University of Verona

A.Y. 2021-22

Machine Learning & Artificial Intelligence

Kernel-based learning methods and Support Vector Machines

Vittorio Murino

Outline

Why SVMs?

1. Conceptually simple
2. Powerful and elegant
3. Fast

Applications

- Facial recognition
- Content–based image retrieval
- Facial expression classification
- Hand–written text interpretation
- 3D object recognition
- Texture Classification
- Text classification
- Traffic prediction
- Disease identification
- Gene sequencing
- Protein folding
- Weather forecasting
- Earthquake prediction
- Automated diagnosis

Many more…

Face recognition demo as seen in Viola, Jones, “Robust Real-time Object Detection”, IJCV 2001

Chronology

1979
Underlying theory developed, enunciation
of the Structural Risk Minimization principle
Vapnik, “Estimation of Dependences Based on Empirical Data”, Moscow, 1979.

1992
SVMs introduced in COLT-92
Boser, Guyon, Vapnik, “A training algorithm for optimal margin classifiers”, Computational Learning Theory, Pittsburgh,1992.

1995
Framework extended to non-separable problems
Cortes, Vapnik, “Support Vector Networks”, *Machine Learning,* 1995.

1999
A fast algorithm for training and testing is proposed
Platt, “Fast training of support vector machines using sequential minimal optimization”, in “Advances in Kernel Methods”, MIT Press, Cambridge, MA, 1999.

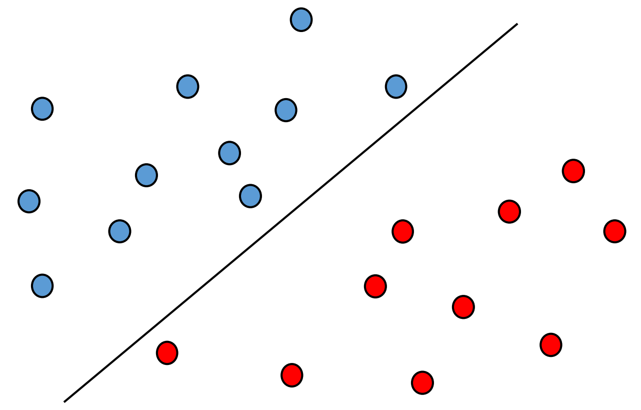
Vladimir Vapnik

Linear SVMs

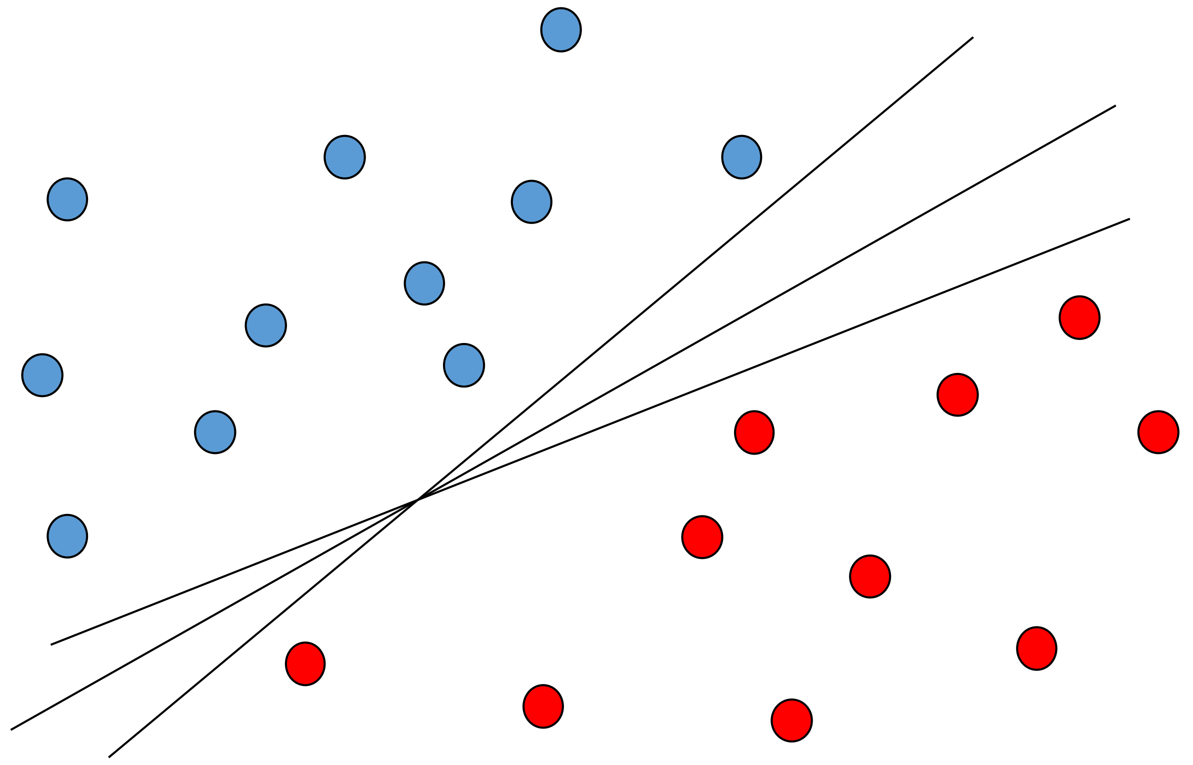
- Binary classifier
- Data is linearly separable
  - exists a hyperplane that divides the classes
  - given a set $\{\mathbf{x}_i, y_i\}$ of tuples and their labels

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad \forall i \in \{1, \ldots, m\}$$

$$\mathbf{w} \in R^n, \quad \mathbf{x}_i \in R^n, \quad b \in R, \quad y_i \in \{+1, -1\}$$

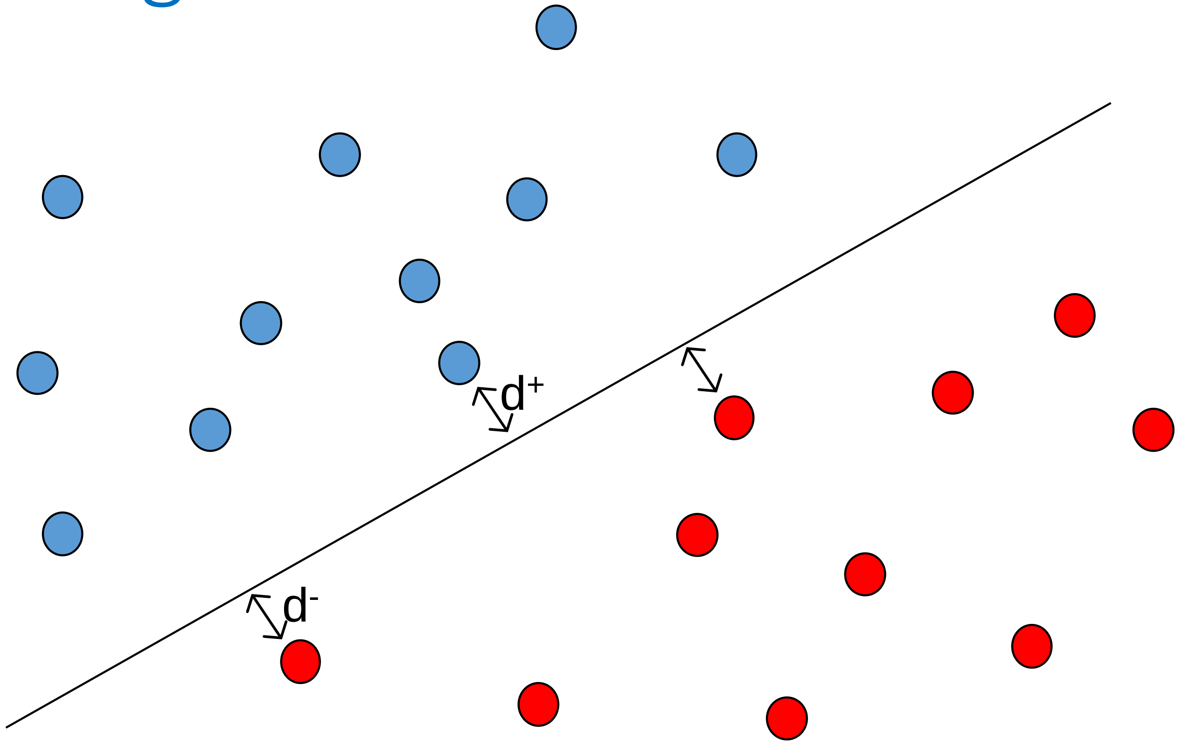


Finding the hyperplane



What is the best separating hyperplane?

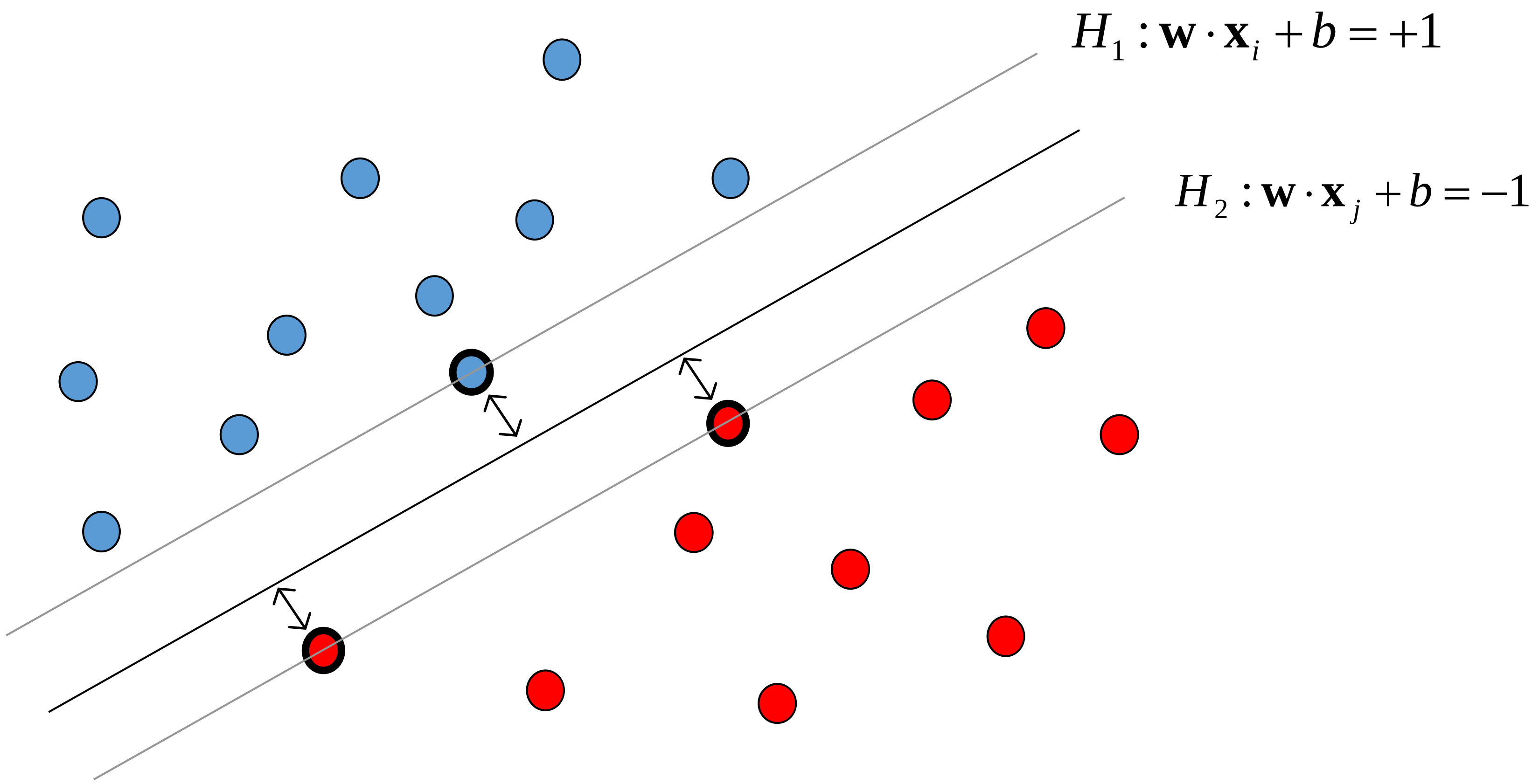
Maximizing the margin



SVMs maximize the margin *$d=d^+ + d^-$*

d^+ (*d^-*) is the minimum distance to the nearest positive (negative) sample

Support Vectors


$H_1 : \mathbf{w} \cdot \mathbf{x}_i + b = +1$
$H_2 : \mathbf{w} \cdot \mathbf{x}_j + b = -1$

The samples which lie on the lines H_1 and H_2 are called *Support Vectors*

Finding the hyperplane

If $\|\mathbf{w}\|$ is the Euclidean norm of $\mathbf{w}$, then:

$$d^+ = d^- = \frac{1}{\|\mathbf{w}\|}$$

Maximizing the margin $d = d^+ + d^- = \frac{2}{\|\mathbf{w}\|}$ is equivalent to minimize the quantity:

$$\min \frac{1}{2} \mathbf{w} \cdot \mathbf{w}$$

with $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad \forall i \in \{1, \dots, m\}$

Solving for the hyperplane

The problem outlined is an instance of constrained quadratic optimization and hence can be solved using the technique of Lagrange multipliers:

$$L_P(\mathbf{w}, b, \boldsymbol{\alpha}) = \tfrac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i \left(y_i (\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \right) \qquad \alpha_i \geq 0$$

to be minimised wrt *primal variables* $\mathbf{w}$ and b, and maximised wrt the dual variables α_i ➔ saddle point

Solving for the hyperplane

Conditions for the saddle point are:

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

which lead to: $\sum_{i=1}^{m} \alpha_i y_i = 0$ and $\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$

The solution vector has then an expansion in terms of a subset of training patterns, namely those patterns whose α_i are non-zero, which are called *Support Vectors*

Solving for the hyperplane

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

  the support vectors lie on the margin and all remaining examples of the training set are irrelevant (i.e., such contraints do not play a role in the optimization).

