coordinates



r

θ

<0

>0

θ

0

0

r

• Data is linearly separable in polar coordinates

• Acts non-linearly in original space Ã x1! Ãr! Φ: x2→ θR2 →R2

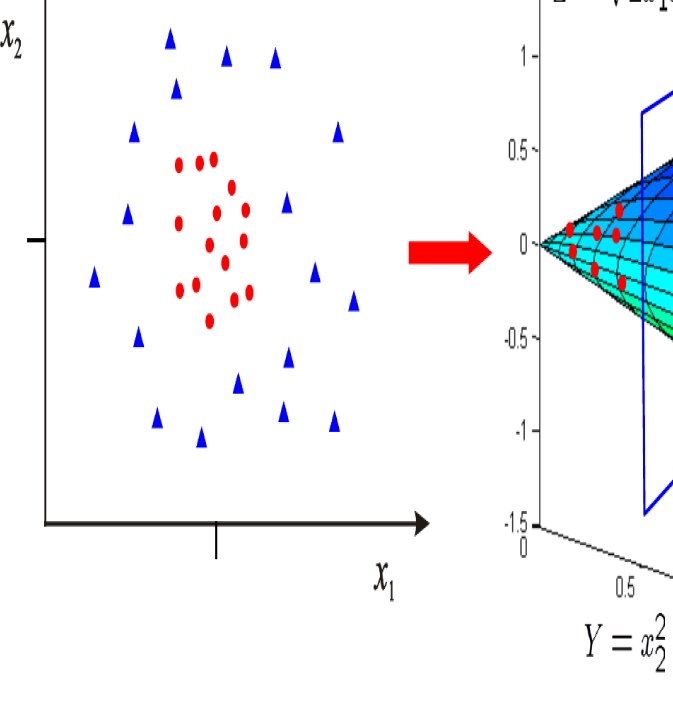
Solution 2: map data to higher dimension

IMAGE

Ã ! ⎛ x12⎞ → ⎝ ⎜√ x2 2⎟⎠ R2 → R3 2x1x2

x1 x2

Φ:



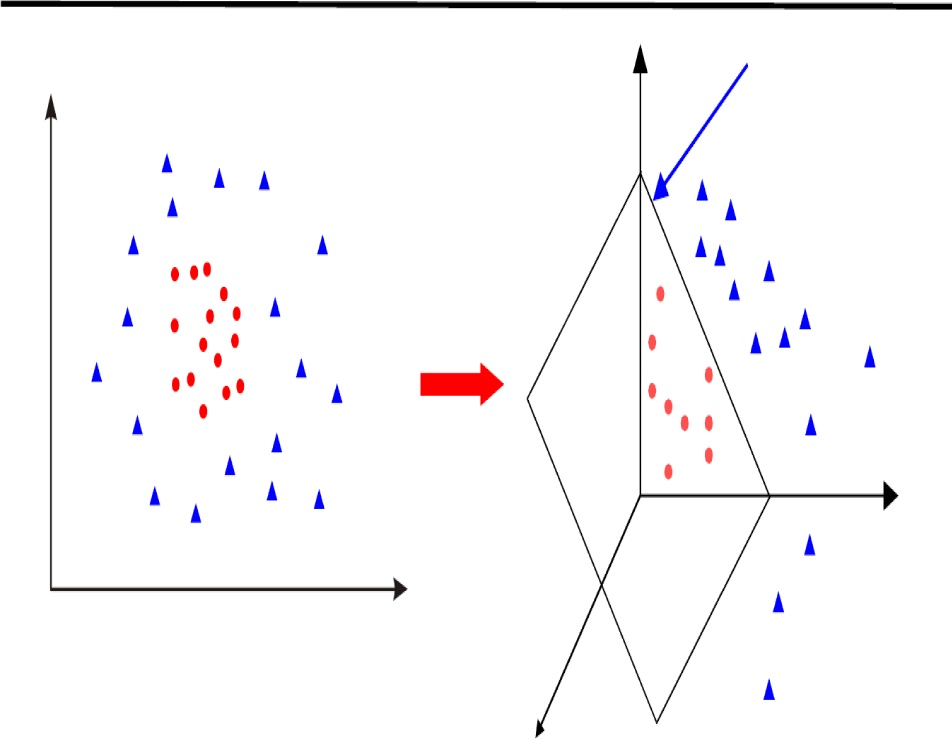
0

0

• Data is linearly separable in 3D

• This means that the problem can still be solved by a linear classifier

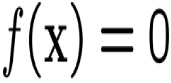
SVM classifiers in a transformed feature space



Φ : x → Φ(x) Rd → R D

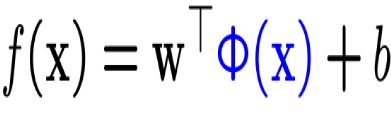
Rd

RD



Φ

Learn classifier linear in w for RD:



Φ(x) is a feature map

Primal Classifier in transformed feature space

IMAGE

Classifier, with w ∈ RD:



Learning, for w ∈ RD

XN wmin ∈RD||w||2 + C imax (0, 1 − yif(xi))

• Simply map x to Φ(x) where data is separable

• Solve for w in high dimensional space RD

• If D >> d then there are many more parameters to learn for w. Can this be avoided?

Dual Classifier in transformed feature space

IMAGE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | f(x) =  → f(x) = | | | X N  i  X N  i | αi y i x i > x + b  αiyi Φ(xi)>Φ(x) + b |
| Learning: | max  αi≥0  → max αi≥0 | X i  X i | αi −  αi − | 1X 2 jk  αj αk y j y k x j > x k  1X 2 jk  αjαkyjykΦ(xj)>Φ(xk) | |
|  |
|  |
| subject to |  |  |  |  |  |
|  | 0 ≤ αi ≤ C for ∀i, and | | | | |

Classifier:

i

Dual Classifier in transformed feature space

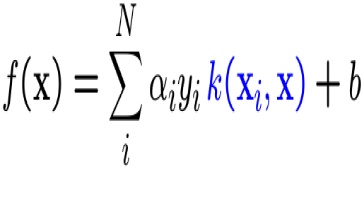
IMAGE

• Note, that Φ(x) only occurs in pairs Φ(xj)>Φ(xi)

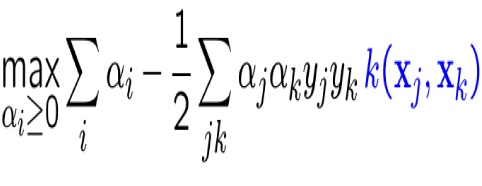
• Once the scalar products are computed, only the N dimensional vector α needs to be learnt; it is not necessary to learn in the D dimensional space, as it is for the primal

• Write k(xj, xi) = Φ(xj)>Φ(xi). This is known as a Kernel

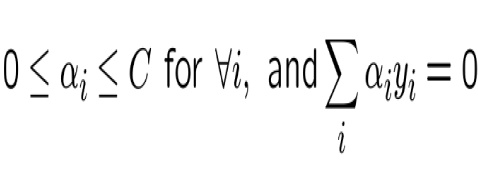
Classifier:



