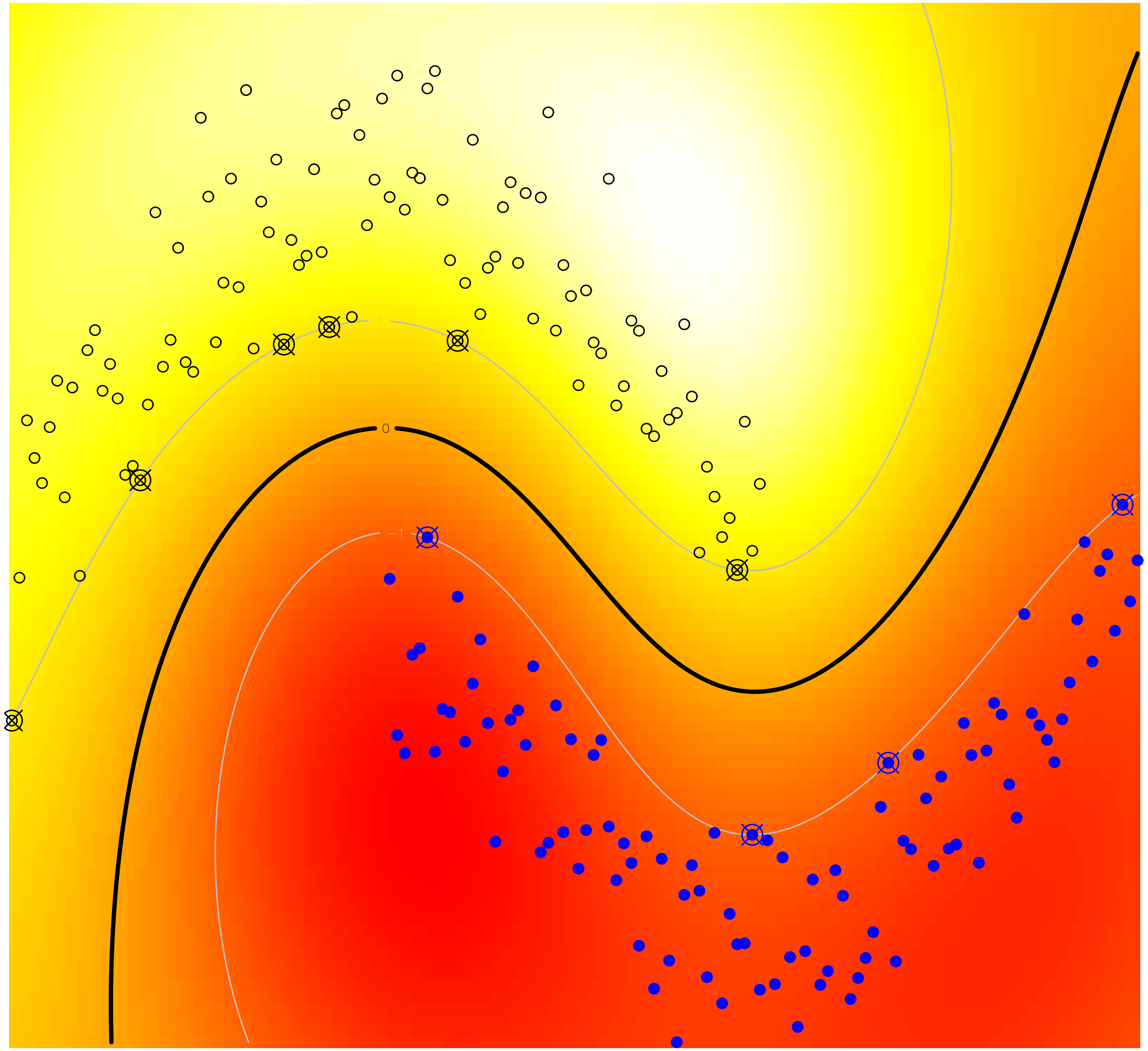
Support Vector Machine (SVM)



First papers on SVM

A Training Algorithm for Optimal Margin Classifiers, Bernhard E. Boser and Isabelle M. Guyon and Vladimir Vapnik, Proceedings of the fifth annual workshop on Computational learning theory (COLT), 1992
http://www.clopinet.com/isabelle/Papers/colt92.ps.Z

Support-Vector Networks, Corinna Cortes and Vladimir Vapnik, Machine Learning, 20(3):273-297, 1995
http://homepage.mac.com/corinnacortes/papers/SVM.ps

Extracting Support Data for a Given Task, Bernhard Schölkopf and Christopher J.C. Burges and Vladimir Vapnik, First International Conference on Knowledge Discovery & Data Mining, 1995
http://research.microsoft.com/~cburges/papers/kdd95.ps.gz

Literature (Papers, Bookchapter, T.Report)

Statistical Learning and Kernel Method, Bernhard Schölkopf, Microsoft Research Technical Report, 2000, ftp://ftp.research.microsoft.com/pub/tr/tr-2000-23.pdf

An Introduction to Kernel-based Learning Algorithms, K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf, IEEE Neural Networks, 12(2):181-201, 2001, http://ieeexplore.ieee.org/iel5/72/19749/00914517.pdf

Support Vector Machines, S. Mika, C. Schäfer, P. Laskov, D. Tax and K.-R. Müller, Bookchapter : Handbook of Computational Statistics (Concepts and Methods), http://www.xplore-stat.de/ebooks/scripts/csa/html/

Support Vector Machines for Classification and Regression, S.R. Gunn, Technical Report, http://www.ecs.soton.ac.uk/~srg/publications/pdf/SVM.pdf

Literature (Paper, Books)

A Tutorial on Support Vector Machines for Pattern Recognition, Christopher J.C. Burges, Kluwer Academic Publishers, Boston
http://research.microsoft.com/~cburges/papers/SVMTutorial.pdf

Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, Bernhard Schölkopf and Alexander J. Smola, MIT Press, 2001

An Introduction to Support Vector Machines and Other Kernel-based Learning Methods, Nello Cristianini and John Shawe-Taylor, Cambridge University Press, 2000

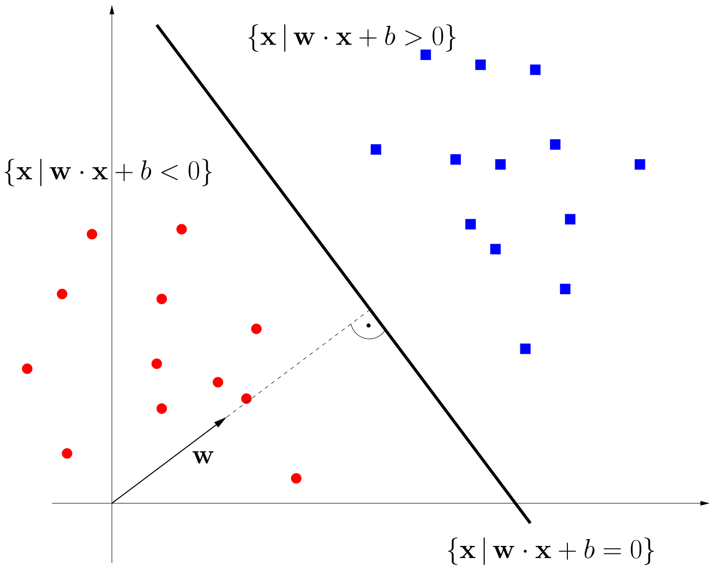
Kernel Methods for Pattern Analysis, John Shawe-Taylor and Nello Cristianini, Cambridge University Press, 2004

The Nature of Statistical Learning Theory, Vladimir N. Vapnik, Springer-Verlag, sec. edition, 1999

Literature (WWW and Source Code)

- http://www.kernel-machines.org/
- http://www.support-vector.net/
- http://www.learning-with-kernels.org/
- http://www.pascal-network.org/
- http://www.r-project.org/ (packages e1071,kernlab)

Scaling Freedom (Problem)



Multiplying $\mathbf{x}$ and b by the same constant κ gives the same hyperplane, represented in terms of different parameters

$$\kappa[(\mathbf{w}\cdot\mathbf{x})+b]=0 \Leftrightarrow (\kappa\mathbf{w}\cdot\mathbf{x}+\kappa b)=0 \Leftrightarrow (\mathbf{w}'\cdot\mathbf{x})+b'=0.$$

Example: $\mathbf{w} := (5,-4)^T, b := 2$;
$5x-4y+2=0 \Leftrightarrow y=\frac{5}{4}x+\frac{1}{2}, \qquad \kappa := 3;$
$15x-12y+6=0 \Leftrightarrow y=\frac{5}{4}x+\frac{1}{2}$

Canonical Hyperplane

A *canonical* hyperplane with respect to given m examples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m) \in \mathcal{X} \times \{\pm 1\}$ is defined as a function