- Substituting the equations in the previous slide into L_P, primal variables are eliminated and the dual form of the optimization problem is found:

$$L_D(\boldsymbol{\alpha})=\sum_i \alpha_i-\tfrac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j(\mathbf{x}_i\cdot\mathbf{x}_j)$$

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

Solving for the hyperplane

- $\mathbf{w}$ is a linear combination of all training data for which $\alpha_i \neq 0$, which are the *support vectors*
- An optimal solution always exists since in a quadratic optimization problem there are no local minima (convex problem)
- Problem complexity is proportional to the number of training vectors: naive implementations are computationally expensive
- Classification of new vector $\mathbf{x}$ is obtained computing

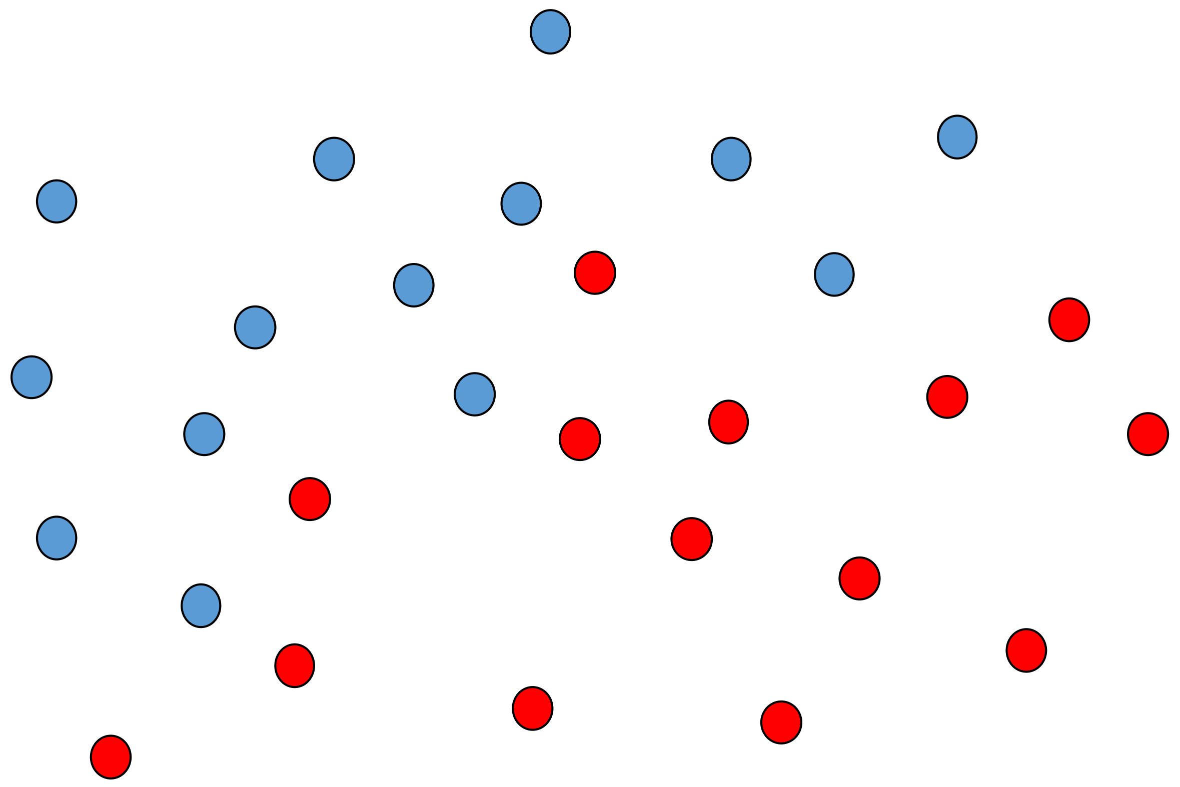
$$f(\mathbf{x}) = sign(\mathbf{w} \cdot \mathbf{x} + b)$$

where b is computed by using $\alpha_i\left(y_i\left(\mathbf{w} \cdot \mathbf{x}_i + b\right) - 1\right) = 0$

Summary: linear SVMs

- Binary classifier
- Finds the best separating hyperplane
- The hyperplane is described by a small subset of training data, called support vectors
- The procedure is fast (polynomially bounded)
- We are guaranteed to find an optimal solution
- The method is statistical, not probabilistic

Not linearly separable classes



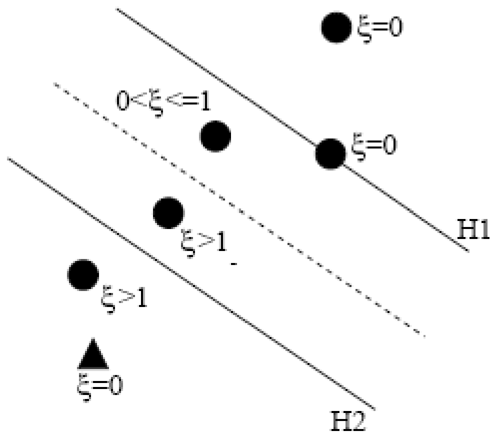
What do we do if data
are not linearly separable?

Slack variables

- Quadratic optimization does not converge for non linearly separable data
- We modify the constraints adding "slack" variables, which enable vectors to cross the margin

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

- The value of the ξ_i indicates the position of the vector with respect to the hyperplane



Optimization with slack variables

The new objective function is

$$\min \frac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\cdot\left(\sum_{1}^{m}\xi_i\right)$$

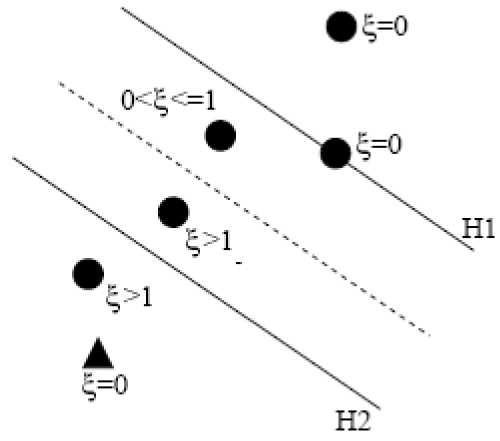
$$y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 + \xi_i \geq 0$$

- C is a user-specified parameter which represents the cost for misclassified data
- C determines the sensitivity of the classifier to errors and its generalization performance

Solution for NLS data

The constrained system can be solved in its dual form, solution is again

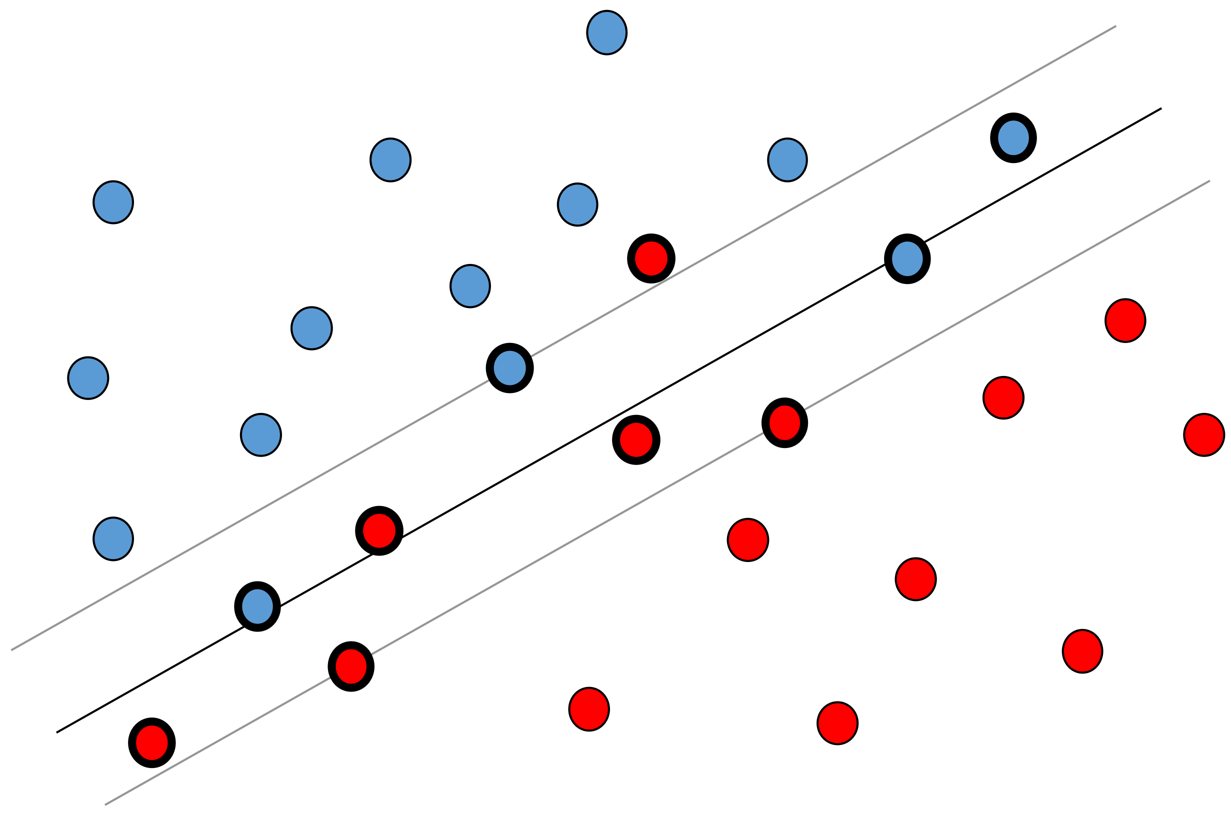
$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$$



Values of ξ can be interpreted:

$\xi_i = 0$	point is correctly classified
$0 < \xi_i < 1$	correctly classified but beyond $\mathrm{H_i}$ but on the wrong side
$\xi_i \geq 1$	incorrectly classified, error

Support vectors in the NLS case



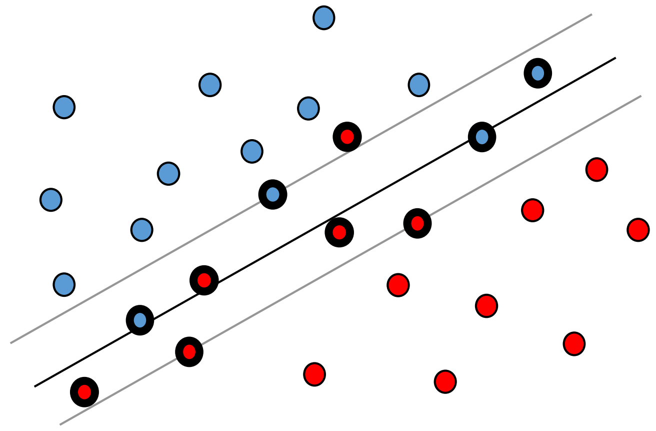
Support vectors are all points for which $\alpha \neq 0$
(misclassified data are also support vectors)

Summary: SVMs in the NLS case

- Slack variables are introduced to allow vectors to cross the margin

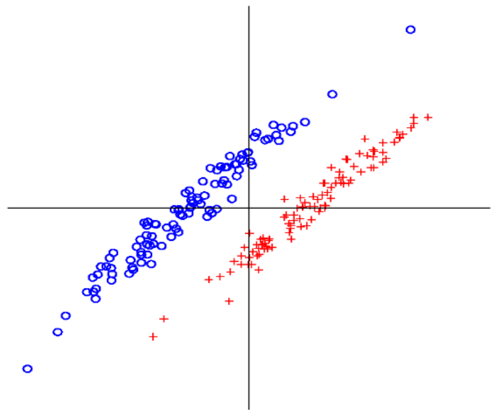
- Support vectors contain every vector which lies on or *beyond* the margin