Learning:



subject to



Special transformations

IMAGE

|  |  |  |  |
| --- | --- | --- | --- |
| Φ: | x1  x2 !2  → ⎝ √ x2 ⎛ ⎜ ⎞  2x1x2 ⎟ ⎠ R2 → R3  Φ(x)>Φ(z) = ³ x2 1 ,x2 2,√ x´ ⎜ ⎝  ⎛z1 2  2x 1 2 z 2  √ 2 2z1z2  = x2 1 12 21 2 1 2 z 2 + x2z2 + 2x x z z | |  |
|  | ⎞ ⎟ ⎠ |
|  |
|  |
|  | = | (x1z1 + x2z2)2 |  |
|  | = | (x> z)2 |  |
| Kernel Trick | |  |  |
| • Classifier can be learnt and applied without explicitly computing Φ(x)  • All that is required is the kernel k(x, z) = (x>z)2 | | | |
|

Ã

x12

• Complexity of learning depends on N (typically it is O(N 3)) not on D

Example kernels

IMAGE

• Linear kernels k(x, x0) = x>x0 0³ >´d • Polynomial kernels k(x, x ) = 1 + x x0 for any d > 0 — Contains all polynomials terms up to degree d 0³ 0´ • Gaussian kernels k(x, x ) = exp −||x − x || 2/2σ 2 for σ > 0 — Infinite dimensional feature space

SVM classifier with Gaussian kernel

IMAGE

N = size of training data

XN f(x) = αiyik(xi, x) + b i

support vector



weight (may be

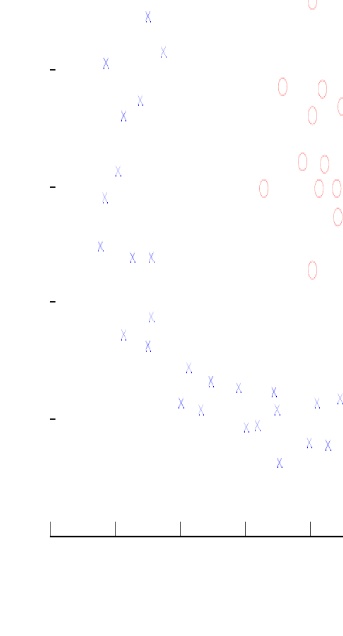
zero) Gaussian kernel k(x, x0) = exp ³ −||x − x0||2/2σ2´ Radial Basis Function (RBF) SVM f(x) =XN αiyi exp ³ −||x − xi|| 2/2σ2´ +b i

RBF Kernel SVM Example

IMAGE

0.6

0.4



0.2feature x

0.4

0.6

0.8

1

feature y

0.2

0

-0.2

-0.4

-0.6 -0.8

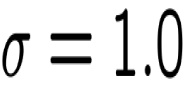
-0.6

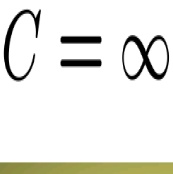
-0.4

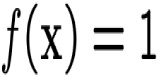
-0.2

0

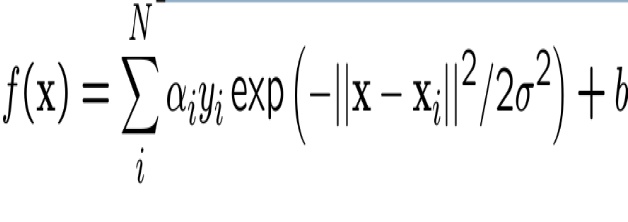
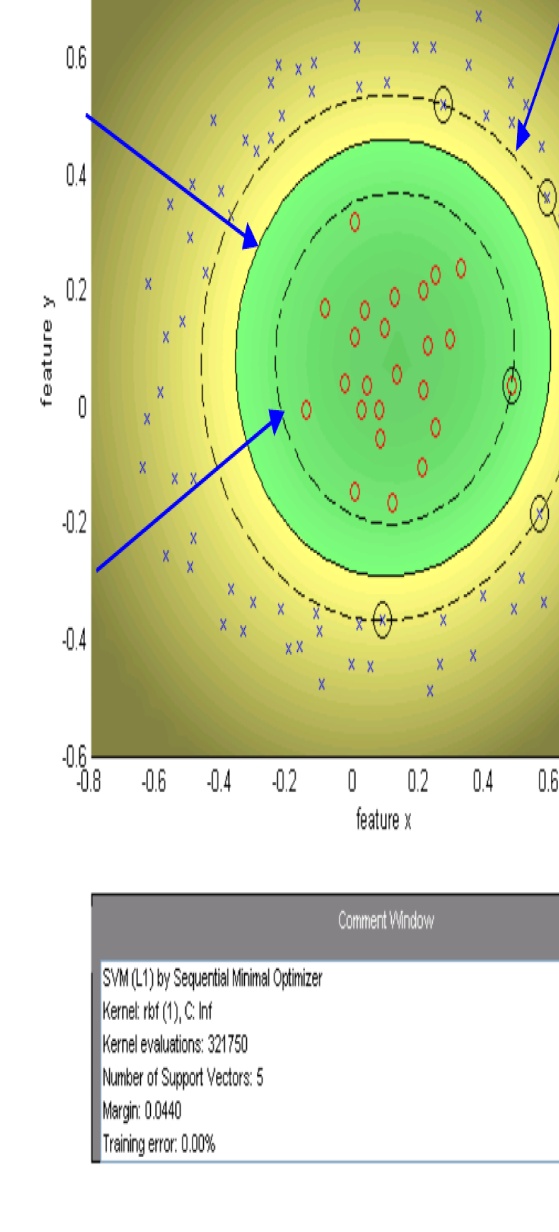
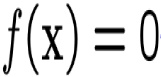
• data is not linearly separable in original feature space