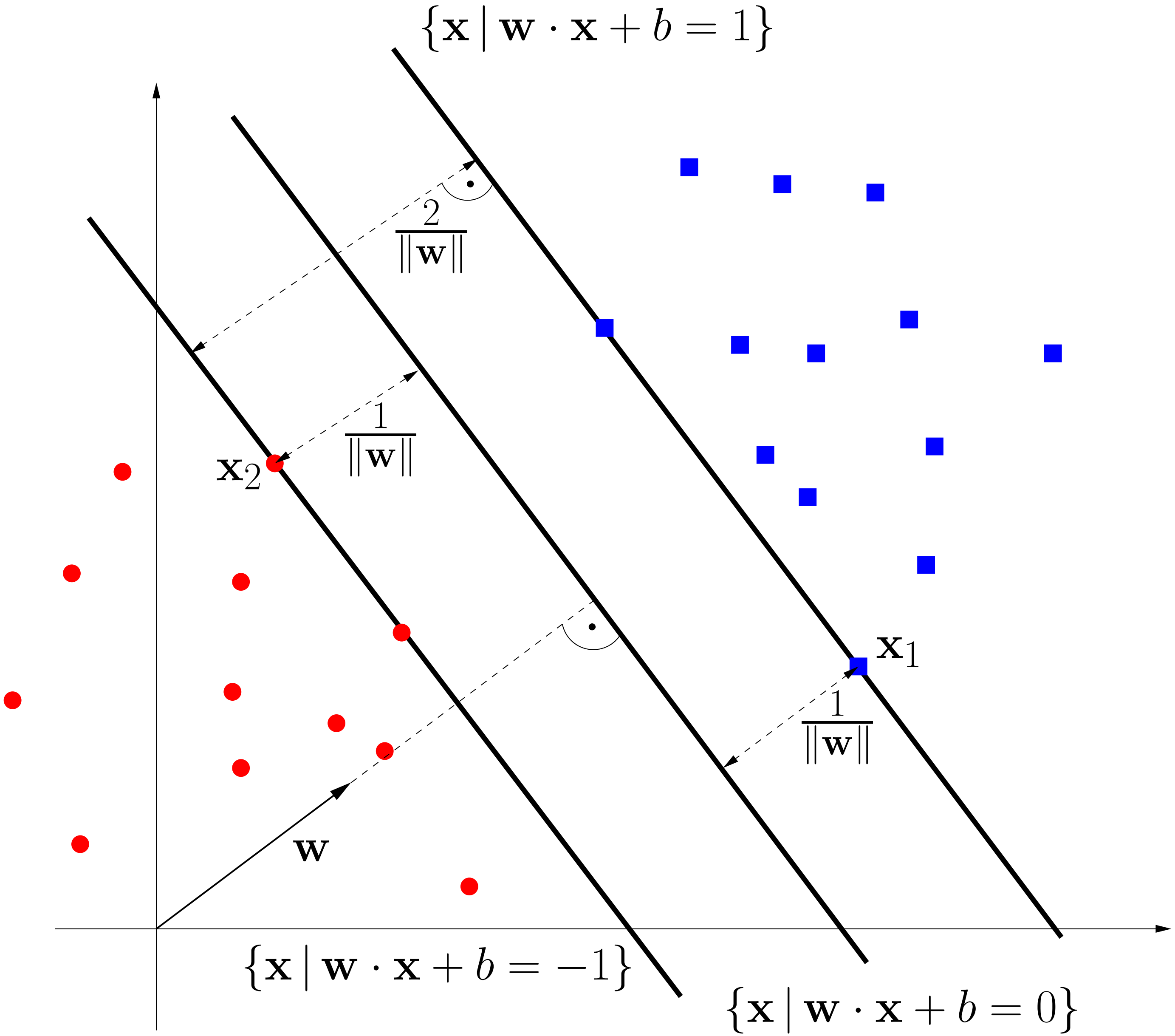
$$f(\mathbf{x}) = (\mathbf{w} \cdot \mathbf{x}) + b$$

where $\mathbf{w}$ is normalized such that

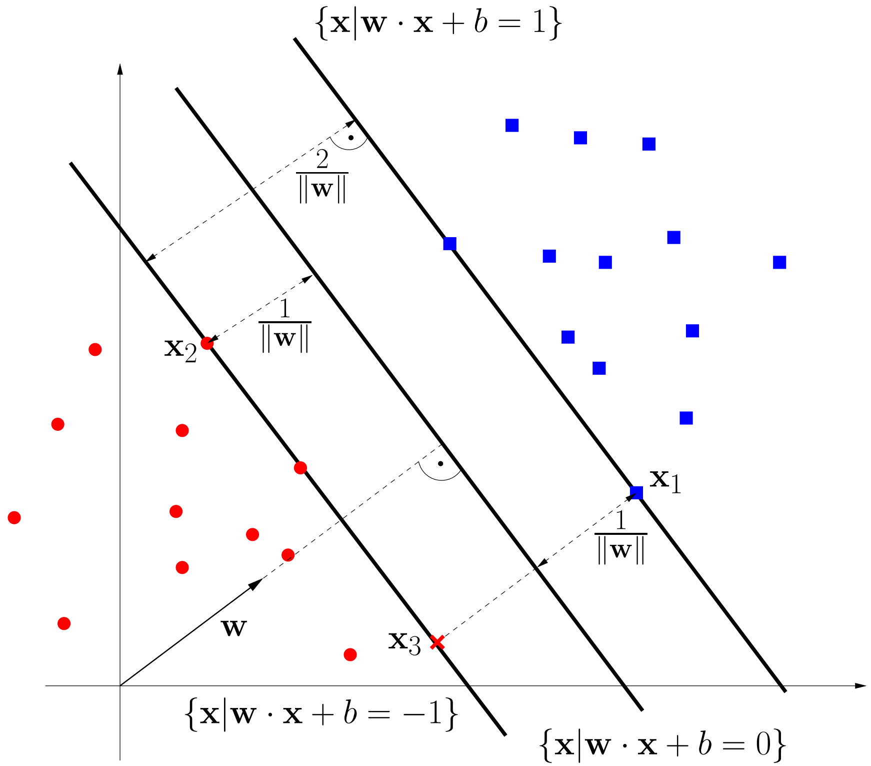
$$\min_{i=1,\ldots,m} |f(\mathbf{x}_i)| = 1$$

i.e. scaling $\mathbf{w}$ and b such that the points closest to the hyperplane satisfy $|(\mathbf{w} \cdot \mathbf{x}_i) + b| = 1$

Plus/Minus and Zero Hyperplane



Optimal Separating Hyperplane



Let $\mathbf{x}_3$ be any point on the "minus" hyperplane and let $\mathbf{x}_1$ be the closest point to $\mathbf{x}_3$.

$$\mathbf{w}\cdot\mathbf{x}_1+b=+1$$
$$\mathbf{w}\cdot\mathbf{x}_3+b=-1$$
$$\mathbf{x}_1=\mathbf{x}_3+\lambda\mathbf{w}$$
$$\mathbf{w}\cdot\underbrace{(\mathbf{x}_3+\lambda\mathbf{w})}_{\mathbf{x}_1}+b=1$$

$\underbrace{\mathbf{w}\cdot\mathbf{x}_3+b}_{-1}+\lambda\|\mathbf{w}\|^2=1\Rightarrow\lambda=\frac{2}{\|\mathbf{w}\|^2}$. We are interested in margin, i.e. $\|\mathbf{x}_1-\mathbf{x}_3\|=\|\lambda\mathbf{w}\|=\lambda\|\mathbf{w}\|=\frac{2}{\|\mathbf{w}\|^2}\|\mathbf{w}\|=\frac{2}{\|\mathbf{w}\|}$

Formulation as an Optimization Problem

Hyperplane with maximum margin (the smaller the norm of the weight vector $\mathbf{w}$, the larger the margin):

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{subject to} & y_i\left((\mathbf{w}\cdot\mathbf{x}_i)+b\right) \geq 1, \quad i=1,\ldots,m \end{array}$$

Introduce Lagrange multipliers $\alpha_i \geq 0$ and a Lagrangian

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i (y_i((\mathbf{x}_i \cdot \mathbf{w}) + b) - 1)$$

At the extrema, we have

$$\frac{\partial}{\partial b} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0, \quad \frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0$$

Optimization and Kuhn-Tucker Theorem

leads to solution

$$\sum_{i=1}^{m} \alpha_i y_i = 0, \qquad \mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i.$$

The extreme point solution obtained has an important property that results from optimization known as the Kuhn-Tucker theorem. The theorem says:

Lagrange multiplier can be non-zero only if the corresponding inequality constraint is an equality at the solution.