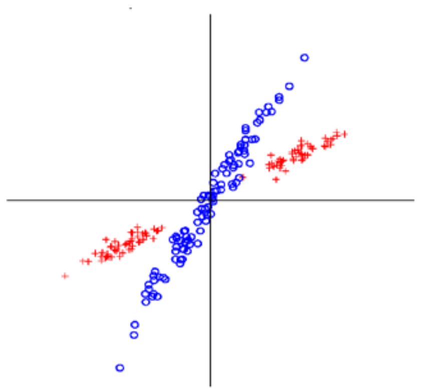
- We now need a free arbitrary parameter, the miclassification cost C



Complex examples

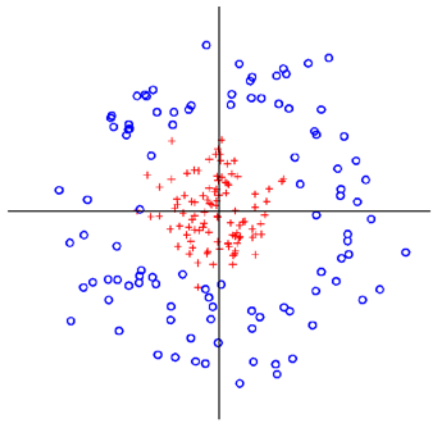
Parallel clouds

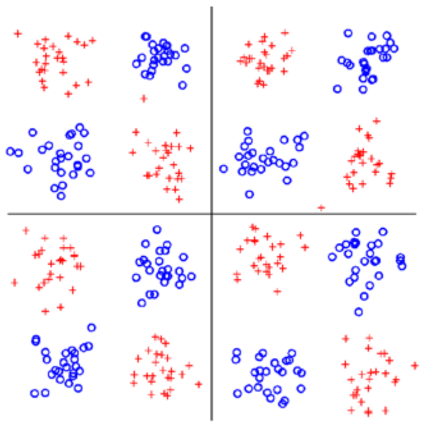




Split

Donut





Checkerboard

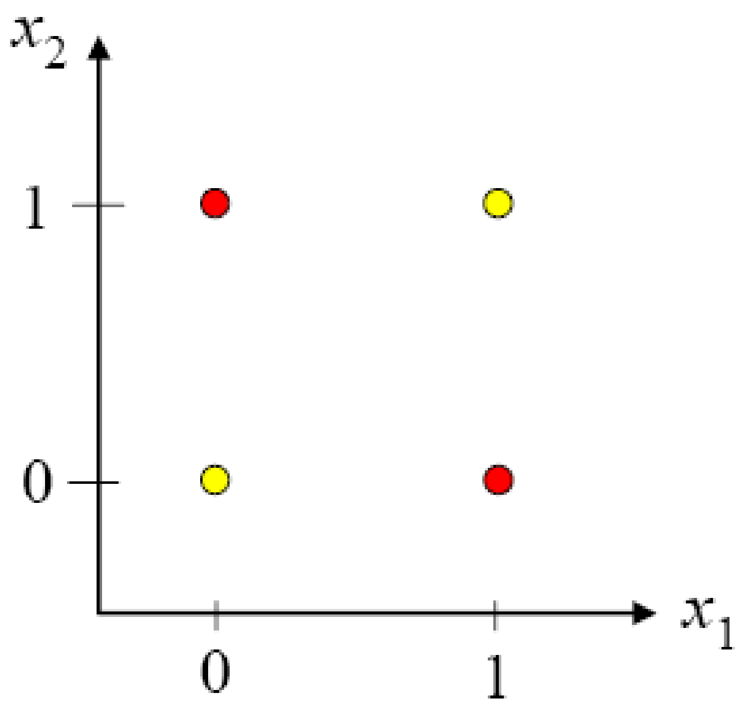
What do we do when linear hyperplanes are not sufficient?

A simpler example: XOR

Not linearly separable

(Minsky & Papert ‘69)

How can we tackle a problem like this with an SVM?



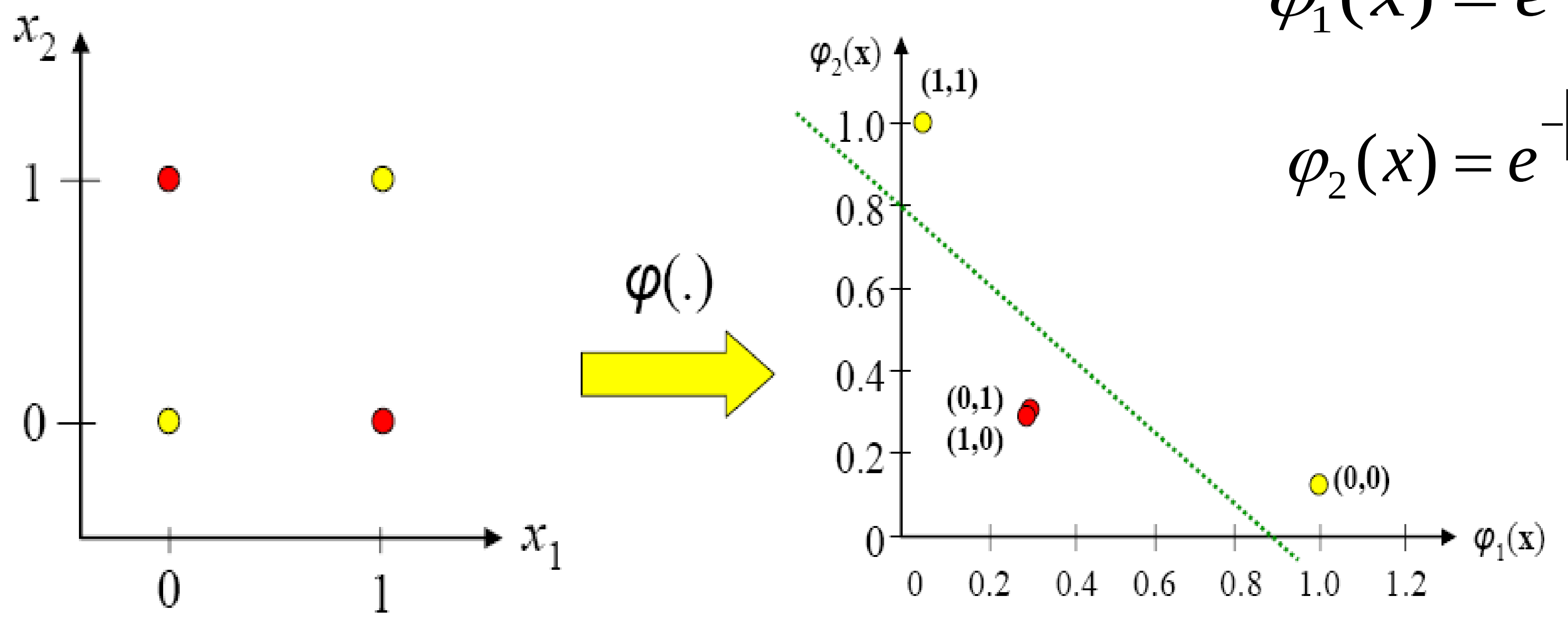
Data mapping

Could it be separable when mapped into another space?

$$x = [x_1, x_2]$$

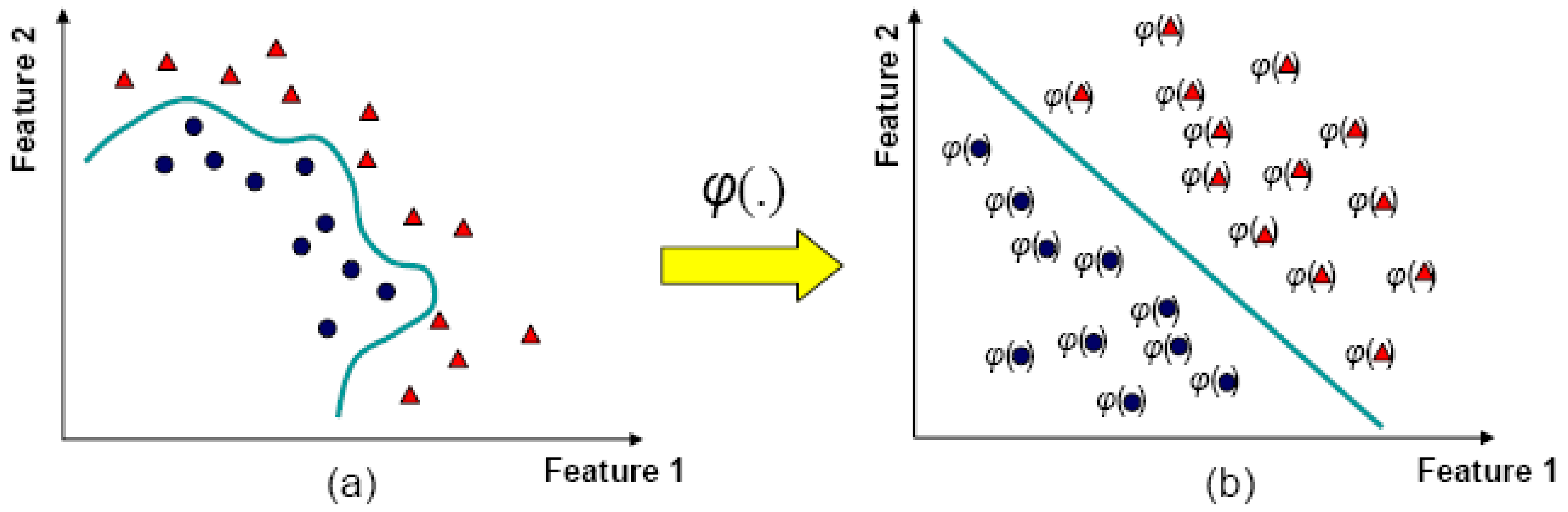
$$\varphi_1(x) = e^{-\left\| x - [0,0]^T \right\|}$$

$$\varphi_2(x) = e^{-\left\| x - [1,1]^T \right\|}$$



Data mapping: idea

A non linearly separable dataset can be mapped to another space (possibly of higher dimension), where a separating hyperplane exists



Data mapping: another example

Let's map the space $\mathrm{X} = \{x, y, z\}$ into the higher dimensional space Z:

$$\begin{array}{lll} \varphi_1(\mathrm{X}) = x & \varphi_2(\mathrm{X}) = y & \varphi_3(\mathrm{X}) = z \\ \varphi_4(\mathrm{X}) = x^2 & \varphi_5(\mathrm{X}) = y^2 & \varphi_6(\mathrm{X}) = z^2 \\ \varphi_7(\mathrm{X}) = xy & \varphi_8(\mathrm{X}) = xz & \varphi_9(\mathrm{X}) = yz \end{array}$$

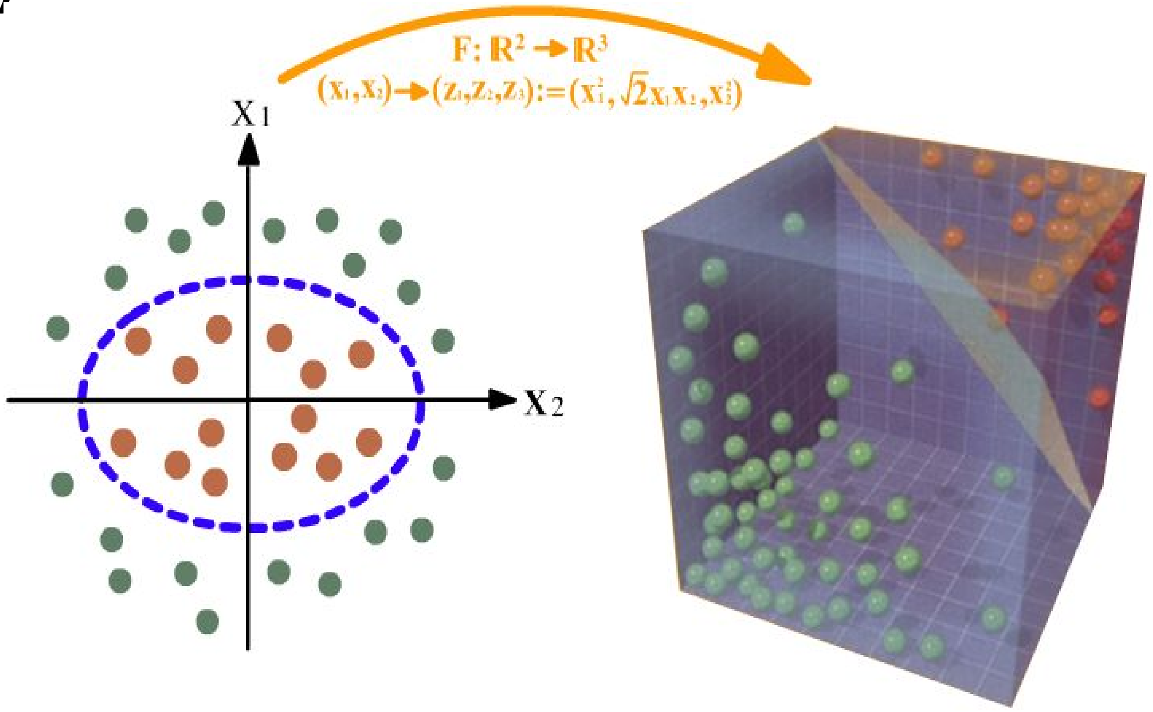
$$\mathrm{Z} = \left(\varphi_1(\mathrm{X}), \varphi_2(\mathrm{X}), \ldots, \varphi_9(\mathrm{X})\right)$$

A linear classifier in the space Z corresponds to a polynomial one in the original space

Mapping: one last example

The specified function $\mathrm{R}^2 \rightarrow \mathrm{R}^3$ maps the cartesian plane onto a 3D cone whose intersection with the x_1x_2 plane gives the elliptical boundary

$$\varphi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix}$$



Problems of the mapping idea

1. Mappings can have huge dimensionality (even infinite)

2. Mappings are in general difficult to compute

3. It is not clear how to find the proper mapping that will separate the data

Applying mapping to SVMs

- Crucial observation is that feature vectors in SVM training appear only in the form of dot products ($\mathbf{x}_i \cdot \mathbf{x}_j$)
- After applying a map φ, the training will be carried over the product $\varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$
- If we could find a function *K* defined as

$$K(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$$

we could train the classifier without even ever bothering to explicitly compute the mapping Φ.