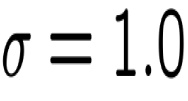


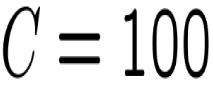


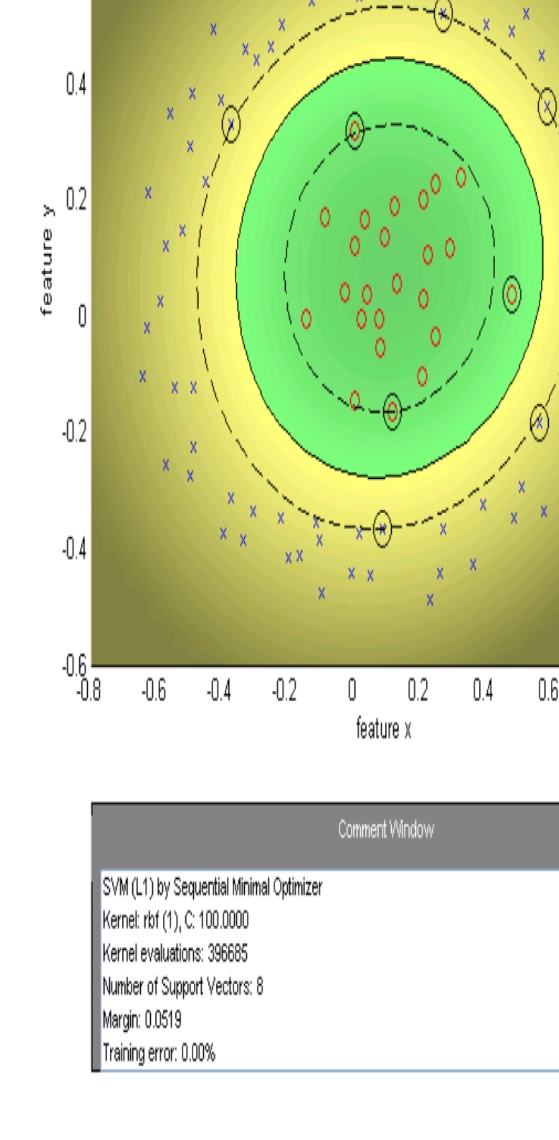


f(x) = −1





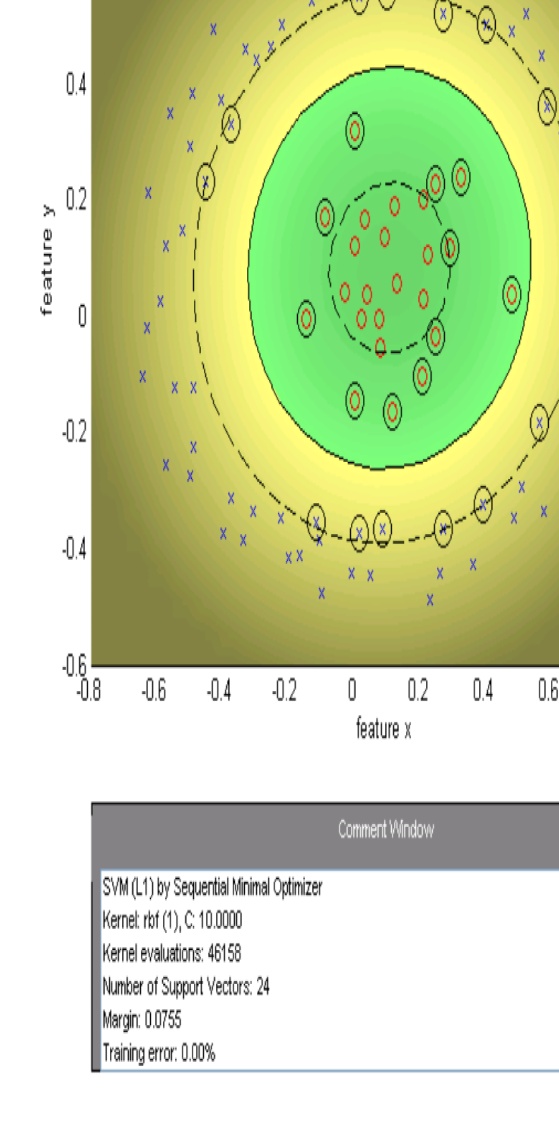
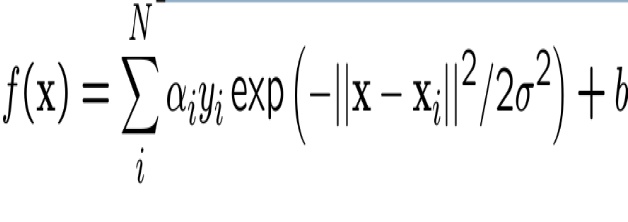