This implies that *only* the training examples $\mathbf{x}_i$ that lie on the plus and minus hyperplane have their corresponding α_i non-zero.

Relevant/Irrelevant Support Vector

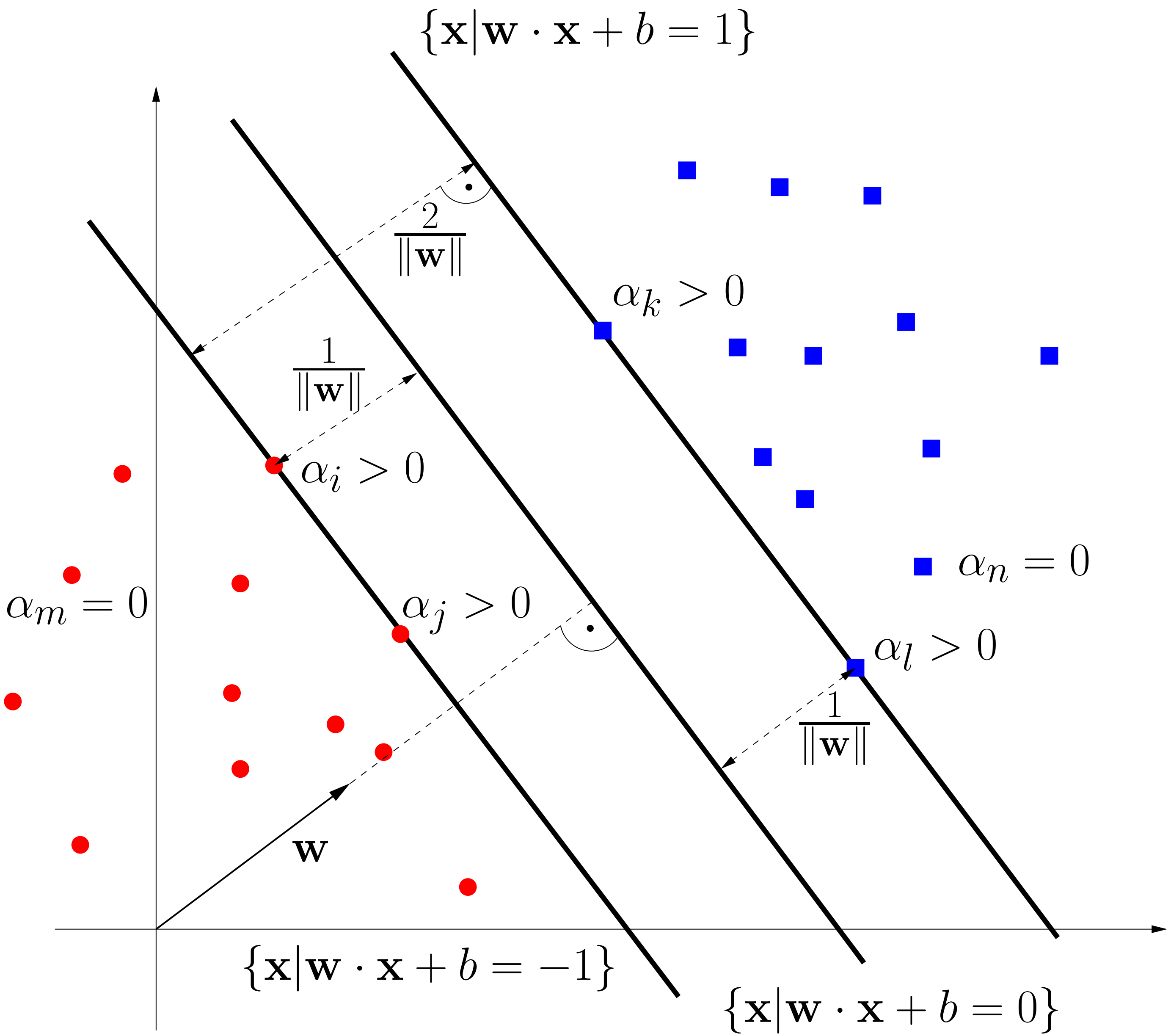
More formally, the Karush-Kuhn-Tucker complementarity conditions say:

$$\alpha_i \left[y_i((\mathbf{x}_i \cdot \mathbf{w}) + b) - 1 \right] = 0, \quad i = 1, \ldots, m$$

the Support Vectors lie on the margin. That means for all training points

$$\begin{array}{lcl} [y_i((\mathbf{x}_i \cdot \mathbf{w}) + b)] > 1 & \Rightarrow \alpha_i = 0 & \rightarrow \mathbf{x}_i \text{ irrelevant} \\ [y_i((\mathbf{x}_i \cdot \mathbf{w}) + b)] = 1 & \text{(on margin)} & \rightarrow \mathbf{x}_i \text{ Support Vector} \end{array}$$

Relevant/Irrelevant Support Vector



Dual Form

The dual form has many advantages

- Formulate optimization problem without $\mathbf{w}$ (mapping $\mathbf{w}$ in high-dimensional spaces).
- Formulate optimization problem by means of α, y_i and dot product $\mathbf{x}_i \cdot \mathbf{x}_j$.
- Quadratic Programming Solver.

maximize $\quad W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j)$

subject to $\quad \alpha_i \geq 0, \quad i = 1, \ldots, m$ and $\sum_{i=1}^{m} \alpha_i y_i = 0$

Hyperplane Decision Function

The solution is determined by the examples on the margin.
Thus

$$
\begin{aligned}
f(\mathbf{x}) &= \operatorname{sgn}\left((\mathbf{x}\cdot\mathbf{w}) + b\right) \\
&= \operatorname{sgn}\left(\sum_{i=1}^{m} y_i\alpha_i\left(\mathbf{x}\cdot\mathbf{x}_i\right) + b\right)
\end{aligned}
$$

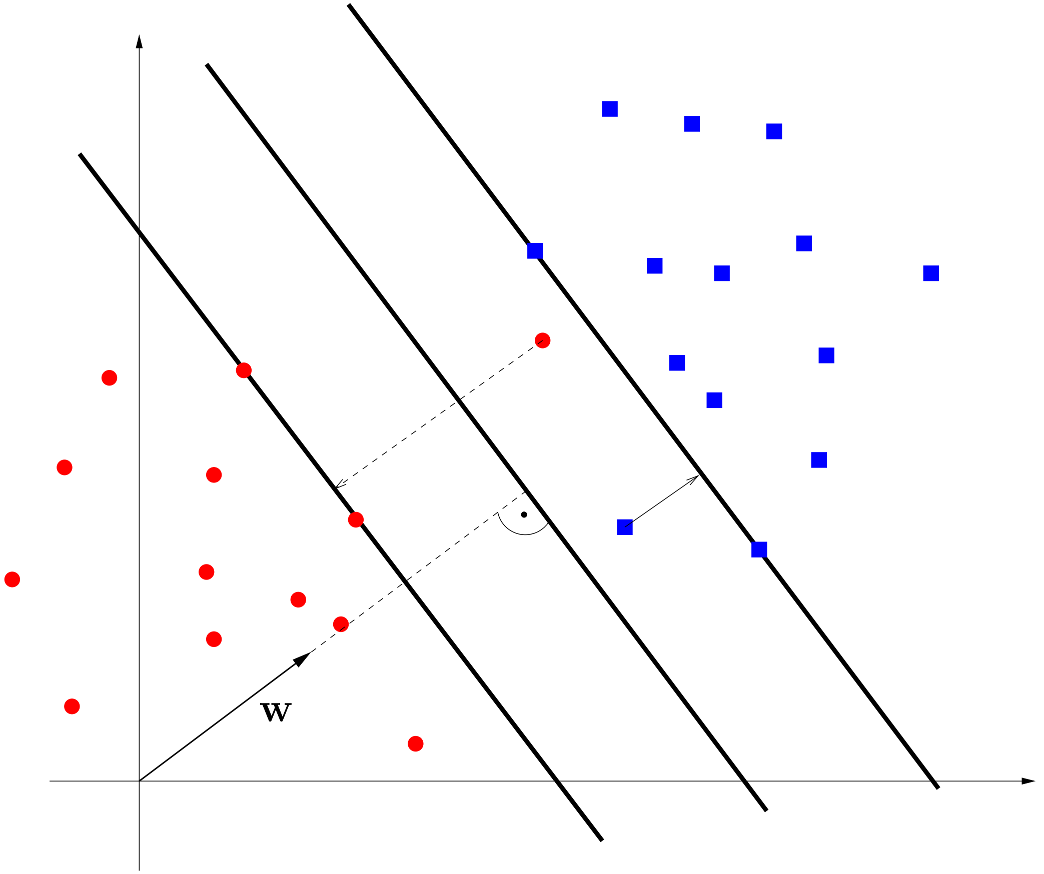
where

$$\alpha_i\left[y_i((\mathbf{x}_i\cdot\mathbf{w}) + b) - 1\right] = 0, \quad i = 1,\ldots,m$$

and

$$\mathbf{w} = \sum_{i=1}^{m}\alpha_i y_i \mathbf{x}_i$$

Non-separable Case



Case where the constraints $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$ cannot be satisfied, i.e. $\alpha_i \rightarrow \infty$