- We call this function **Kernel** (and the technique *kernel trick*)

The kernel concept

- Using a kernel, a SVM can work in a space of huge dimensionality while using the original algorithm

- The mapping to the target space is never calculated explicitly, so it can be arbitrarily complicated

- We avoid the curse of dimensionality because the resulting classification algorithm is independent from the size of the target space

- Kernels can be seen as a problem specific module fitted in a general purpose algorithm

How to find a kernel function?

- A valid kernel should satisfy the Mercer condition
- The symmetric function k: $X^2 \rightarrow R$ defined

$$k(x, y) = \langle \varphi(x), \varphi(y) \rangle$$

is a kernel iff given a set $S = \{x_1, x_2 \ldots x_m\} \subset X$ and defined the Gram matrix G of k as

$$G = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_j) & \cdots & k(x_1, x_m) \\ \vdots & \ddots & & & \vdots \\ k(x_i, x_1) & & k(x_i, x_j) & & k(x_i, x_m) \\ \vdots & & & \ddots & \vdots \\ k(x_m, x_1) & \cdots & k(x_m, x_j) & \cdots & k(x_m, x_m) \end{bmatrix}$$

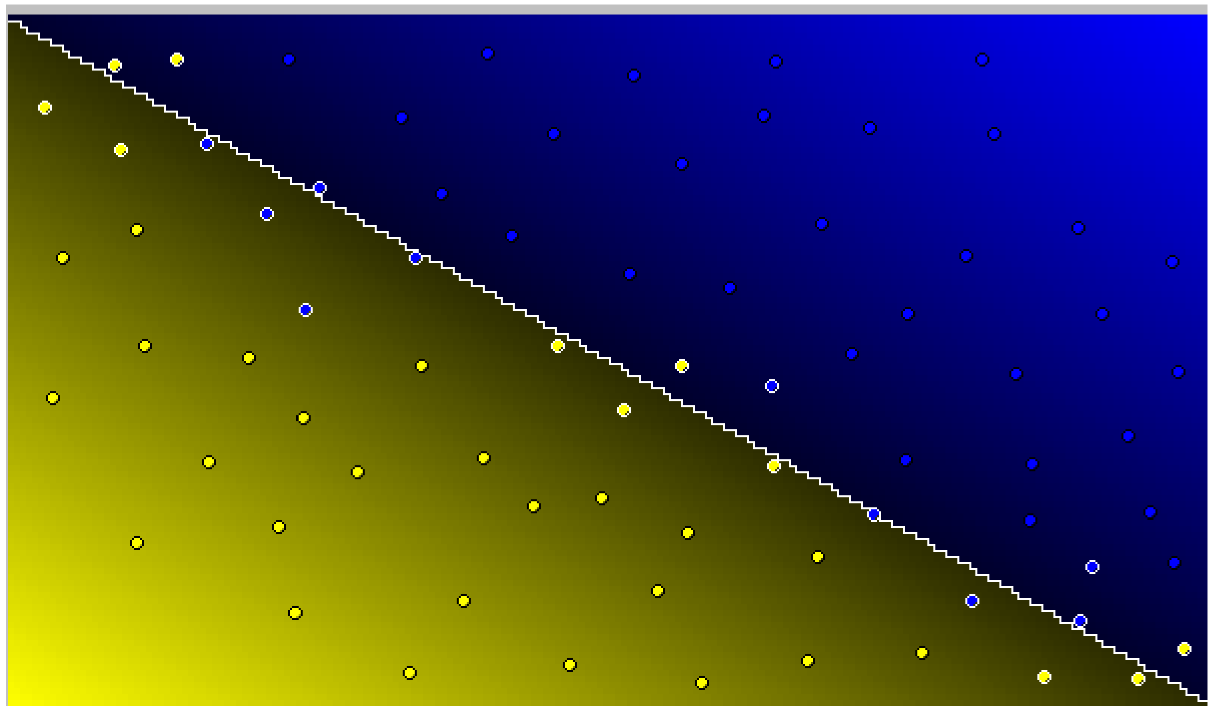
G is semidefinite positive (all eigenvalues ≥ 0)

Standard kernels

Linear	$K(x,z)=\langle x,z\rangle$
Polynomial	$K(x,z)=\left(\langle x,z\rangle+1\right)^{p}$
Radial basis functions	$K(x,z)=e^{-\frac{\lVert x-z\rVert^{2}}{2\sigma^{2}}}$
Sigmoid	$K(x,z)=\tanh(a\langle x,z\rangle+b)$