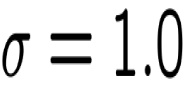


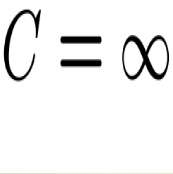
Decrease C, gives wider (soft) margin

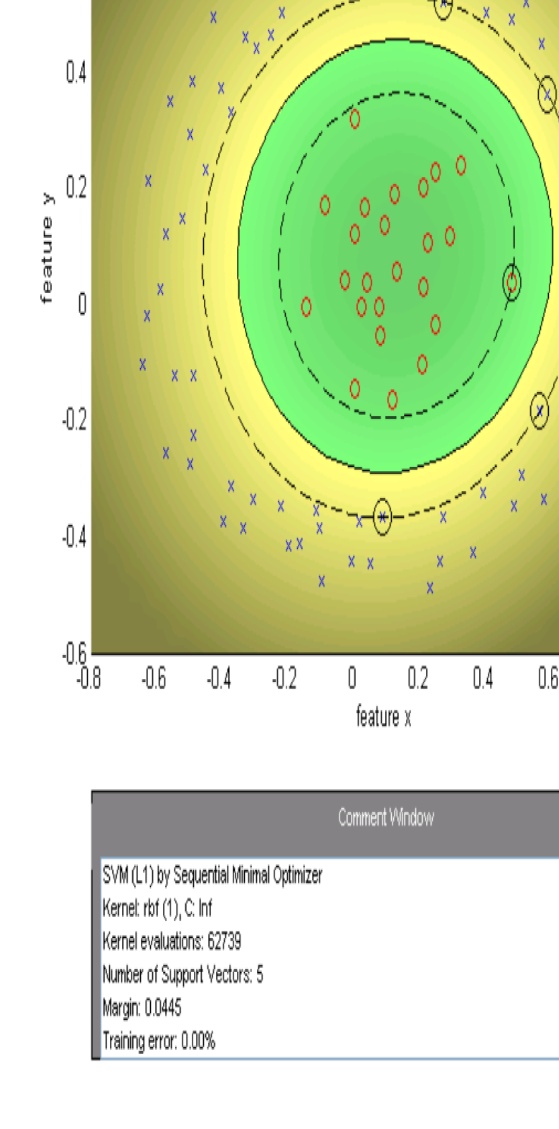
σ = 1.0

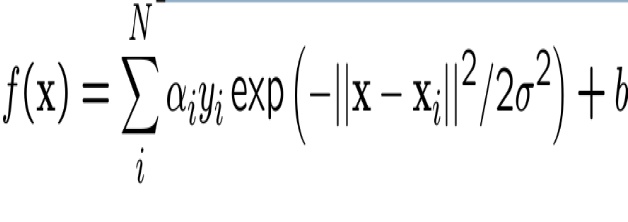
C = 10

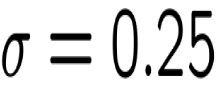


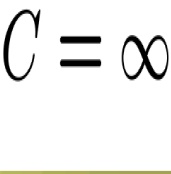


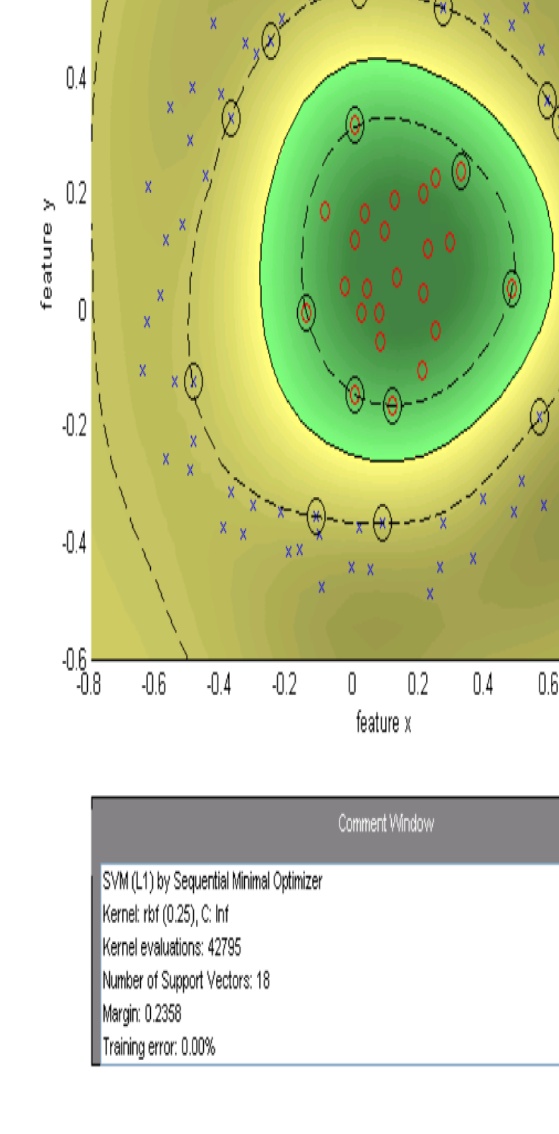








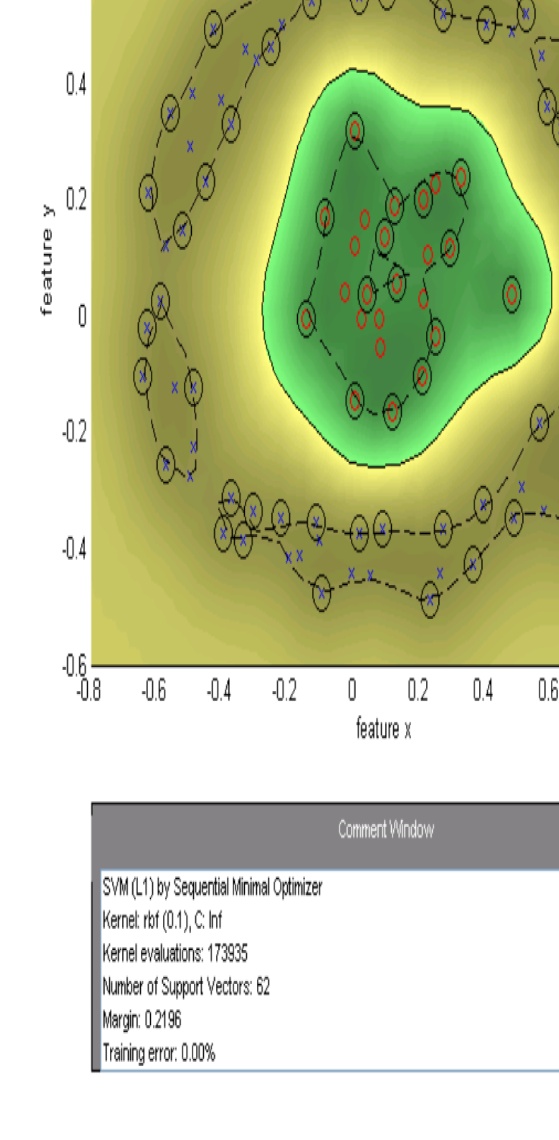
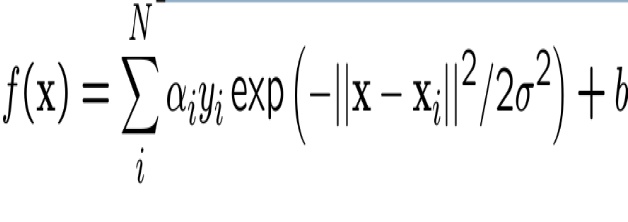




Decrease sigma, moves towards nearest neighbour classifier

σ = 0.1

C=∞



Kernel Trick - Summary

IMAGE

• Classifiers can be learnt for high dimensional features spaces, without actually having to map the points into the high dimensional space

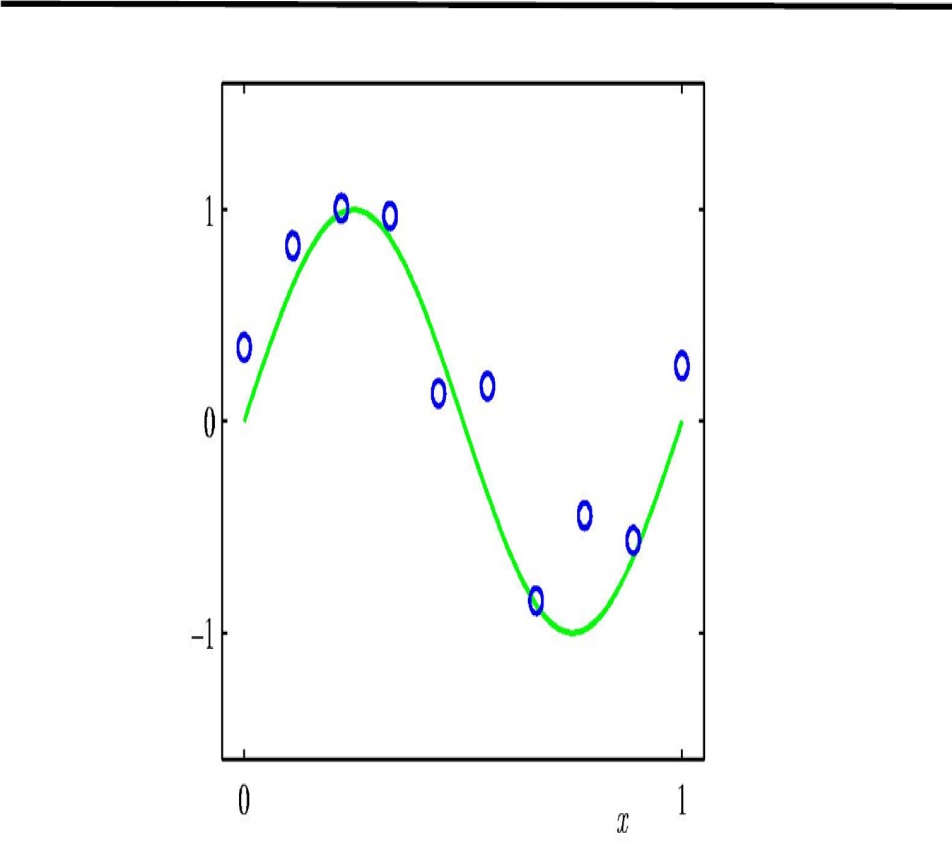
• Data may be linearly separable in the high dimensional space, but not linearly separable in the original feature space

• Kernels can be used for an SVM because of the scalar product in the dual form, but can also be used elsewhere – they are not tied to the SVM formalism

• Kernels apply also to objects that are not vectors, e.g.

k(h, h 0) = P 00 k min(hk, hk) for histograms with bins hk, hk

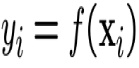
Regression



y

• Suppose we are given a training set of N observations ((x1, y1), . . . , (xN , yN )) with xi ∈ Rd, yi ∈ R

• The regression problem is to estimate f(x) from this data such that



Learning by optimization

IMAGE

• As in the case of classification, learning a regressor can be formulated as an optimization:

Minimize with respect to f ∈ F

XN l (f(xi), yi) + λR (f ) i=1



loss function regularization

• There is a choice of both loss functions and regularization

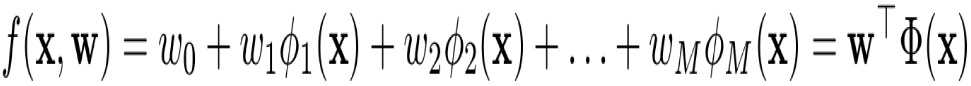
• e.g. squared loss, SVM “hinge-like” loss

• squared regularizer, lasso regularizer

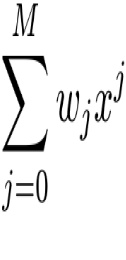
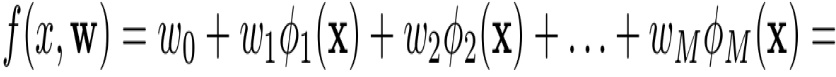
Choice of regression function – non-linear basis functions