Relax Constraints (Soft Margin)

Modify the constraints to

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \text{ with } \xi_i \geq 0$$

and add

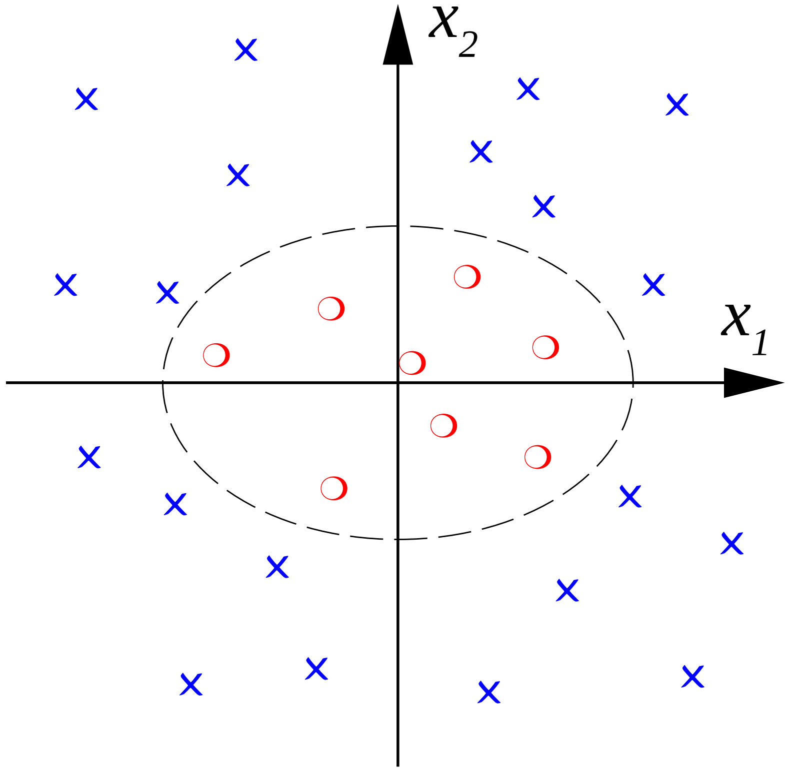
$$C \cdot \sum_{i=1}^{m} \xi_i$$

i.e. distance of error points to their correct place in the objective function

$$\text{minimize} \qquad \frac{1}{2}\|\mathbf{w}\|^2 + C \cdot \sum_{i=1}^{m} \xi_i$$

Same dual, with additional constraints $0 \leq \alpha_i \leq C$

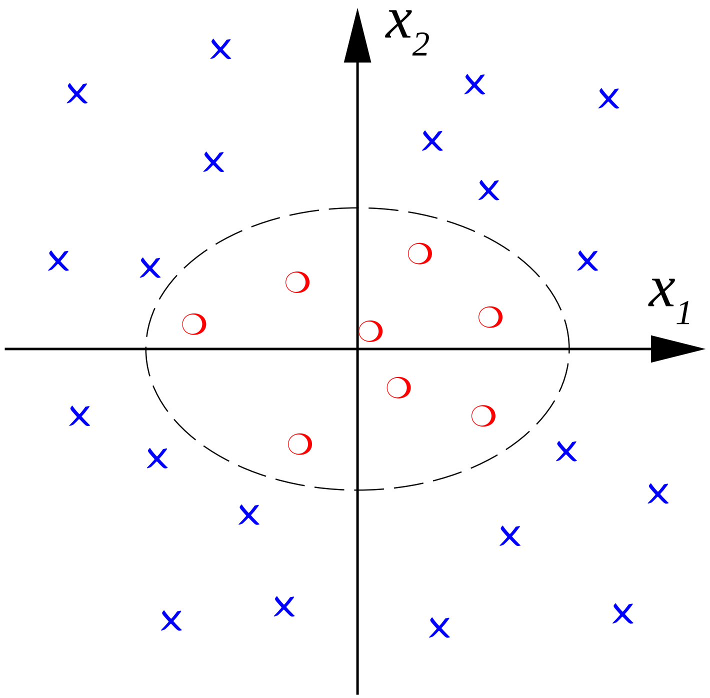
Non-separable Case (Part 2)

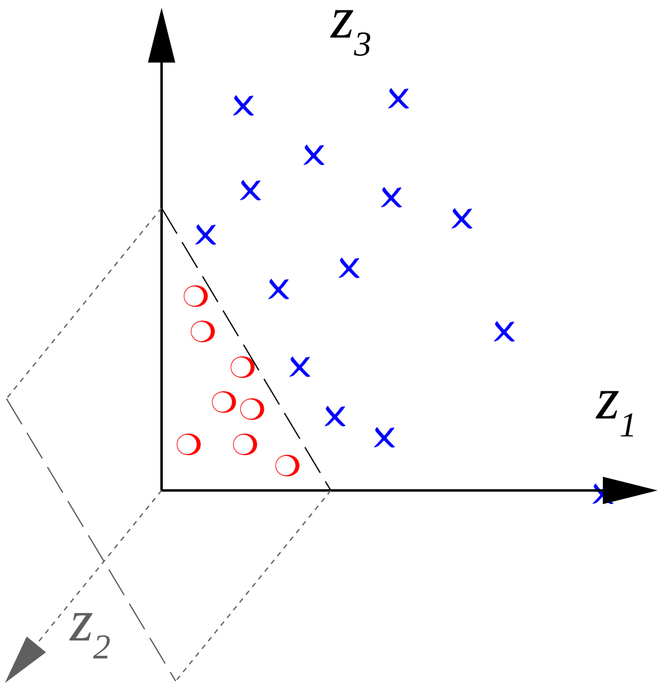


This data set is not properly separable with lines (also when using many slack variables)

Separate in Higher-Dim. Space

Map data in higher-dimensional space and separate it there with a hyperplane





$$\begin{aligned} \Phi : \mathbb{R}^2 &\rightarrow \mathbb{R}^3 \\ (x_1, x_2) &\mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}x_1x_2, x_2^2) \end{aligned}$$

Feature Space

Apply the mapping

$$\begin{aligned} \Phi : \mathbb{R}^N &\rightarrow \mathcal{F} \\ \mathbf{x} &\mapsto \Phi(\mathbf{x}) \end{aligned}$$

to the data $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m \in \mathcal{X}$ and construct separating hyperplane in $\mathcal{F}$ instead of $\mathcal{X}$. The samples are preprocessed as $(\Phi(\mathbf{x}_1), y_1), \ldots, (\Phi(\mathbf{x}_m), y_m) \in \mathcal{F} \times \{\pm 1\}$.

Obtained decision function:

$$\begin{aligned} f(\mathbf{x}) &= \operatorname{sgn}\left(\sum_{i=1}^{m} y_i \alpha_i \left(\Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}_i) \right) + b \right) \\ &= \operatorname{sgn}\left(\sum_{i=1}^{m} y_i \alpha_i \, k(\mathbf{x}, \mathbf{x}_i) + b \right) \end{aligned}$$

Kernels

A *kernel* is a function k, such that for all $\mathbf{x}, \mathbf{y} \in \mathcal{X}$

$$k(\mathbf{x}, \mathbf{y}) = (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})),$$

where Φ is a mapping from $\mathcal{X}$ to an dot product feature space $\mathcal{F}$.

The $m \times m$ matrix K with elements $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ is called kernel matrix or Gram matrix. The kernel matrix is symmetric and positive semi-definite, i.e. for all $a_i \in \mathbb{R}$, $i = 1, \ldots, m$, we have

$$\sum_{i,j=1}^{m} a_i a_j K_{ij} \geq 0$$

Positive semi-definite kernels are exactly those giving rise to a positive semi-definite kernel matrix K for all m and all sets $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\} \subseteq \mathcal{X}$.

The Kernel Trick Example

Example : compute 2nd order products of two "pixels", i.e.

$$\mathbf{x} = (x_1, x_2) \text{ and } \Phi(\mathbf{x}) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

$$\begin{aligned}
(\Phi(\mathbf{x}) \cdot \Phi(\mathbf{z})) &= (x_1^2, \sqrt{2}x_1x_2, x_2^2)(z_1^2, \sqrt{2}z_1z_2, z_2^2)^T \\
&= ((x_1, x_2)(z_1, z_2)^T)^2 \\
&= (\mathbf{x} \cdot \mathbf{z}^T)^2 \\
&= : k(\mathbf{x}, \mathbf{z})
\end{aligned}$$

Kernel without knowing Φ

Recall: mapping $\Phi : \mathbb{R}^N \to \mathcal{F}$. SVM depends on the data through dot products in $\mathcal{F}$, i.e. functions of the form

$$\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$

- With k such that $k(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$, it is not necessary to even know what $\Phi(\mathbf{x})$ is.