Classification examples

Number of Support Vectors: **21** (-ve: 10, +ve: 11) Total number of points: 75



Linear kernel

Polynomial, deg 1

Polynomial, deg 2

Polynomial, deg 3

Polynomial, deg 4

Polynomial deg 3, cost 100

Polynomial deg 3, cost 1000

Polynomial deg 3, cost 10000

Radial basis function, sigma 0.5

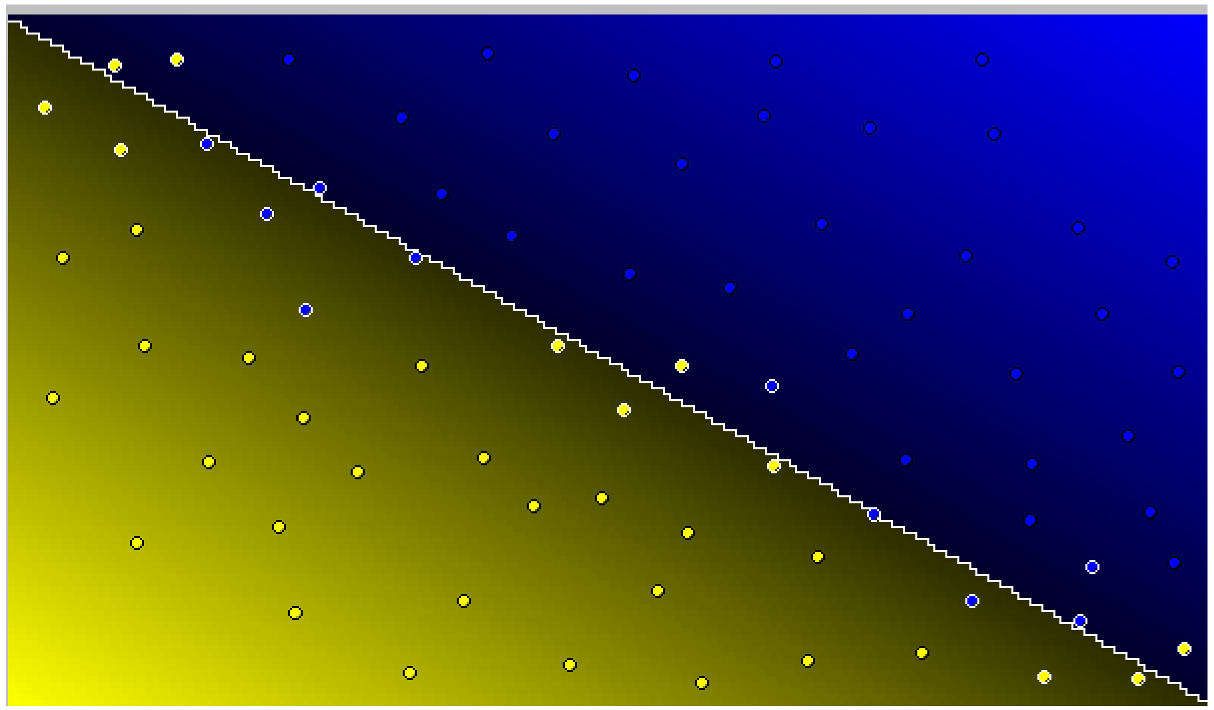
Radial basis function, sigma 1

Radial basis function, sigma 2

Radial basis function, sigma 7

Classification examples

Number of Support Vectors: **21** (-ve: 10, +ve: 11) Total number of points: 75



Linear kernel

Polynomial, deg 1

Polynomial, deg 2

Polynomial, deg 3

Polynomial, deg 4

Polynomial deg 3, cost 100

Polynomial deg 3, cost 1000

Polynomial deg 3, cost 10000

Radial basis function, sigma 0.5

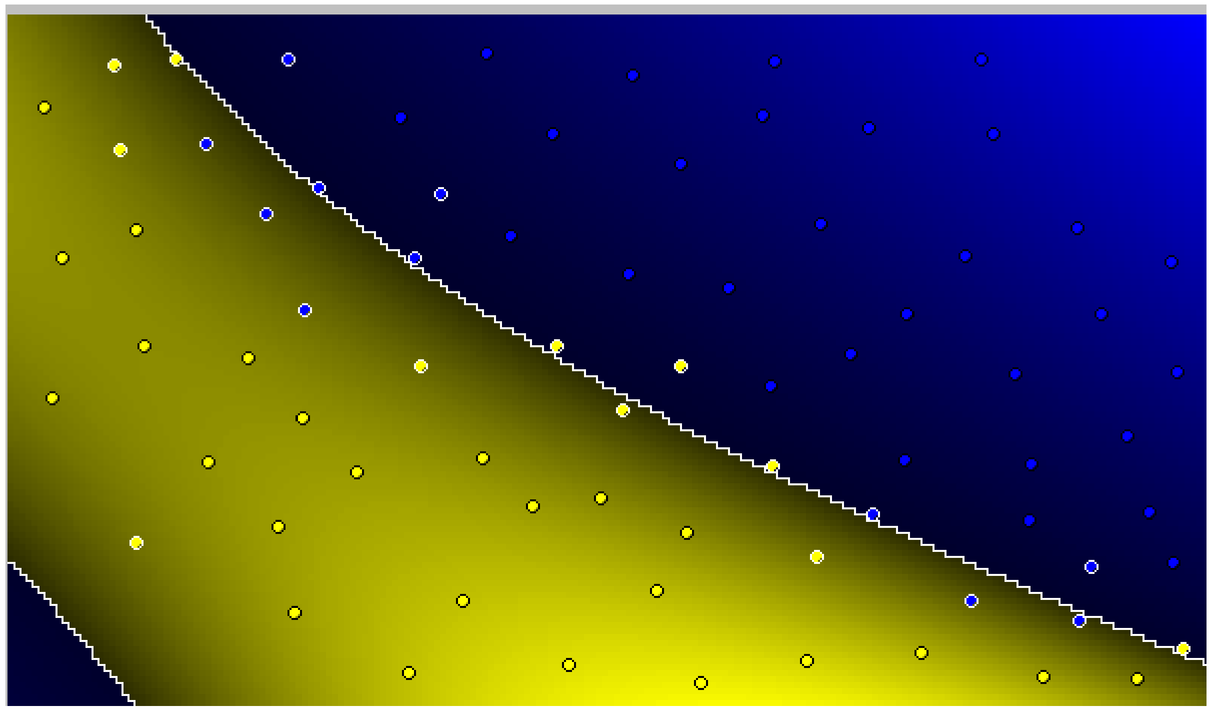
Radial basis function, sigma 1

Radial basis function, sigma 2

Radial basis function, sigma 7

Classification examples

Number of Support Vectors: **22** (-ve: 11, +ve: 11) Total number of points: 75



Linear kernel

Polynomial, deg 1

Polynomial, deg 2

Polynomial, deg 3

Polynomial, deg 4

Polynomial deg 3, cost 100

Polynomial deg 3, cost 1000

Polynomial deg 3, cost 10000

Radial basis function, sigma 0.5

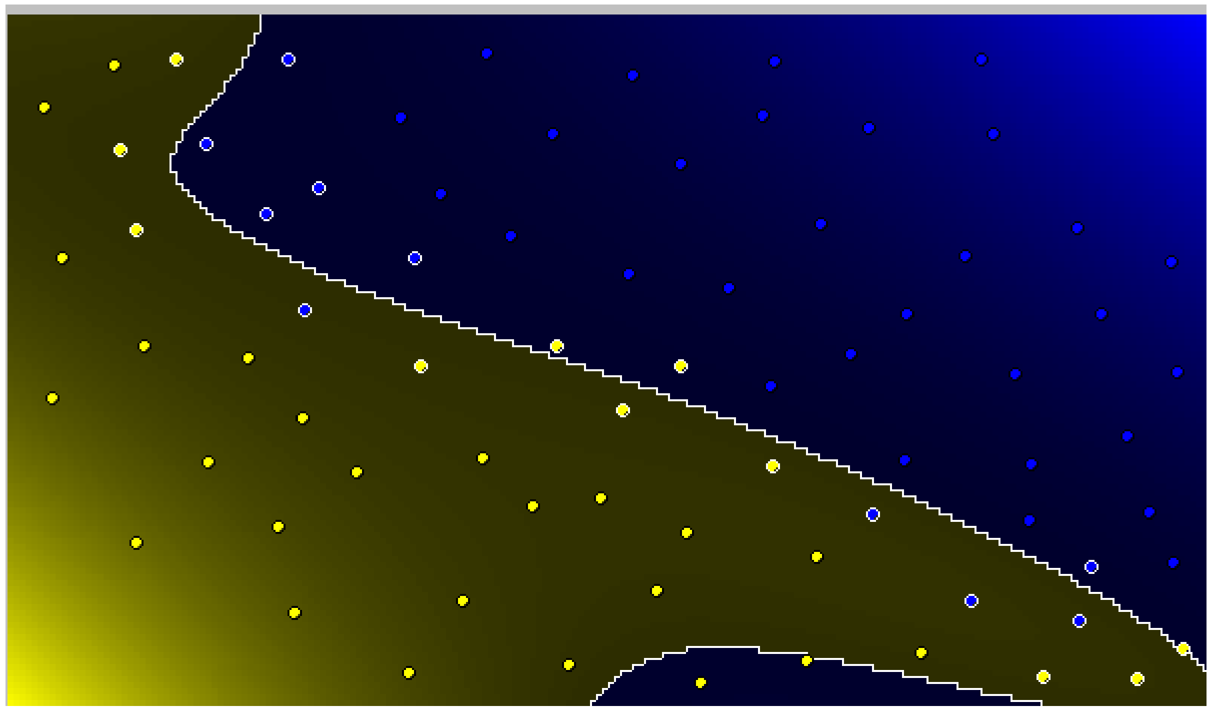
Radial basis function, sigma 1

Radial basis function, sigma 2

Radial basis function, sigma 7

Classification examples

Number of Support Vectors: **21** (-ve: 10, +ve: 11) Total number of points: 75



[Linear kernel]

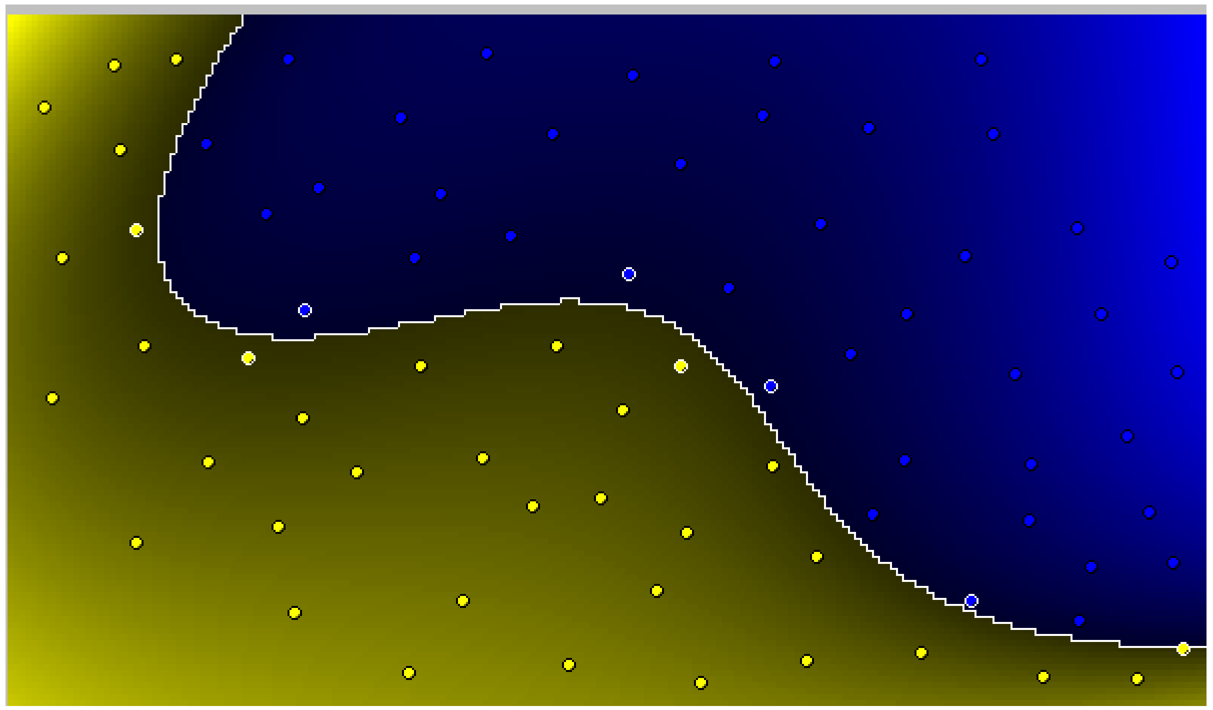
[Polynomial, deg 1]
[Polynomial, deg 2]
[Polynomial, deg 3]
[Polynomial, deg 4]

[Polynomial deg 3, cost 100]
[Polynomial deg 3, cost 1000]
[Polynomial deg 3, cost 10000]

[Radial basis function, sigma 0.5]
[Radial basis function, sigma 1]
[Radial basis function, sigma 2]
[Radial basis function, sigma 7]