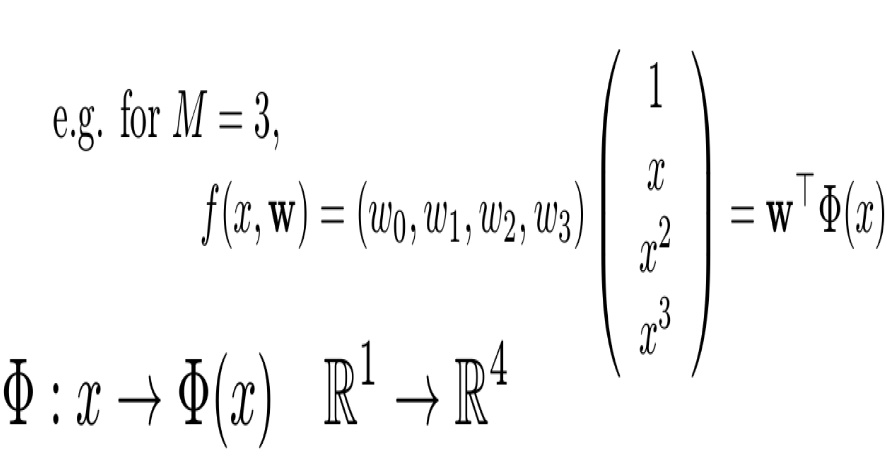
IMAGE

• Function for regression y(x, w) is a non-linear function of x, but linear in w:

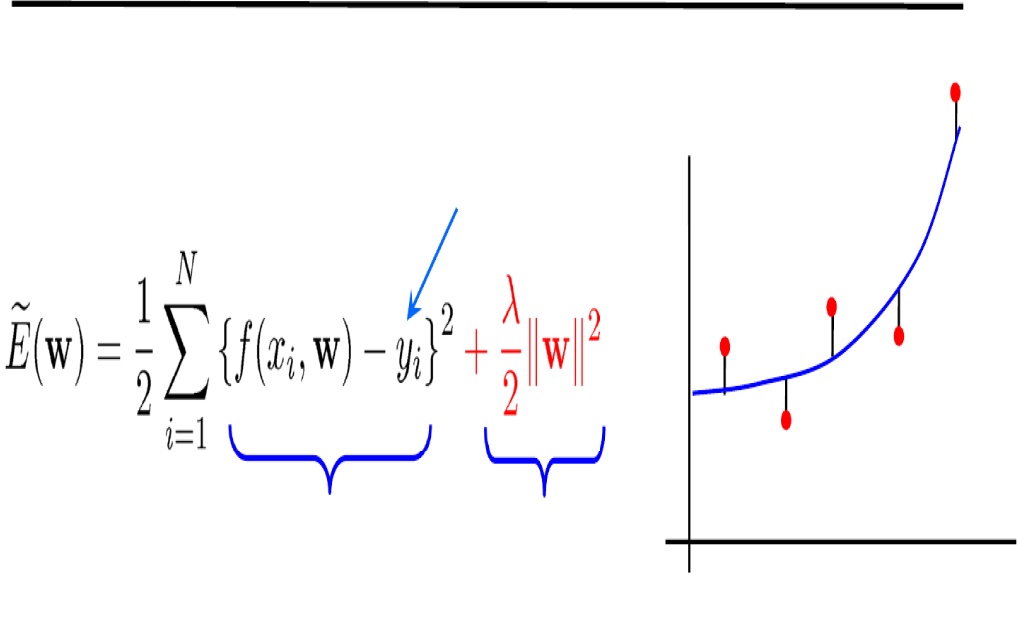


• For example, for x ∈ R, polynomial regression with φj (x) = xj :





Least squares “ridge regression”



• Cost function – squared loss:  
target value

yi

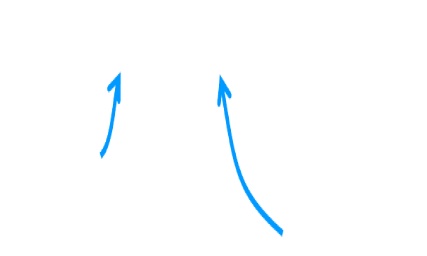
loss function regularization

xi

• Regression function for x (1D):

f(x, w) = w0 + w1φ1(x) + w2φ2(x) + . . . + wM φM (x) = w>Φ(x)

• NB squared loss arises in Maximum Likelihood estimation for an error model



yi = y˜i + ni ni   
∼ N(0, σ2)

true value

measured value

Solving for the weights w

IMAGE

Notation: write the target and regressed values as N-vectors =⎜⎜ ⎜⎜⎛ yy2 1⎟⎟⎟⎟⎞ =⎜⎜⎜ ⎜⎛ Φ(xΦ(x12) )>>w w⎟⎞ ⎟⎟⎟==⎢⎡ ⎢ ⎢ ⎢⎢11.φ1(x1)...φM (x1)⎥⎥⎥ ⎥⎤⎜⎜⎜ ⎜⎛ ww01⎞⎟⎟⎟⎟ φ1(x2) ... φM (x2) y ⎜⎜ . f ⎜⎜ . ⎝ ⎟⎟⎠ ⎝ ⎟⎟⎠⎢ ⎣. Φw

..⎥⎥⎦⎜⎜⎝ ⎟⎟⎠



yN Φ(xN )>w 1 φ1(xN ) . . . φM (xN ) wM

.

.

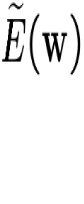
Φ is an N × M design matrix

e.g. for polynomial regression with basis functions up to x2

⎡ 1x1x12⎤ =⎢⎢⎢⎢ 1x2x2 ⎥⎥⎥⎛⎜ ⎞ ⎥⎝ w

2 w0 ⎢⎢ ..⎥ 1 ⎟⎠ ⎣ ..⎥⎦ w2 1 xN x2

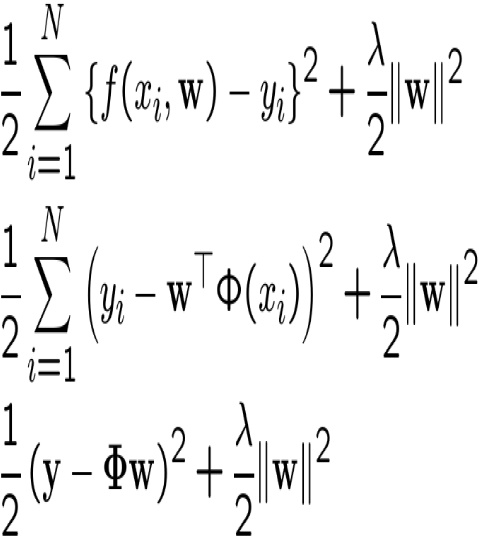
Φw



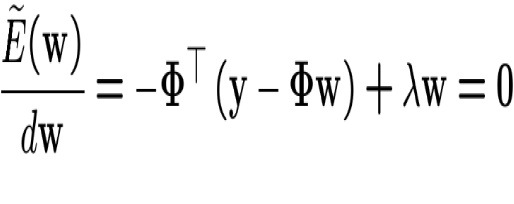
FORMULA

FORMULA

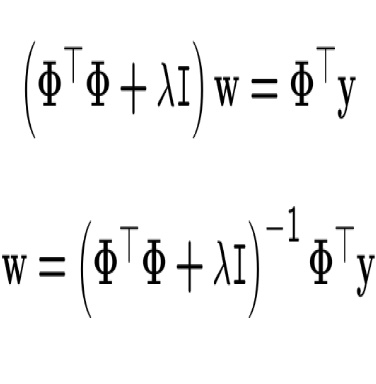
FORMULA



Now, compute where derivative w.r.t. w is zero for minimum



Hence



M w basis = functions, ³Φ> Φ N + data λpointsI´−1



=

Φ> y

IMAGE

FORMULA

Mx1 MxM MxN Nx1

• This shows that there is a unique solution.