Example: $k(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{\|\mathbf{u}-\mathbf{v}\|^2}{\gamma}\right)$, in this example $\mathcal{F}$ is infinite dimensional.

Feature Space (Optimization Problem)

Quadratic optimization problem (soft margin) with kernel:

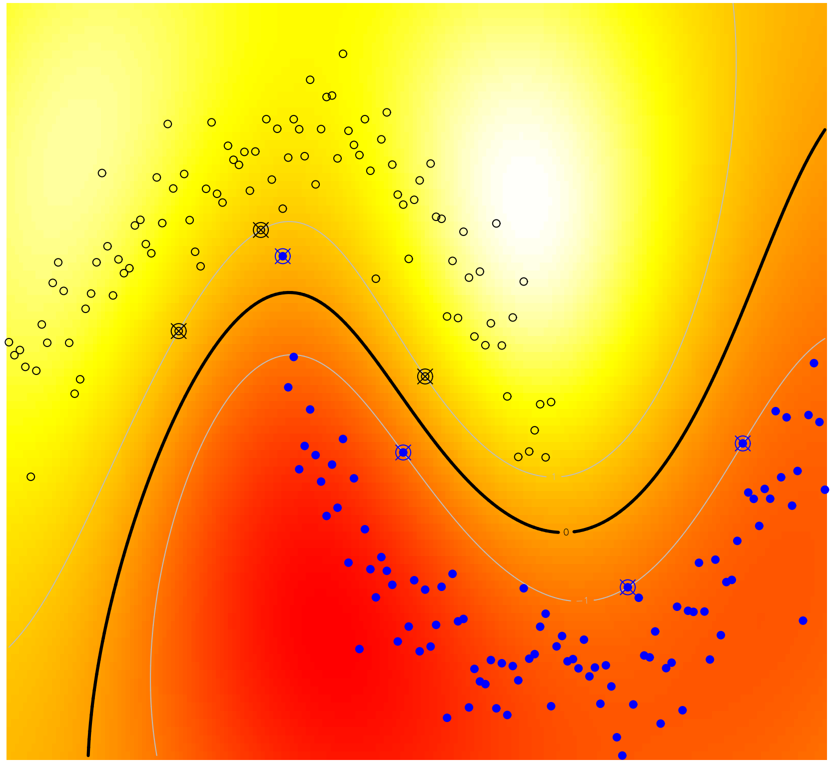
maximize $$W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$$

subject to $$0 \leq \alpha_i \leq C, \quad i = 1, \ldots, m \text{ and } \sum_{i=1}^{m} \alpha_i y_i = 0$$

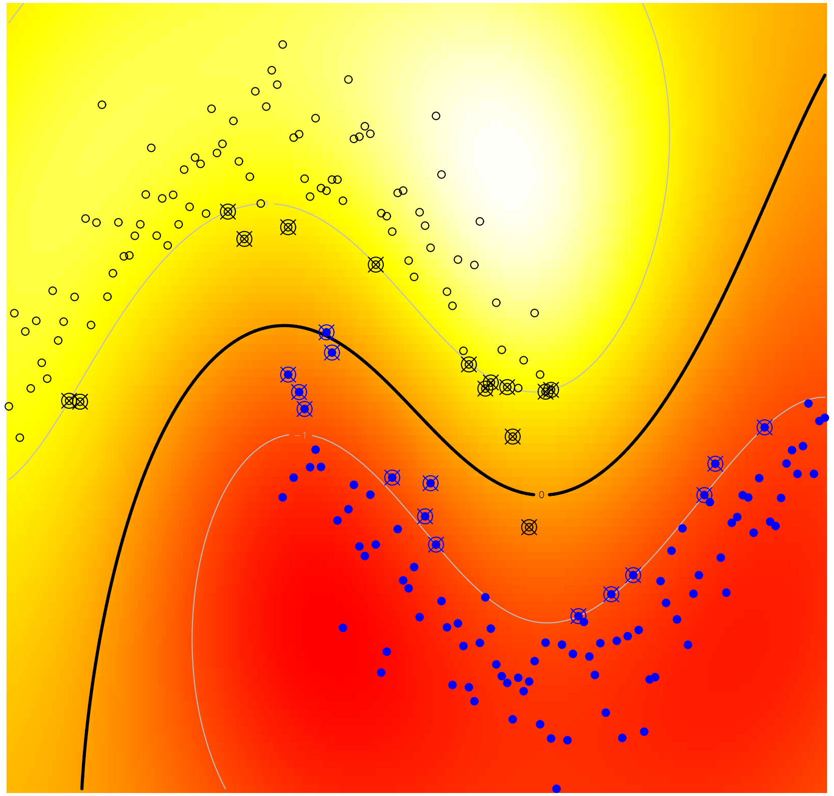
(Standard) Kernels

$$
\begin{aligned}
\text{Linear} \quad k_0(\mathbf{u}, \mathbf{v}) &= (\mathbf{u} \cdot \mathbf{v}) \\
\text{Polynomial} \quad k_1(\mathbf{u}, \mathbf{v}) &= ((\mathbf{u} \cdot \mathbf{v}) + \Theta)^d \\
\text{Gaussian} \quad k_2(\mathbf{u}, \mathbf{v}) &= \exp\left(-\frac{\|\mathbf{u} - \mathbf{v}\|^2}{\gamma}\right) \\
\text{Sigmoidal} \quad k_3(\mathbf{u}, \mathbf{v}) &= \tanh(\kappa(\mathbf{u} \cdot \mathbf{v}) + \Theta)
\end{aligned}
$$