Classification examples

Number of Support Vectors: 8 (-ve: 4, +ve: 4) Total number of points: 75



Linear kernel

Polynomial, deg 1

Polynomial, deg 2

Polynomial, deg 3

Polynomial, deg 4

Polynomial deg 3, cost 100

Polynomial deg 3, cost 1000

Polynomial deg 3, cost 10000

Radial basis function, sigma 0.5

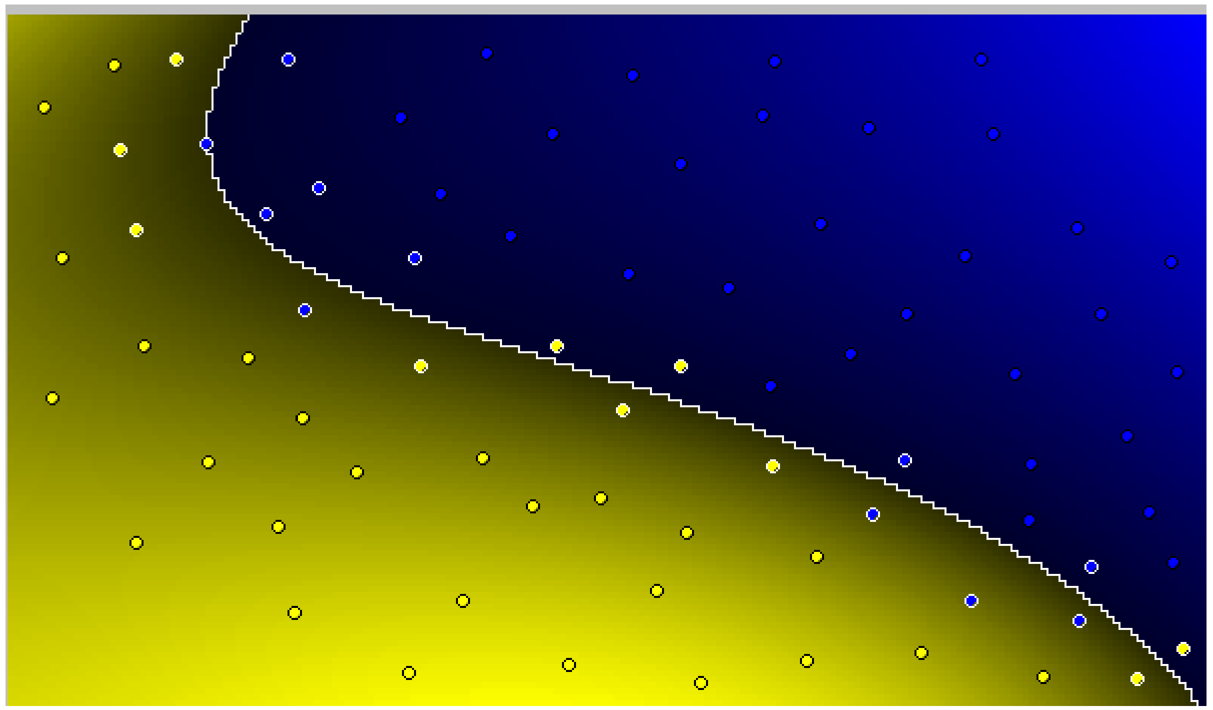
Radial basis function, sigma 1

Radial basis function, sigma 2

Radial basis function, sigma 7

Classification examples

Number of Support Vectors: **21** (-ve: 11, +ve: 10) Total number of points: 75



Linear kernel

Polynomial, deg 1

Polynomial, deg 2

Polynomial, deg 3

Polynomial, deg 4

Polynomial deg 3, cost 100

Polynomial deg 3, cost 1000

Polynomial deg 3, cost 10000

Radial basis function, sigma 0.5

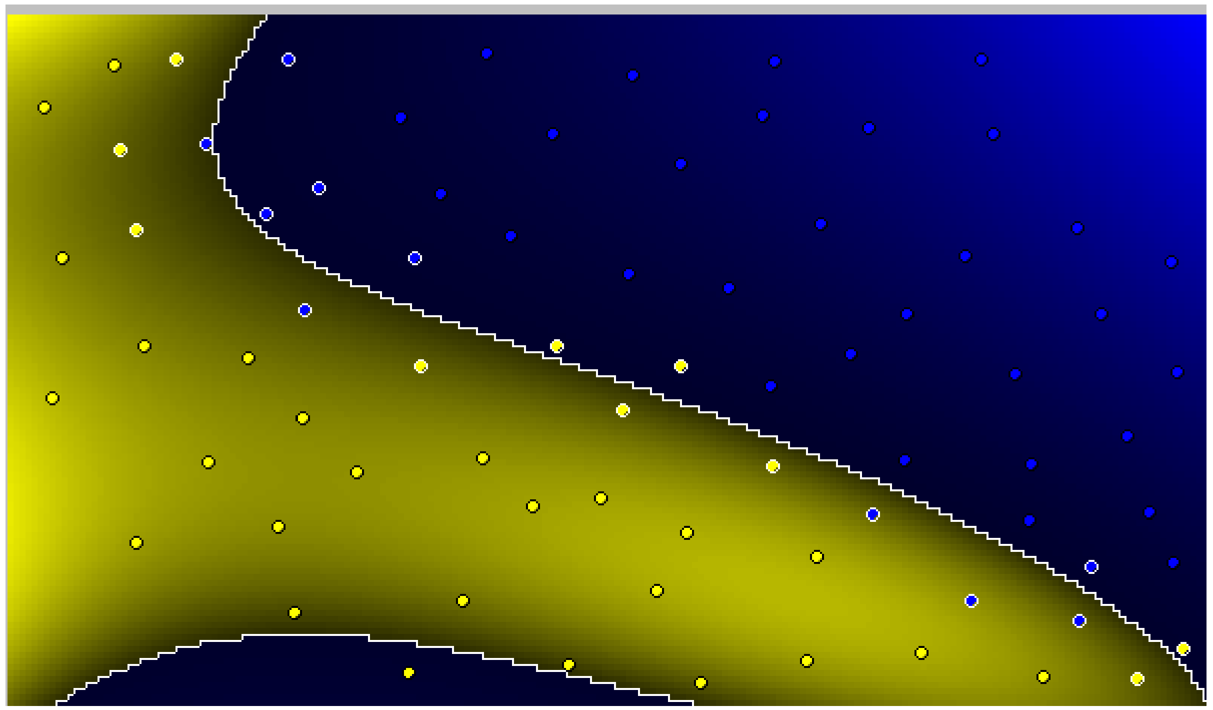
Radial basis function, sigma 1

Radial basis function, sigma 2

Radial basis function, sigma 7

Classification examples

Number of Support Vectors: **20** (-ve: 10, +ve: 10) Total number of points: 75



Linear kernel

Polynomial, deg 1

Polynomial, deg 2

Polynomial, deg 3

Polynomial, deg 4

Polynomial deg 3, cost 100

Polynomial deg 3, cost 1000

Polynomial deg 3, cost 10000

Radial basis function, sigma 0.5

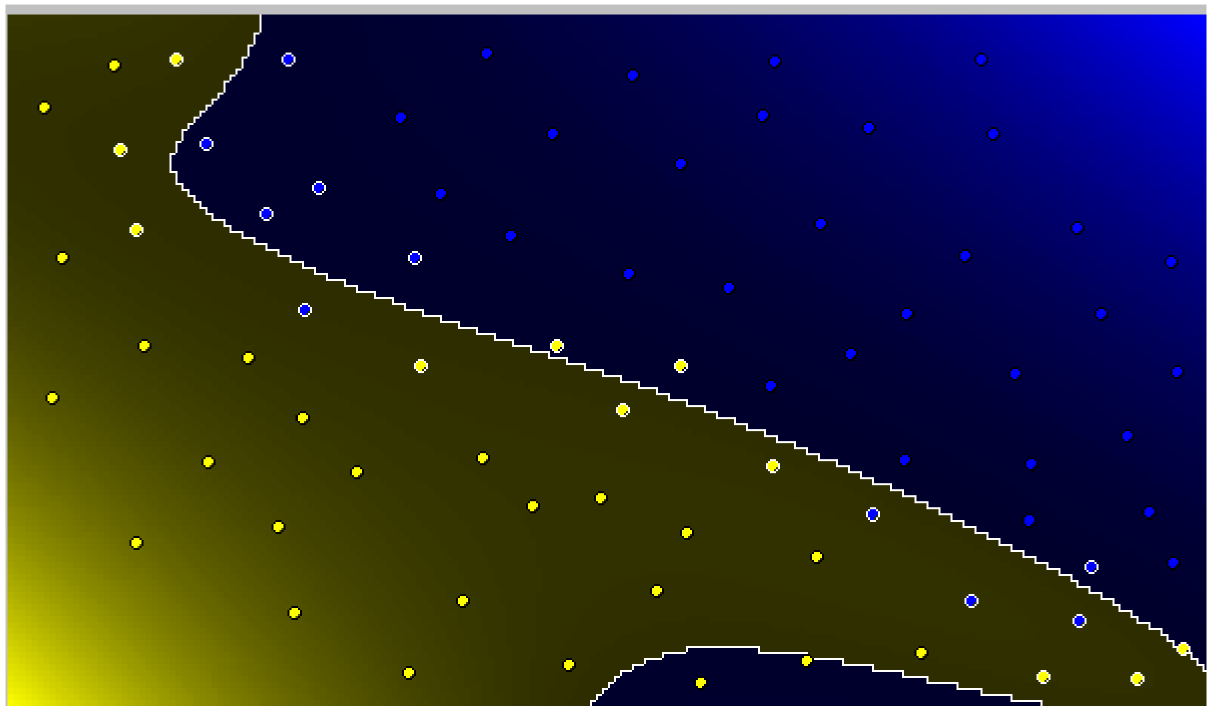
Radial basis function, sigma 1

Radial basis function, sigma 2

Radial basis function, sigma 7

Classification examples

Number of Support Vectors: **21** (-ve: 10, +ve: 11) Total number of points: 75



[Linear kernel]

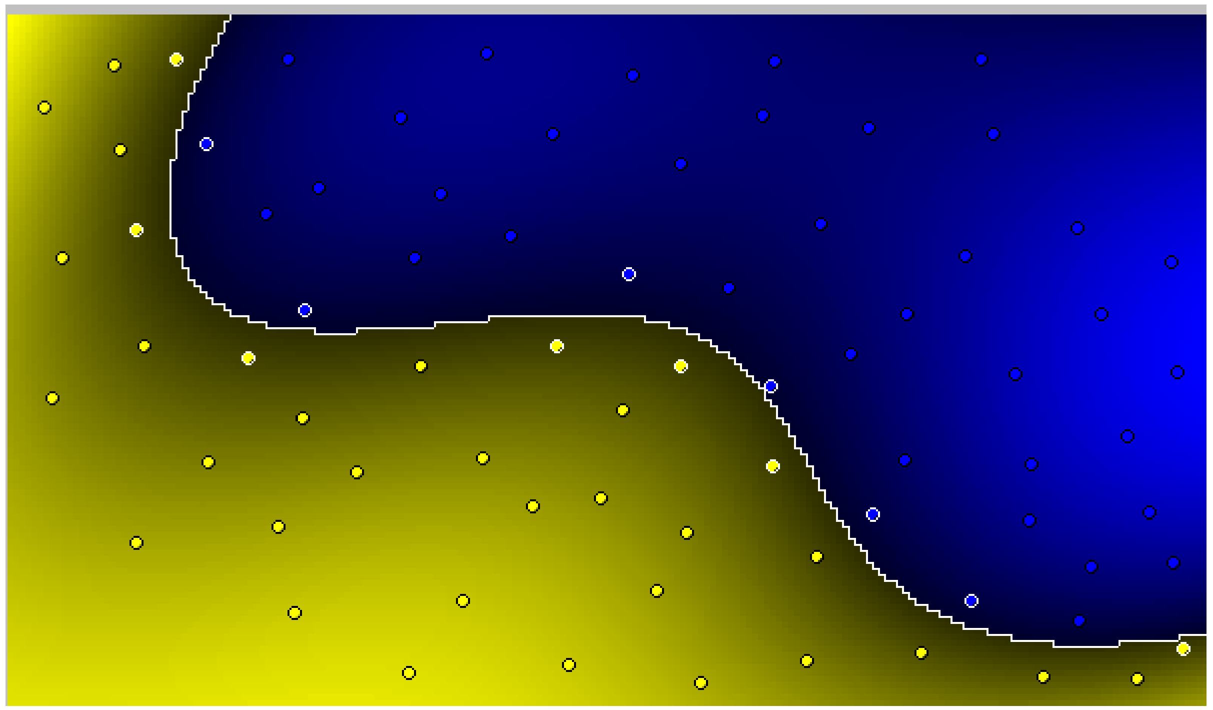
[Polynomial, deg 1]
[Polynomial, deg 2]
[Polynomial, deg 3]
[Polynomial, deg 4]

[Polynomial deg 3, cost 100]
[Polynomial deg 3, cost 1000]
[Polynomial deg 3, cost 10000]

[Radial basis function, sigma 0.5]
[Radial basis function, sigma 1]
[Radial basis function, sigma 2]
[Radial basis function, sigma 7]

Classification examples

Number of Support Vectors: **13** (-ve: 6, +ve: 7) Total number of points: 75



[Linear kernel]()

[Polynomial, deg 1]()

[Polynomial, deg 2]()

[Polynomial, deg 3]()

[Polynomial, deg 4]()

[Polynomial deg 3, cost 100]()

[Polynomial deg 3, cost 1000]()

[Polynomial deg 3, cost 10000]()

[Radial basis function, sigma 0.5]()

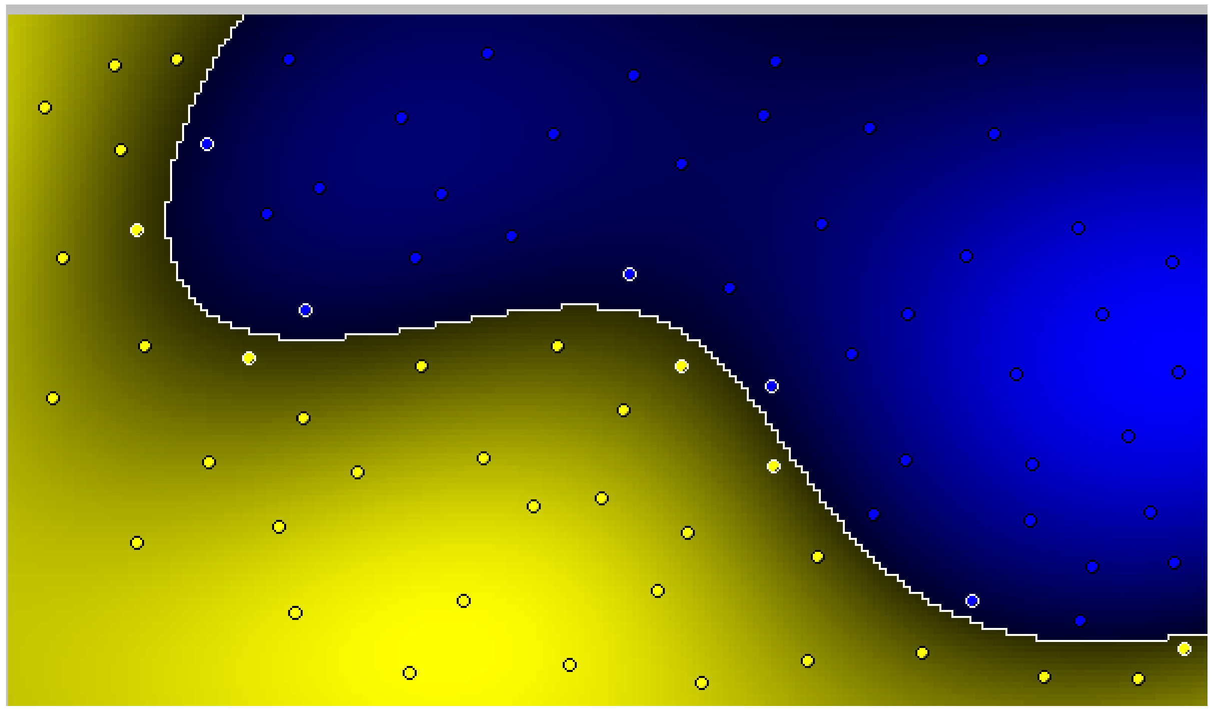
[Radial basis function, sigma 1]()

[Radial basis function, sigma 2]()

[Radial basis function, sigma 7]()

Classification examples

Number of Support Vectors: **10** (-ve: 5, +ve: 5) Total number of points: 75



Linear kernel

Polynomial, deg 1

Polynomial, deg 2

Polynomial, deg 3

Polynomial, deg 4

Polynomial deg 3, cost 100

Polynomial deg 3, cost 1000

Polynomial deg 3, cost 10000

Radial basis function, sigma 0.5

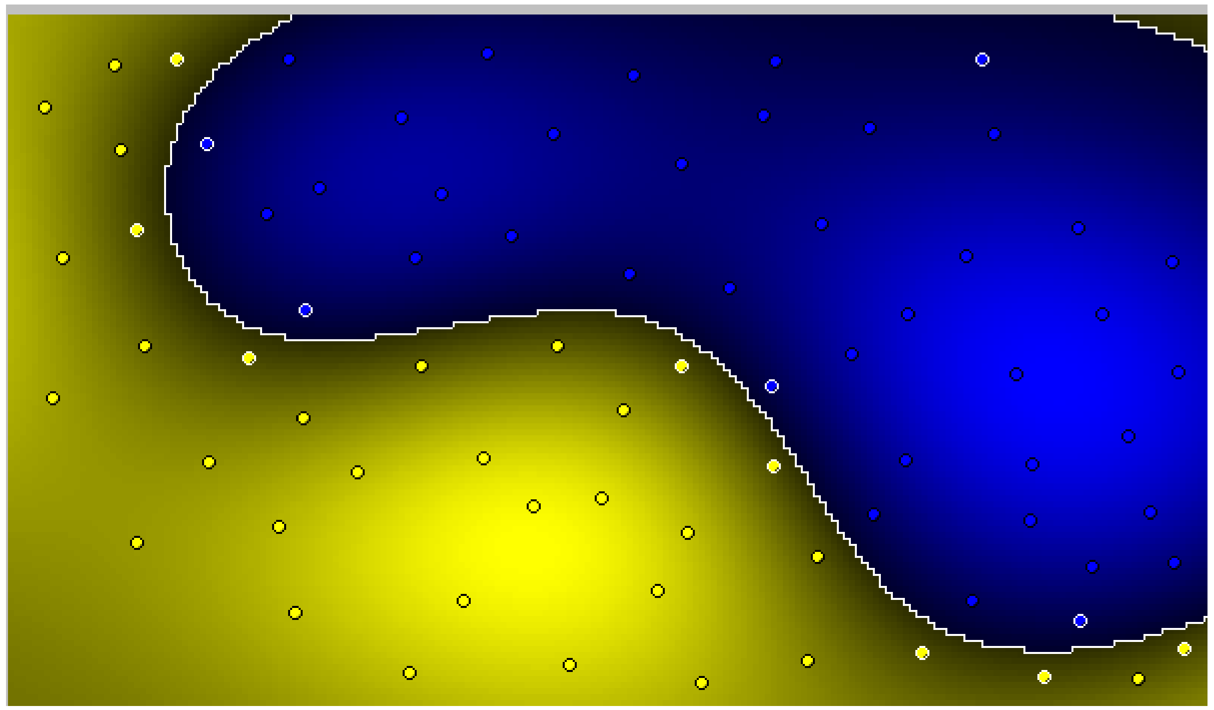
Radial basis function, sigma 1

Radial basis function, sigma 2

Radial basis function, sigma 7

Classification examples

Number of Support Vectors: **13** (-ve: 5, +ve: 8) Total number of points: 75



[Linear kernel]