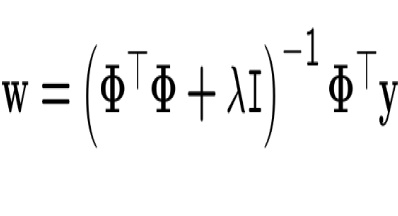
• If λ = 0 (no regularization), then w = (Φ>Φ )−1Φ>y = Φ+y

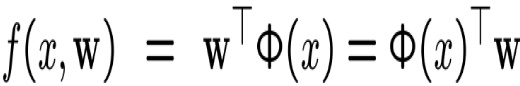
where Φ+ is the pseudo-inverse of Φ (pinv in Matlab)

• Adding the term λI improves the conditioning of the inverse, since if Φ is not full rank, then (Φ>Φ + λI) will be (for suﬃciently large λ)

• As λ → ∞, w → 1 λΦ >y → 0

• Often the regularization is applied only to the inhomogeneous part of w, i.e. to w˜ , where w = (w0, w˜)





= Φ(x)>³ Φ >Φ + λI´−1 Φ>y = b(x)>y

Output is a linear blend, b(x), of the training values {yi}

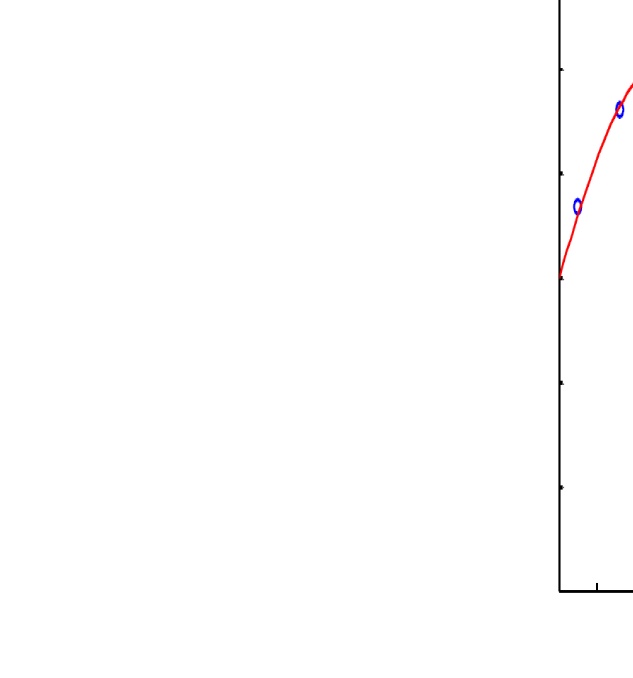
Example 1: polynomial basis functions

ideal fit

• The red curve is the true function (which is not a polynomial)

1.5

Sample points Ideal fit



• The data points are samples from thecurve with added noise in y.

• There is a choice in both the degree, M, ofthe basis functions used, and in the strengthof the regularization