SVM Results for Gaussian Kernel

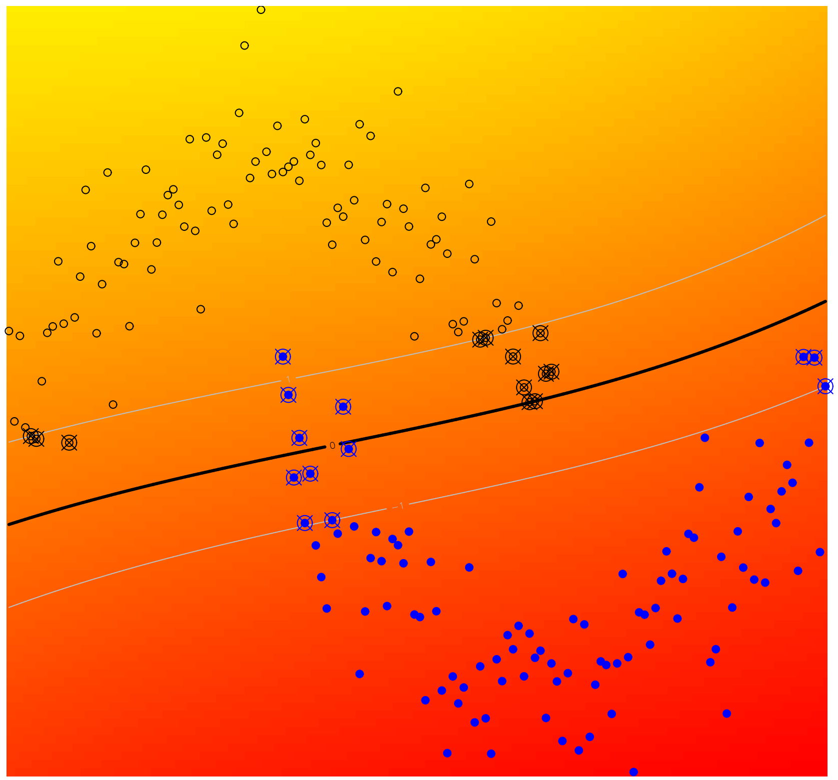


$\gamma = 0.5,\ C = 50$

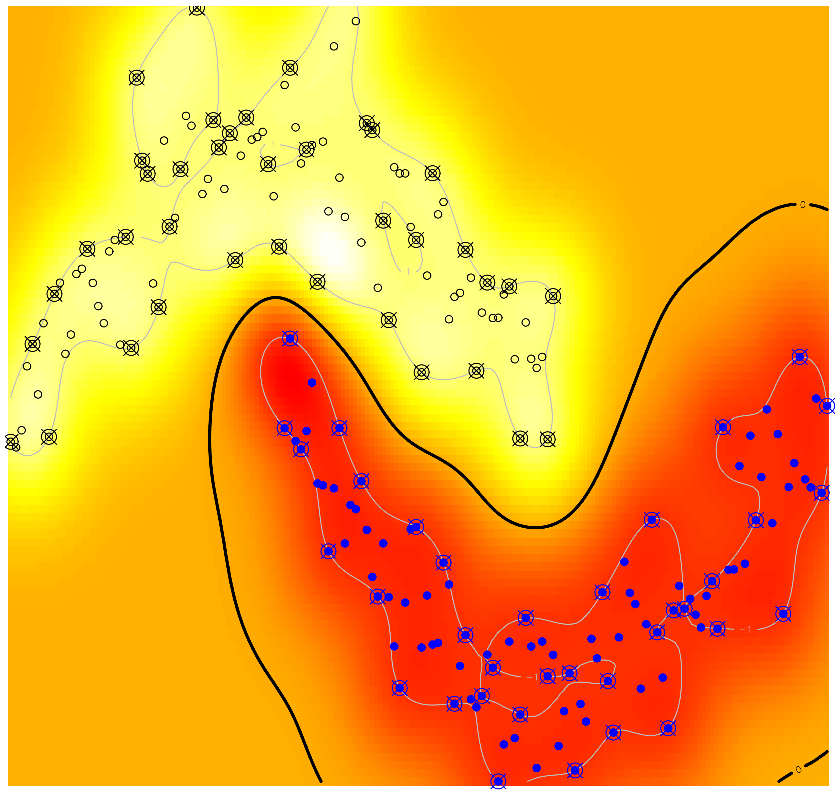


$\gamma = 0.5,\ C = 1$

SVM Results for Gaussian Kernel (cont.)



$\gamma = 0.02,\ C = 50$



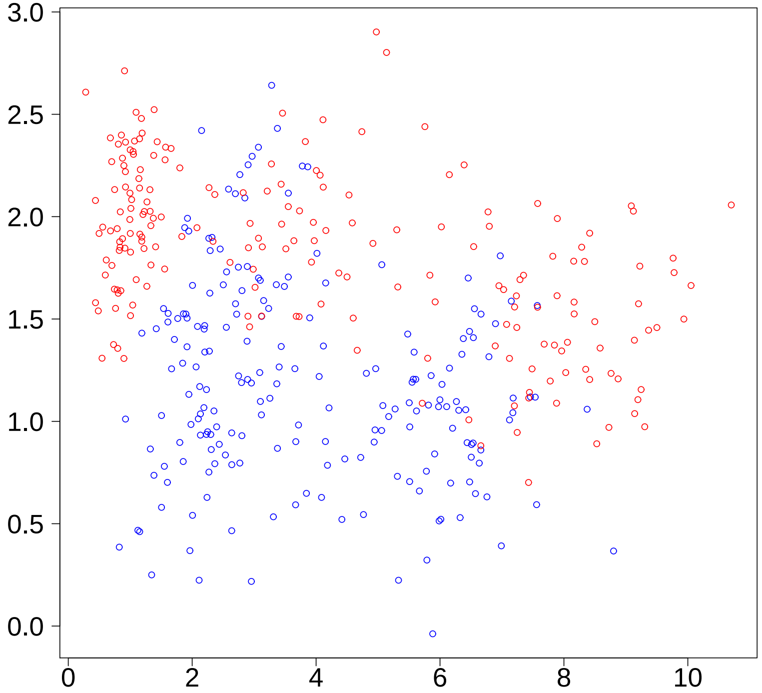
$\gamma = 10,\ C = 50$

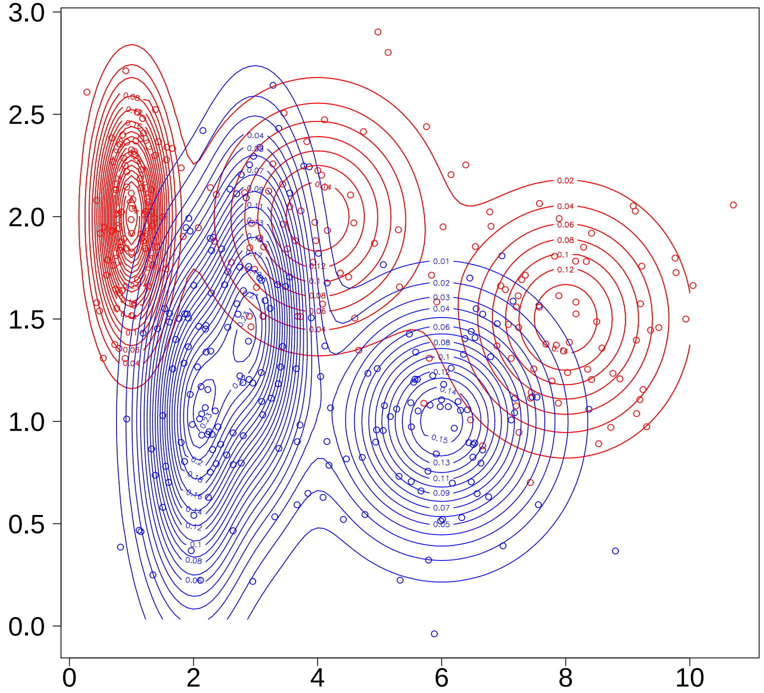
Link to Statistical Learning Theory

Pattern Classification:

Learn $f : \mathcal{X} \to \{\pm 1\}$ from input-output training data

We assume that data is generated from some unknown (but fixed) probability distribution $P(\mathbf{x}, y)$





(Empirical) Risk

Learn f from training set $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m) \in \mathcal{X} \times \{\pm 1\}$, such that the expected missclassification error on a test set, also drawn from $P(\mathbf{x}, y)$,

$$R[f] = \int \frac{1}{2}|f(\mathbf{x}) - y| dP(\mathbf{x}, y) \quad \text{Expected Risk}$$

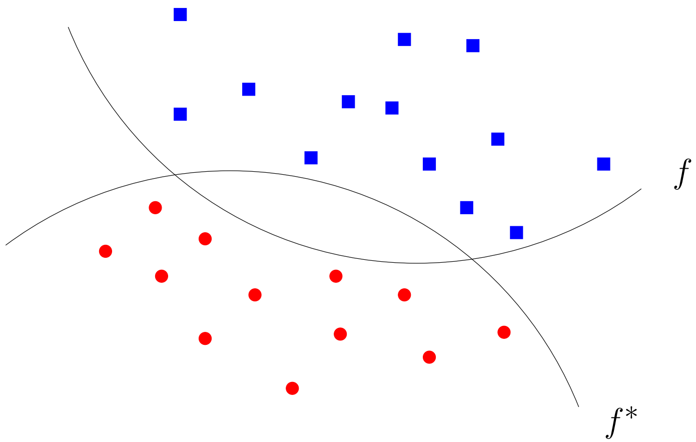
is minimal.

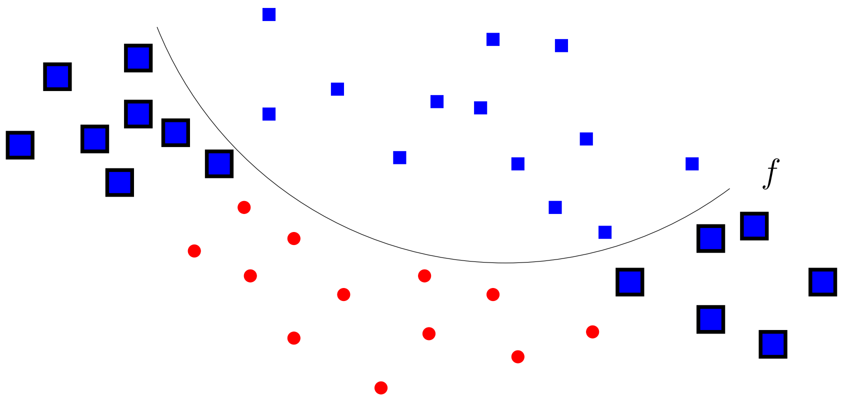
Problem: we cannot minimize the expected risk, because P is unknown. Minimize instead the average risk over training set, i.e.

$$R_{\text{emp}}[f] = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2}|f(\mathbf{x}_i) - y_i| \quad \text{Empirical Risk}$$

Problems with Empirical Risk

Minimizing the empirical risk (training error), does not imply a small test error. To see this, note that for each function f and any test set $(\overline{\mathbf{x}}_1, \overline{y}_1), \ldots, (\overline{\mathbf{x}}_m, \overline{y}_m) \in \mathcal{X} \times \{\pm 1\}$, satisfying $\{\overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_{\overline{m}}\} \cap \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} = \{\}$, there exists another function f^* such that $f^*(\mathbf{x}_i) = f(\mathbf{x}_i)$ for all $i = 1, \ldots, m$, *but* $f^*(\overline{\mathbf{x}}_i) \neq f(\overline{\mathbf{x}}_i)$ (on all test samples) for all $i = 1, \ldots, \overline{m}$