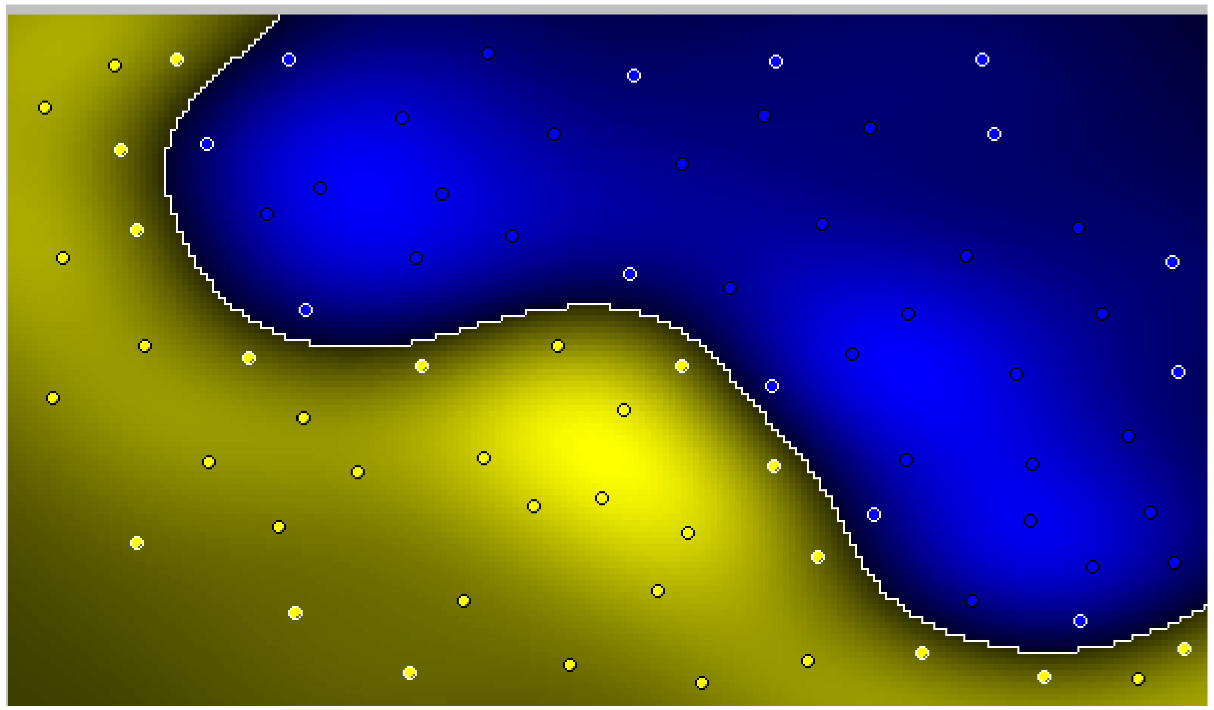
[Polynomial, deg 1]
[Polynomial, deg 2]
[Polynomial, deg 3]
[Polynomial, deg 4]

[Polynomial deg 3, cost 100]
[Polynomial deg 3, cost 1000]
[Polynomial deg 3, cost 10000]

[Radial basis function, sigma 0.5]
[Radial basis function, sigma 1]
[Radial basis function, sigma 2]
[Radial basis function, sigma 7]

Classification examples

Number of Support Vectors: **27** (-ve: 13, +ve: 14) Total number of points: 75



[Linear kernel]

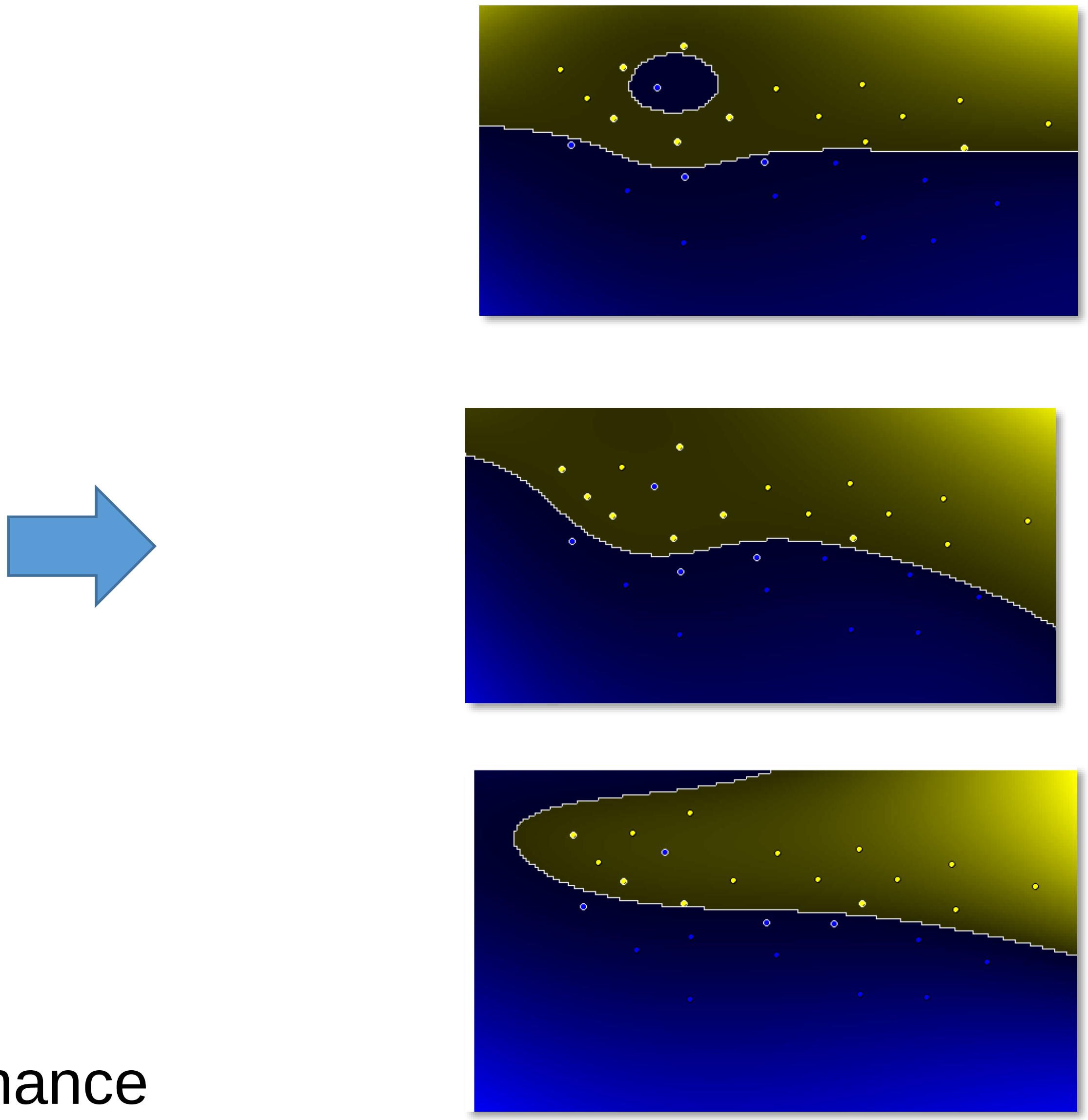
[Polynomial, deg 1]
[Polynomial, deg 2]
[Polynomial, deg 3]
[Polynomial, deg 4]

[Polynomial deg 3, cost 100]
[Polynomial deg 3, cost 1000]
[Polynomial deg 3, cost 10000]

[Radial basis function, sigma 0.5]
[Radial basis function, sigma 1]
[Radial basis function, sigma 2]
[Radial basis function, sigma 7]

Comments

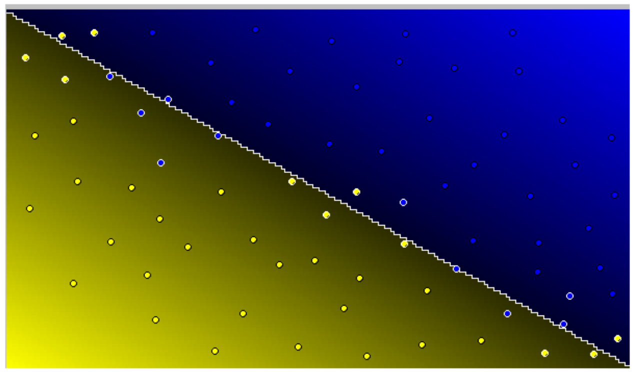
- Kernel selection and parameter tuning are critical
- Cost C has a huge impact on the generalization ability
- Lowering degree or larger sigma can avoid overfitting
- Number of support vectors is a measure of generalization performance



Kernel selection

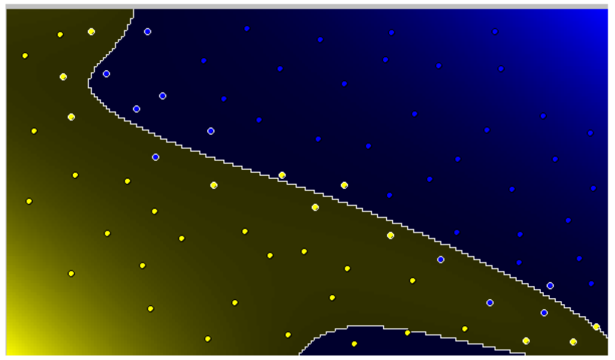
Linear kernel

- Used when the feature space is huge (for example in text classification, which uses individual word counts as features)
- Shown to be a special case of the RBF kernel
- No additional parameters



Polynomial

- Has numerical difficulties approaching 0 or infinity
- A good choice for well known and well conditioned tasks
- One additional parameter (degree p)



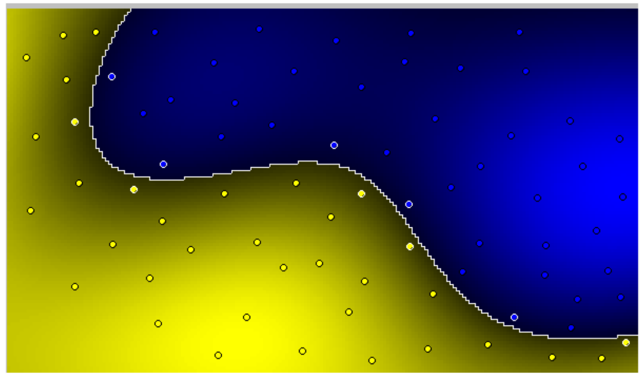
Kernel selection

Radial basis functions

- Indicated in general as the best choice in the literature
- One additional parameter (sigma σ)

Sigmoid

- Two additional parameters (a and b)
- For some values of a and b, the kernel doesn't satisfy the Mercer condition
- From neural networks
- Not recommended in the literature



Choosing the right kernel is still an art!

Cookbook approach

1. Conduct simple scaling on the data
2. Consider RBF kernel
3. Use cross-validation to find the best parameter C and σ
4. Use the best C and σ to train using the whole training set
5. Test

Cookbook problems

- Parameter search can be very time consuming

  → Solution: conduct parameter search hierarchically

- RBF kernels are sometimes subject to overfitting

  → Solution: use high degree polynomials kernels

- Parameter search must be repeated for every chosen features; there no reuse of computations

  → Solution: compare features on random subsets of the entire dataset to contain computational cost

- Search ranges for C and σ are tricky to choose

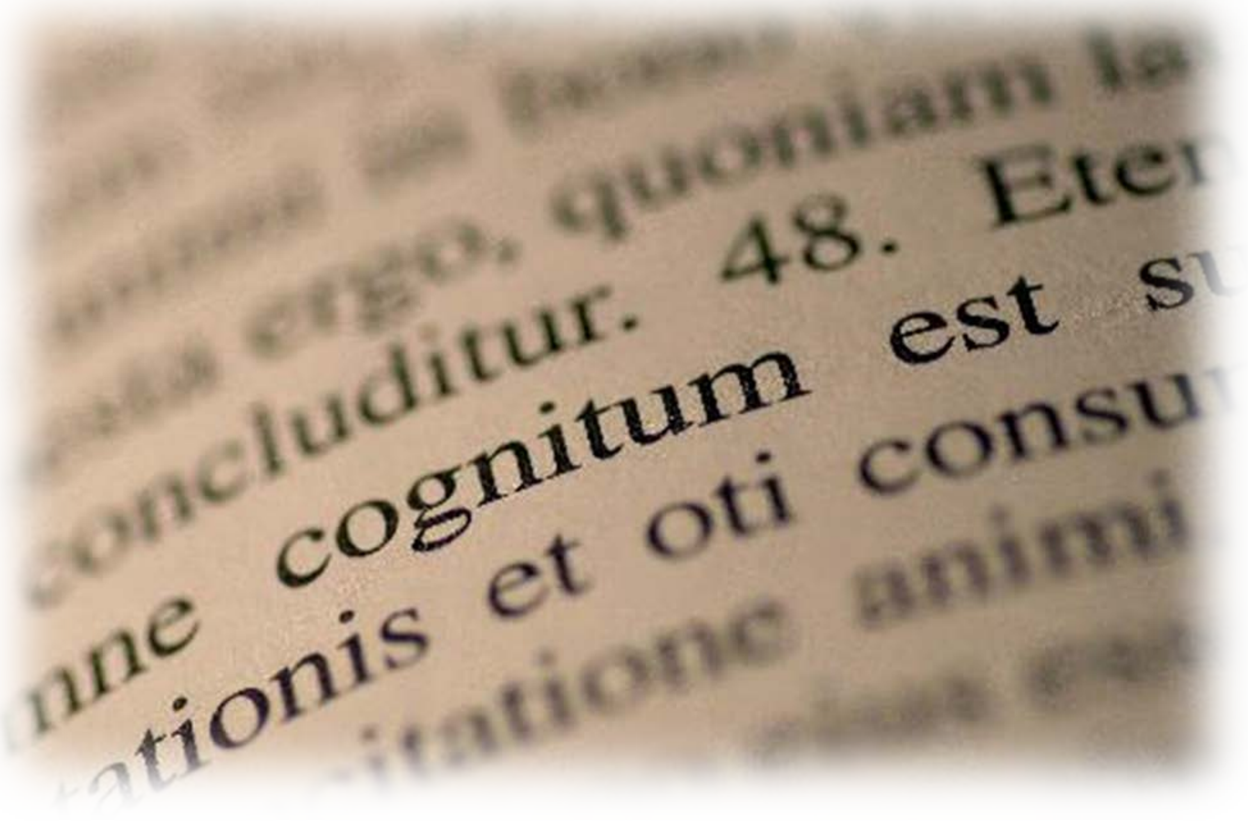
  → Solution: literature suggests using exponentially growing values like $C = 2^{[-5..15]}$ and $\sigma = 2^{[-15..5]}$