1

0.5

y

0

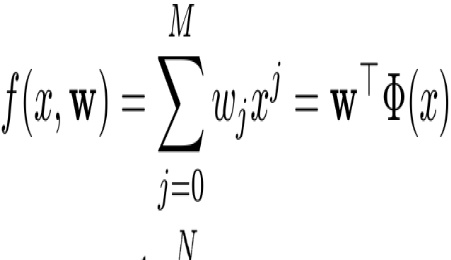
-0.5

-1

-1.5

0

0.1



0.2

0.3

0.4

0.5 x

0.6

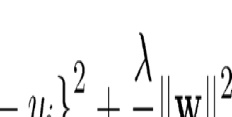
0.7

0.8

0.9

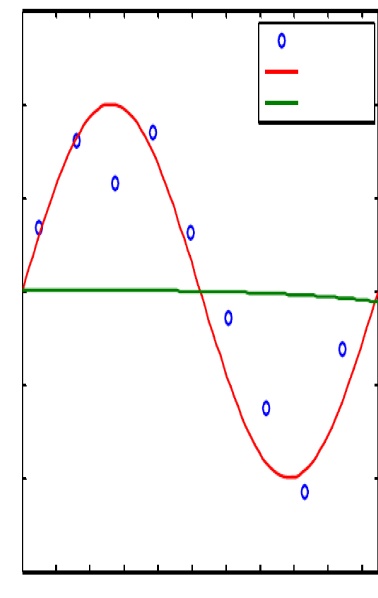
Φ : x → Φ(x) R → RM+1

1



w is a M+1 dimensional vector

N = 9 samples, M = 7



0.5 x

1.5

1

Sample pointsIdeal fit

lambda = 100

0.5

0

-0.5

-1

-1.5

0

0.1

0.2

0.3

0.4

0.6

0.7

0.8

0.9

1

y

1.5



1

0.5

0

-0.5

-1

-1.5

0

0.1

0.2

0.3

0.4

0.5 x

Sample points Ideal fit lambda = 0.001

y

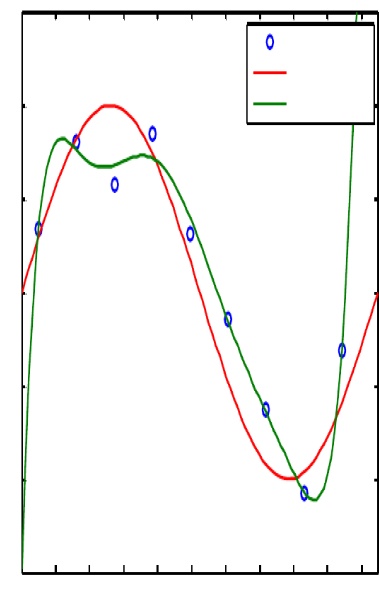
1

0.9

0.8

0.7

0.6



0.5x

1.5

1

Sample pointsIdeal fitlambda = 1e-010

0.5

0

-0.5

-1

-1.5

0

0.1

0.2

0.3

0.4

0.6

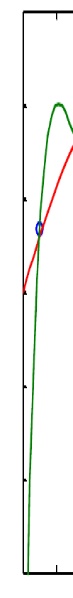
0.7

0.8

0.9

1

y



1.5

1

0.5

0

-0.5

-1

-1.5

0

0.1

0.2

0.3

0.4

0.5 x

Sample points Ideal fit lambda = 1e-015

y

0.6

0.7

0.8

0.9

1

M=3

M=5

least-squares fit

least-squares fit

1.5

1.5

Sample points Ideal fit

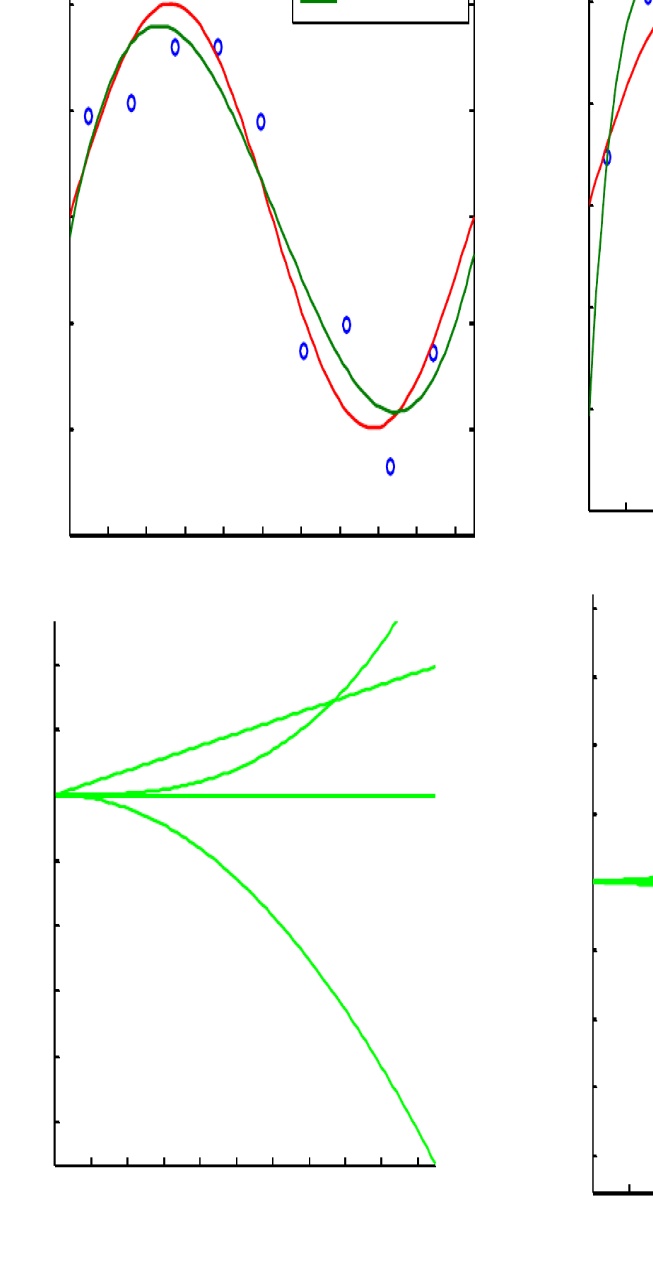
Sample points Ideal fit

Least-squares solution

1

Least-squares solution

1



0.2

0.3

0.2

0.3

0.4

0.5x

0.6

0.5

0.5

0

y

y

0

-0.5

-0.5

-1

-1

-1.5

15

10

5

0

-5

-10

-15

-20

-25

0

Polynomial basis functions

0.3 0.4 0.5 0.6

x

-1.5

0

400

300

200

100

0

-100

-200

-300

-400

0

0.1

0

0.1

0.2

0.7

0.8

0.9

1

y

y

0.1

0.2

0.3

0.4

0.5x

0.6

0.7

0.8

0.9

1

0.1

0.4 0.5 0.6 0.7 Polynomial basisxfunctions

0.7

0.8

0.8

0.9

0.9

1

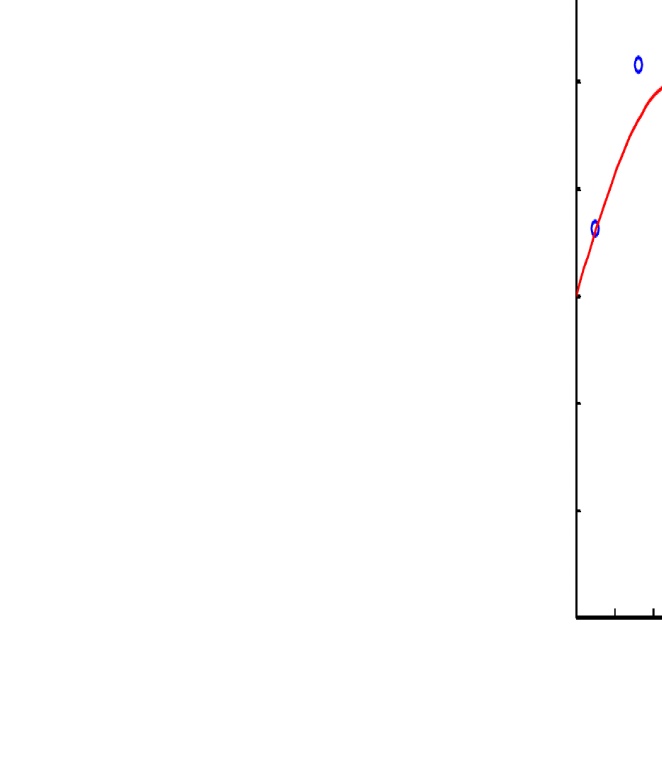
1

Example 2: Gaussian basis functions

• The red curve is the true function (which is

ideal fit

1.5



1

0.5

• Basis functions are centred on the trainingdata (N points)

• There is a choice in both the scale, sigma,of the basis functions used, and in thestrength of the regularization

y

0

-0.5

-1

-1.5

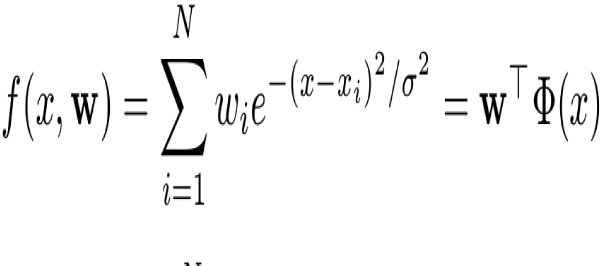
0

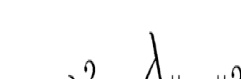
0.1

0.2

not a polynomial)

• The data points are samples from thecurve with added noise in y.