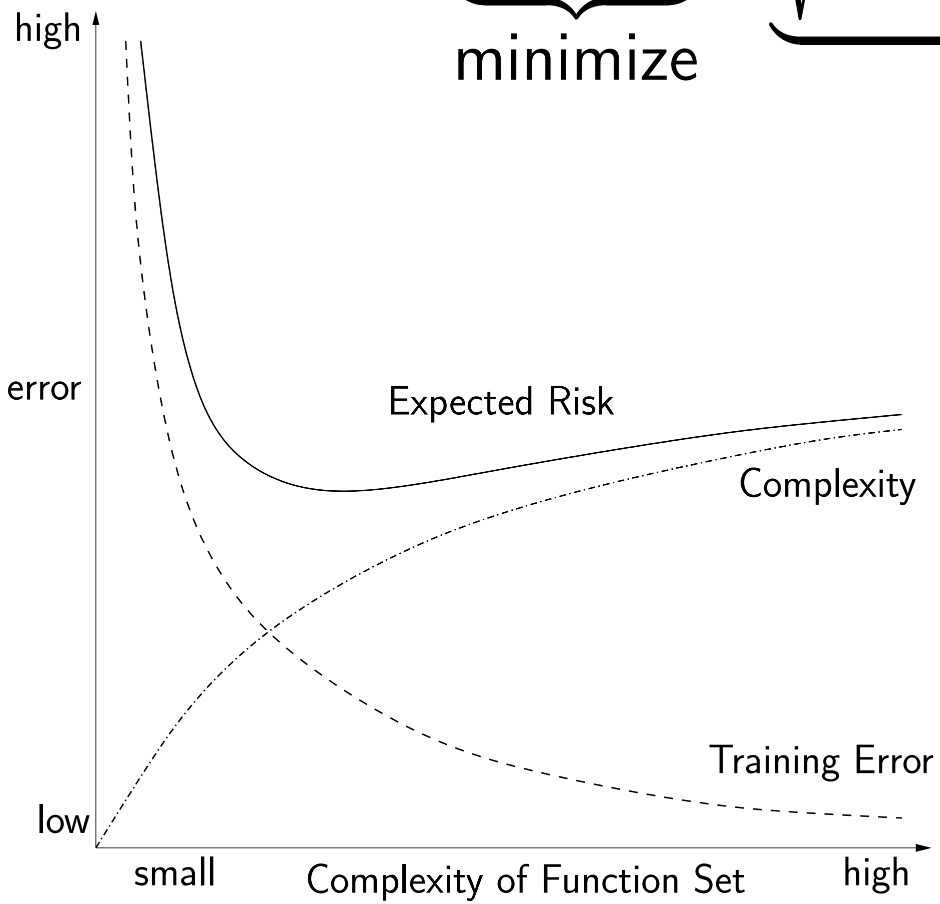




A Bound for Pattern Classification

For any $f \in F$ and $m > h$, with a probability of at least $1 - \eta$,

$$R[f] \leq \underbrace{R_{\text{emp}}[f]}_{\text{minimize}} + \underbrace{\sqrt{\frac{h(\ln \frac{2m}{h} + 1) - \ln(\eta/4)}{m}}}_{\text{minimize}}$$

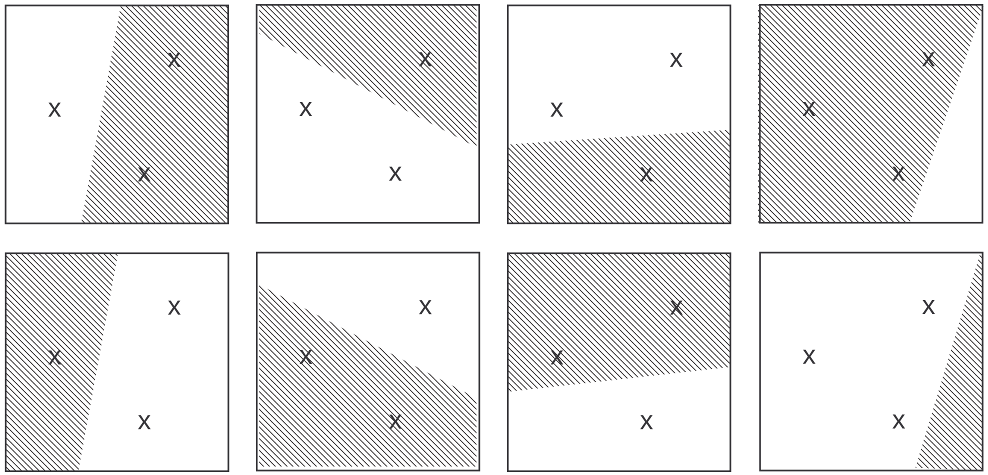


Measure Complexity of Function Set

How to measure the complexity of a given function set ?

The Vapnik-Chervonenkis (short VC) dimension is defined as the maximum number of points that can be labeled in all possible ways.

- In $\mathbb{R}^2$ we can shatter three non-collinear points.
- But we can never shatter four points in $\mathbb{R}^2$.
- Hence the VC dimension is $h = 3$.



VC Dimension

- Separating hyperplanes in $\mathbb{R}^N$ have VC dimension $N + 1$.
- Hence: separating hyperplanes in high-dimensional feature spaces have extremely large VC dimension, and may not generalize well.
- But, "margin" hyperplanes can still have a small VC dimension.

VC Dimension of Margin Hyperplanes

Consider hyperplanes $(\mathbf{w} \cdot \mathbf{x}) = 0$ where w is normalized such they are in a canonical form w.r.t. a set of points $X^* = \{\mathbf{x}_1, \ldots, \mathbf{x}_r\}$, i.e.,

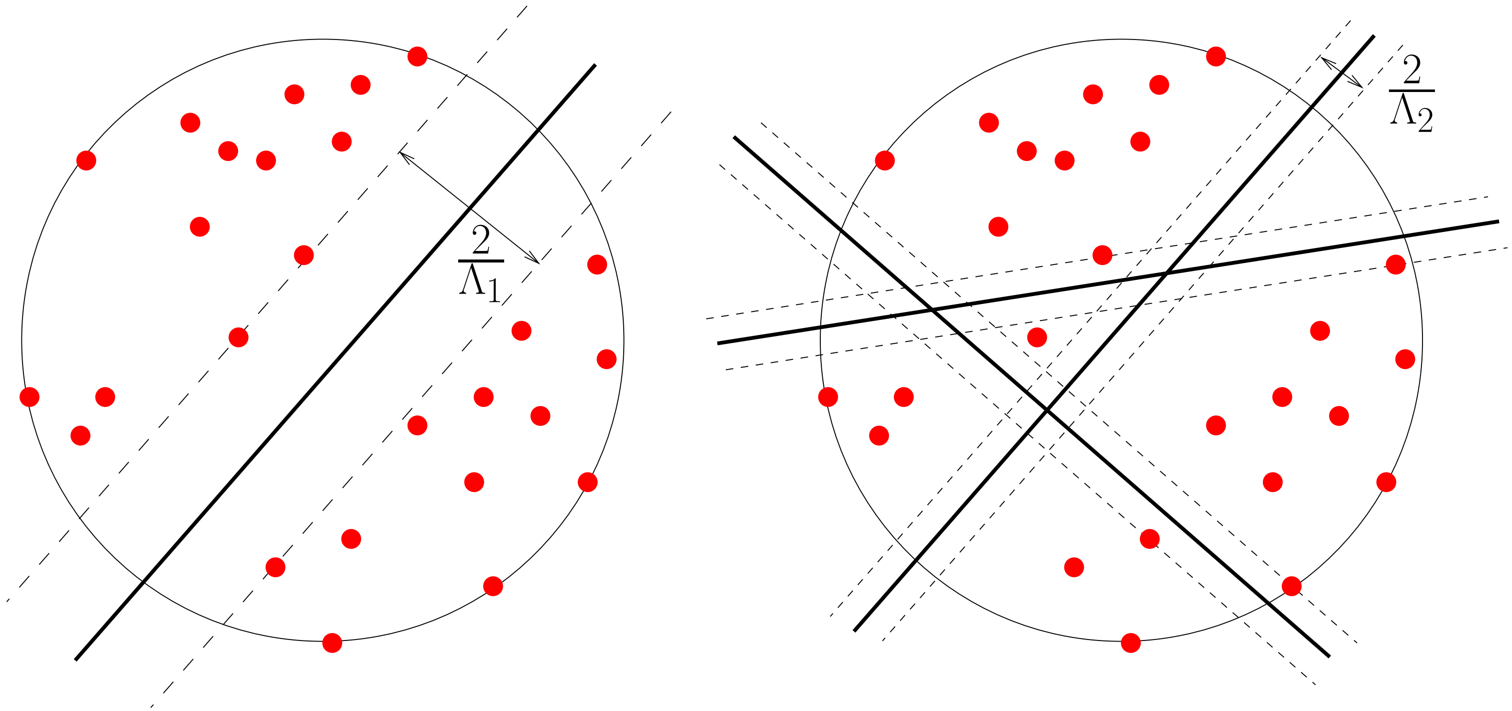
$$\min_{i=1,\ldots,r} |(\mathbf{w} \cdot \mathbf{x}_i)| = 1.$$