Sample points Ideal fit

0.3

0.4

0.5 x

0.6

0.7

0.8

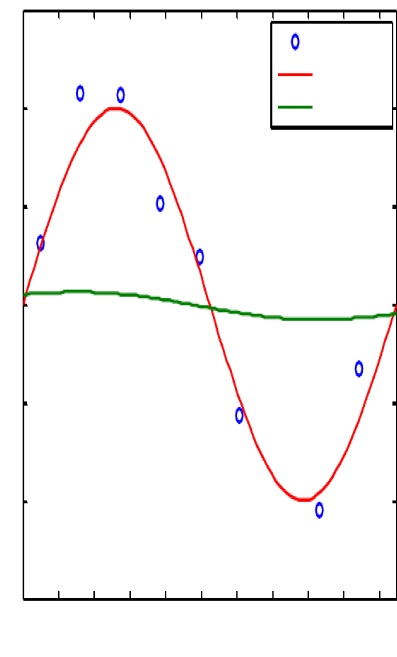
0.9

1

Φ : x → Φ(x) R → RN

w is a N-vector

N = 9 samples, sigma = 0.334



1.5

1

Sample pointsIdeal fit

lambda = 100

0.5

0

-0.5

-1

-1.5

0

0.1

0.2

0.3

0.4

0.5x

0.6

0.7

0.8

0.9

1

y

1.5

1



0.2

0.5

0

-0.5

-1

-1.5

0

0.1

0.3

0.4

0.5 x

0.6

0.7

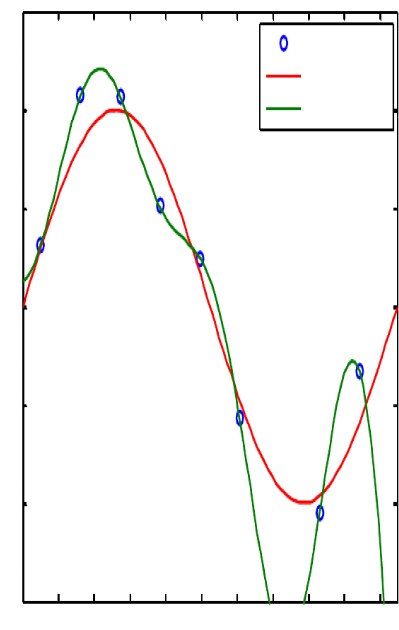
Sample points Ideal fit lambda = 0.001

0.8

0.9

1

y



0.5x

1.5

1

Sample pointsIdeal fitlambda = 1e-010

0.5

0

-0.5

-1

-1.5

0

0.1

0.2

0.3

0.4

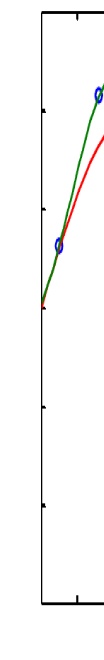
0.6

0.7

0.8

0.9

1



1.5

1

0.5

0

-0.5

y

-1

-1.5 0

0.1

Sample points Ideal fit lambda = 1e-015

y

0.2

0.3

0.4

0.5 x

0.6

0.7

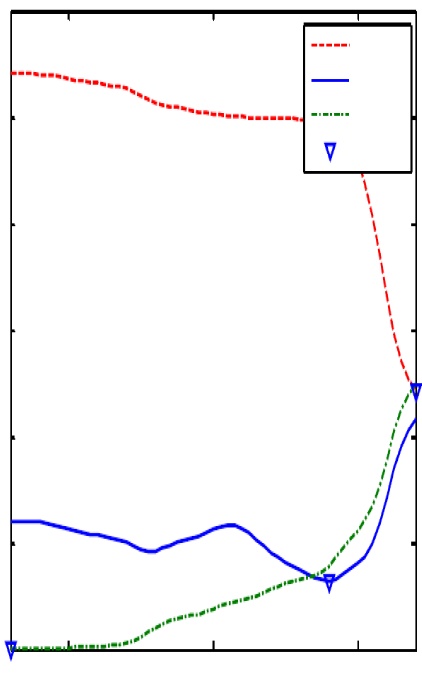
0.8

0.9

1

Choosing lambda using a validation set

IMAGE



6

5

Ideal fitValidationTrainingMin error

4

3

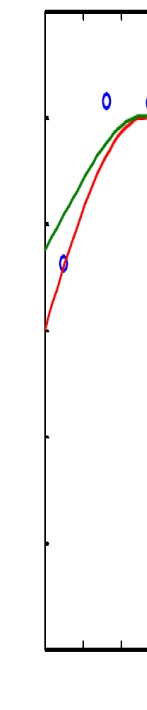
2

1

0



log 



1.5

1

0.5

y

0

-0.5

-1

-1.5

0

0.1

0.2

Sample points Ideal fit Validation set fit

error norm

0.3

0.4

0.5 x

0.6

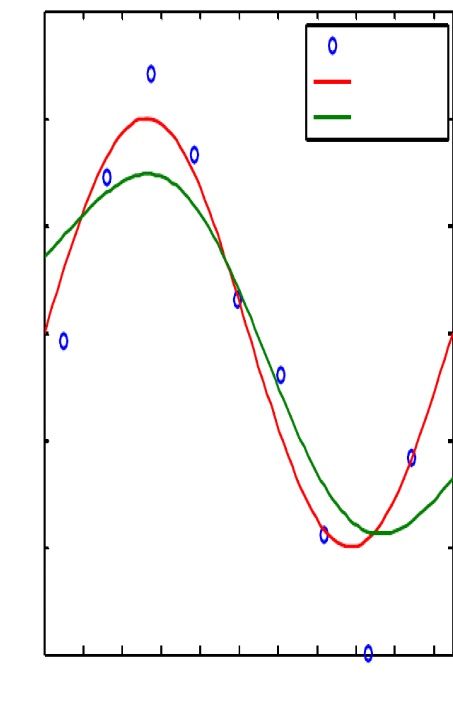
0.7

0.8

0.9

1

FORMULA



1.5

1

Sample pointsIdeal fitValidation set fit

0.5

0

y

-0.5

-1

-1.5 0

0.1

0.2

0.7

0.8

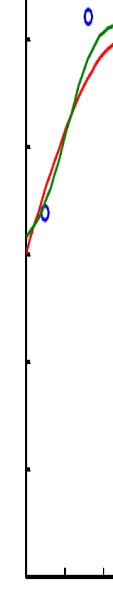
0.9

1

0.3 0.4 0.5 0.6x

y

1.5



1

0.5

0

-0.5

-1

-1.5

0

0.1

0.2

FORMULA

0.4 0.5 0.6 0.7 x

Gaussian basis functions

Sample points Ideal fit Validation set fit

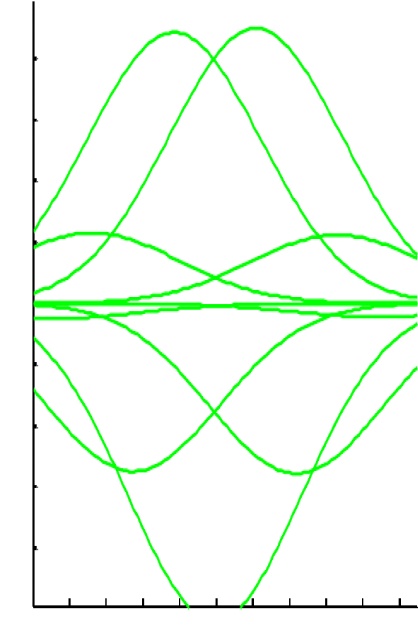
0.3

0.8

0.9

1

Gaussian basis functions



0.5x

2000

1500

1000

500

0

-500

-1000

-1500

-2000

0

0.1

0.2

0.3

0.4

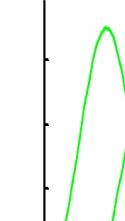
0.6

0.7

0.8

0.9

1



0.8

0.6

0.4

0.2

0

-0.2

-0.4

-0.6

-0.8

y

0

0.1

y

1

0.9

0.8

0.7

0.6

0.5 x

0.4

0.3

0.2

Application: regressing face pose

IMAGE

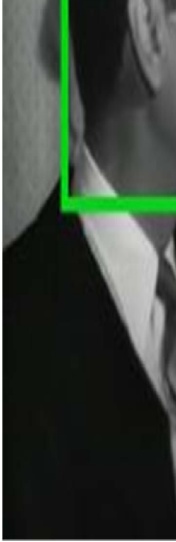
• Estimate two face pose angles:

• yaw (around the Y axis)

• pitch (around the X axis)

• Compute a HOG feature vector for each face region

• Learn a regressor from the HOG vector to the two pose angles



Summary and dual problem

IMAGE

So far we have considered the primal problem where XM f(x, w) = wiφi(x) = w>Φ(x) i=1

and we wanted a solution for w ∈ RM

As in the case of SVMs, we can also consider the dual problem where XN XN w= aiΦ(xi) and f(x, a) = aiΦ(xi)>Φ(x) i=1 i

and obtain a solution for a ∈ RN .

Again

• there is a closed form solution for a,

• the solution involves the N × N Gram matrix k(xi, xj) = Φ(xi)>Φ(xj),

• so we can use the kernel trick again to replace scalar products

Background reading and more

IMAGE

• Bishop, chapters 6 & 7 for kernels and SVMs

• Hastie et al, chapter 12

• Bishop, chapter 3 for regression

• More on web page:

http://www.robots.ox.ac.uk/~az/lectures/ml