The set of decision functions $f_{\mathbf{w}}(\mathbf{x}) = \text{sgn}(\mathbf{x} \cdot \mathbf{w})$ defined on X^* and satisfying the constraint $\|\mathbf{w}\| \leq \Lambda$ has VC dimension satisfying

$$h \leq \min(R^2\Lambda^2 + 1, N + 1)$$

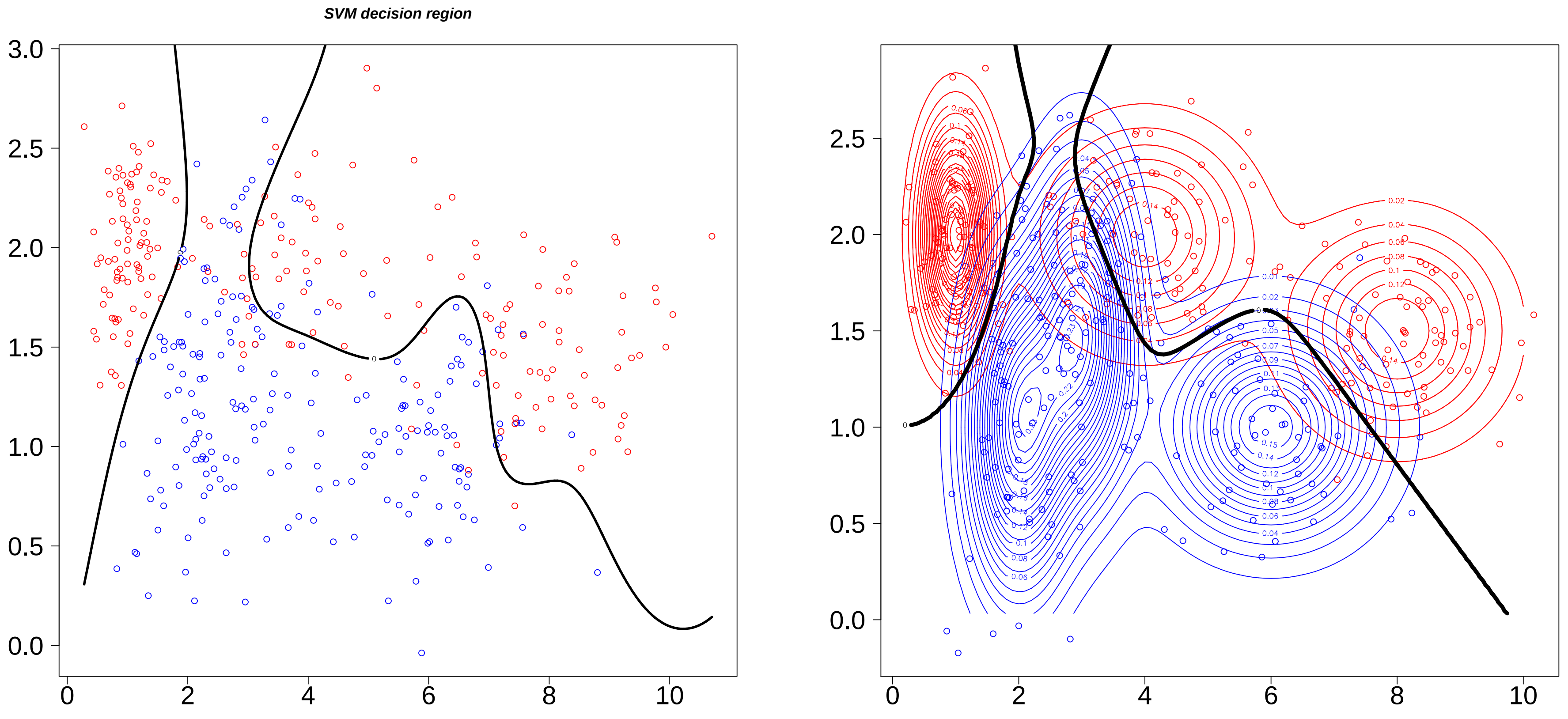
Here, R is the smallest sphere around the origin containing X^*.

Smallest Sphere and VC Dimension



Hyperplanes with a large margin ($\frac{2}{\Lambda_1}$) induce only a small number of possibilities to separate the data, i.e. the VC dimension is small (left figure). In contrast, smaller margins ($\frac{2}{\Lambda_2}$) induce more separating hyperplanes, i.e. the VC dimension is large (right figure).

SVM (RBF Kernel) and Bayes Decision



Implementation (Matrix notation)

Recall:

$$\text{maximize} \qquad W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j)$$

$$\text{subject to} \qquad \alpha_i \geq 0, \quad i = 1, \ldots, m \text{ and } \sum_{i=1}^{m} \alpha_i y_i = 0$$

This can be expressed in a matrix notation as

$$\min_{\alpha} \frac{1}{2} \alpha^T H \alpha + c^T \alpha$$

Implementation (Matrix notation) cont.

where

$$H = ZZ^T, \quad c^T = (-1, \ldots, -1)$$

with constraints

$$\alpha^T Y = 0, \alpha_i \geq 0, i = 1, \ldots, m$$

where

$$Z = \begin{bmatrix} y_1 \mathbf{x}_1 \\ y_2 \mathbf{x}_2 \\ \vdots \\ y_m \mathbf{x}_m \end{bmatrix} \quad Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}.$$

Implementation (Maple)

```
CalcAlphaVector := proc(LPM::Matrix, Kernel, C)

local H,i,j,k,rows,a,b,c,bl,bu,Aeq,
      beq,Y,QPSolution;

rows := RowDimension(LPM);
H := Matrix(1..rows,1..rows);
a := Vector(2);
b := Vector(2);
beq := Vector([0]);
Y := Vector(rows);

for k from 1 to rows do
  Y[k] := LPM[k,3];
end do;
```

Implementation (Maple) cont.

```
for i from 1 to rows do
  for j from 1 to rows do

    a[1] := LPM[i,1];
    a[2] := LPM[i,2];

    b[1] := LPM[j,1];
    b[2] := LPM[j,2];

    H[i,j] := Y[i] * Y[j] * Kernel(a,b);

  end do;
end do;
```

Implementation (Maple) cont.

```
c   := Vector(1..rows, fill = -1);
bl  := Vector(1..rows, fill = 0);
bu  := Vector(1..rows, fill = C);
Aeq := convert(Transpose(Y),Matrix);

QPSolution := QPSolve([c,H],
              [NoUserValue,NoUserValue,Aeq,beq],
              [bl,bu]);

return QPSolution[2];

end proc:
```

Using SVM in R (e1071)

Example, how to use SVM classification in R:

```
library(e1071);

# load famous Iris Fisher data set
data(iris);
attach(iris);

# number of rows
n <- nrow(iris);
# number of folds
folds <- 10
samps <- sample(rep(1:folds, length=n),
                n, replace=FALSE)
```

Using SVM in R (e1071) cont.

```
# Using the first fold:
train <- iris[samps!=1,] # fit the model
test <- iris[samps==1,] # predict

# features 1 -> 4 from training set
x_train <- train[,1:4];
# class labels
y_train <- train[,5];

# determine the hyperplane
model <- svm(x_train,y_train,type="C-classification",
             cost=10,kernel="radial",
             probability = TRUE, scale = FALSE);
```

Using SVM in R (e1071) cont.

```
# features 1 -> 4 from test set (no class labels)
x_test <- test[,1:4];

# predict class labels for x_test
pred <- predict(model,x_test,decision.values = TRUE);

# get true class labels
y_true <- test[,5];

# check accuracy:
table(pred, y_true);
# determine test error
sum(pred != y_true)/length(y_true);
# do that for all folds, i.e. train <- iris[samps!=2,]
# test <- iris[samps==2,] , ........
```

Summary (SVM)

- Optimal Separating Hyperplane
- Formulation as an Optimization Problem (Primal,Dual)
- Only Support Vectors are Relevant (Sparseness)
- Mapping in High-dimensional Spaces
- Kernel Trick
- Replacement for Neural Networks?