SVM vs ANN

- ANNs can suffer from multiple local minima
  SVMs have a global and unique solution
- ANN complexity is dependent on the dimensionality of the input space
  SVM complexity is independent on the dimensionality of the input space
- ANNs use empirical risk minimization
  SVMs use structural risk minimization
- Generally speaking SVMs:
  - are less prone to overfitting than ANNs
  - often (but not always) outperform (traditional) ANNs (deep ANNs are another story)
  - are faster and more predictable than ANNs
  - are less capable at finding implicit relations between input data
  - require more problem-specific knowledge

A concrete example:
TEXT CATEGORIZATION

Classify text into relevant classes

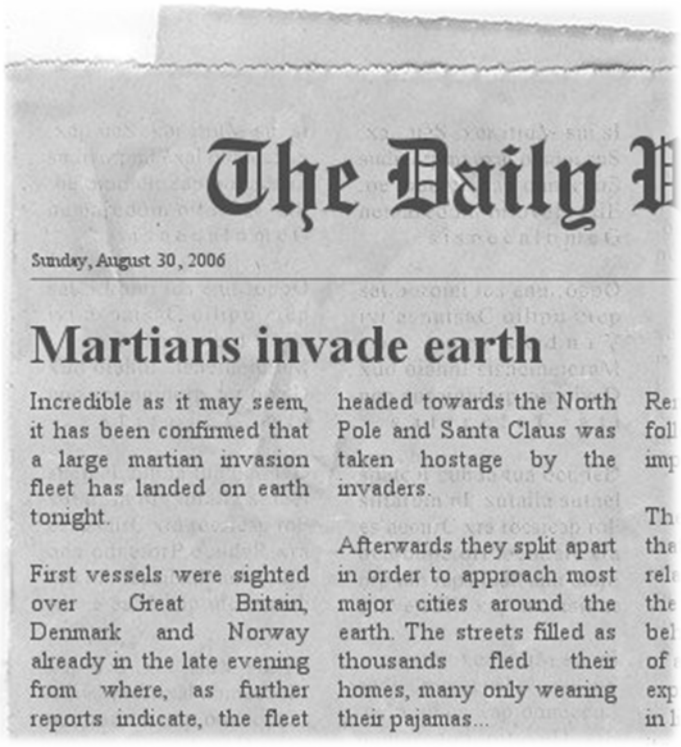


Definition of text categorization

- Given a fixed number of predefined, non mutually exclusive categories classify documents into them

- Categories work like tags: a document can be classified in none or more than one

- Supervised learning task



The Daily

Sunday, August 30, 2006

Martians invade earth

Incredible as it may seem, it has been confirmed that a large martian invasion fleet has landed on earth tonight.

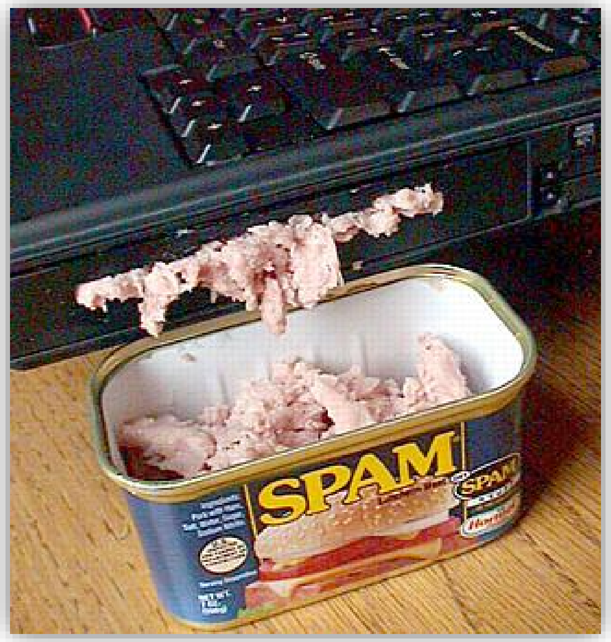
First vessels were sighted over Great Britain, Denmark and Norway already in the late evening from where, as further reports indicate, the fleet headed towards the North Pole and Santa Claus was taken hostage by the invaders.

Afterwards they split apart in order to approach most major cities around the earth. The streets filled as thousands fled their homes, many only wearing their pajamas...

martians, invasion, Santa Claus, North Pole, hoax…

Applications

- Improve web search (semantic web)
- Automatic indexing and retrieval of documents
- Author profiling and identification
- News stories classification
- Context identification
- Spam filtering



Approach

- Consider each category as indipendent
- Construct a separate classifier for each one

- Material taken from:

  Thorsten Joachims, “*Text Categorization with SVMs: Learning with Many Relevant Features*”, Proceedings of ECML-98, 1997

Text representation

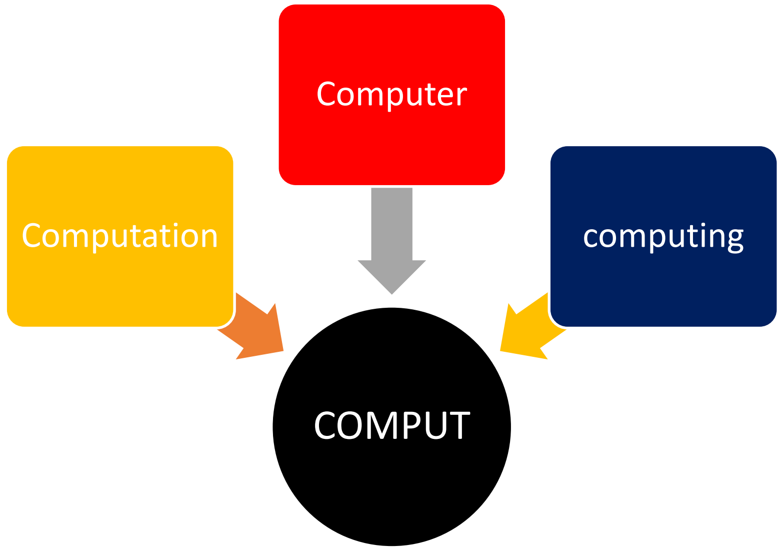
- Which features characterize a text?
    - The text itself
    - Letters/syllables/words distributions
    - Words and relative ordering
    - Word pairs
    - Punctuation
    - ???

Feature construction

- Information retrieval literature suggests:
  - words are suitable for text classification
  - relative ordering is of minor importance
  - better results are obtained extracting stems

- Word stem: obtained by removing case and flection information

- A document representation is constructed concatenating the stem counts



Feature scaling

- Individual stems are scaled on their *inverse document frequency* (IDF)

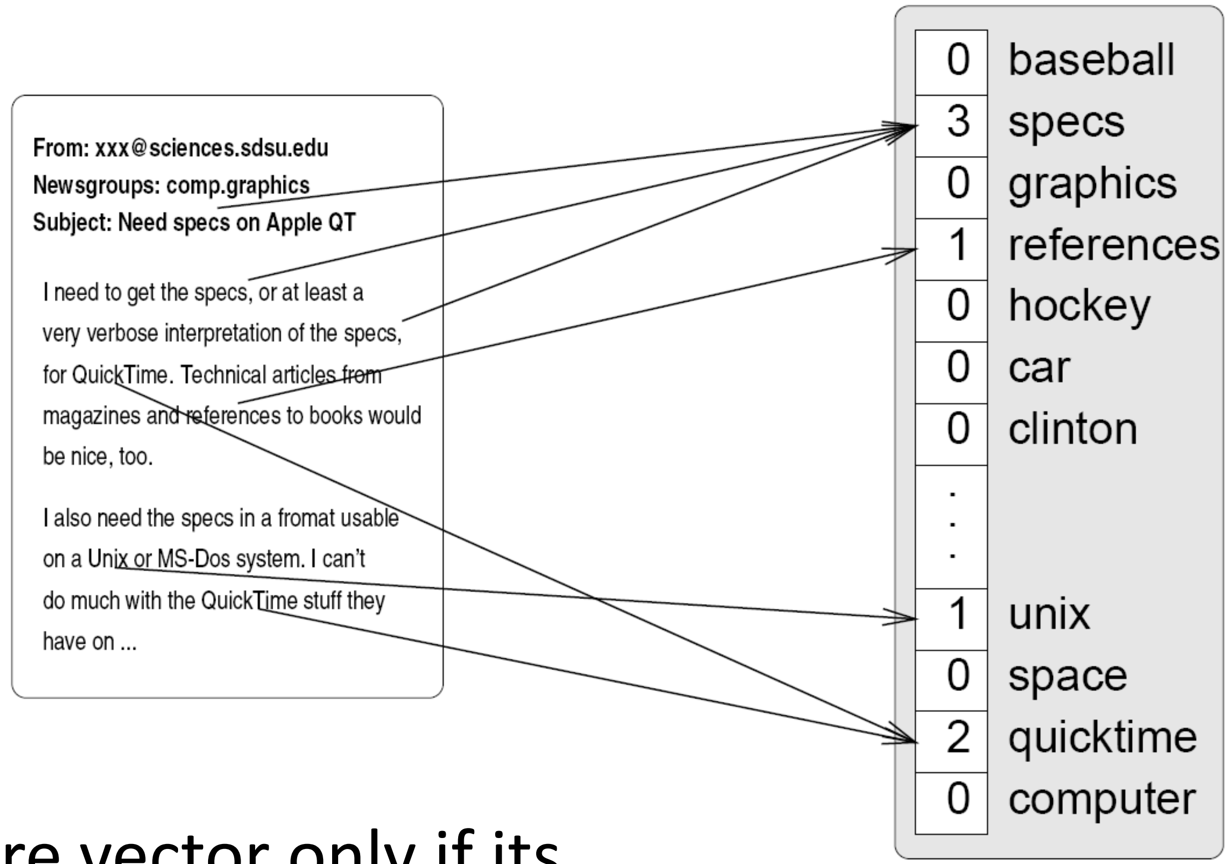
$$IDF(w_i) = \log\left(\frac{n}{DF(w_i)}\right)$$

  where *n* is the total number of documents and $DF(w_i)$ is the *document frequency*, the number of documents the word (*stem*) w_i occurs in

- Intuitively, the relevance of a word is lower if it does occur in many documents.

- To account for different document lengths, each feature vector is normalized to unit length

Model construction


From: xxx@sciences.sdsu.edu
Newsgroups: comp.graphics
Subject: Need specs on Apple QT

I need to get the specs, or at least a very verbose interpretation of the specs, for QuickTime. Technical articles from magazines and references to books would be nice, too.

I also need the specs in a fromat usable on a Unix or MS-Dos system. I can't do much with the QuickTime stuff they have on ...

0	baseball
3	specs
0	graphics
1	references
0	hockey
0	car
0	clinton
. . .	
1	unix
0	space
2	quicktime
0	computer

- A stem is added to the feature vector only if its magnitude is larger than a predefined threshold
- Common conjunctions and articles like “and”, “or”, “the” are ignored

Model considerations

- The described representation can produce feature spaces of many thousands dimensions
- Often the dimensionality of the feature space exceeds the number of available samples
- In classical cases, the problem dimensionality must be severely trimmed to keep it tractable

Datasets

- Reuters 21578
  - Compiled by David Lewis in 1987
  - Composed of manually categorized news articles
  - 9603 training documents
  - 3299 test documents
  - 90 categories
  - 9947 distinct terms after stemming and removal
- Ohsumed corpus
  - Compiled by William Hersh in 1991
  - Composed of abstracts from a on-line medical information database (MEDLINE)
  - 50216 total documents
  - 10000 used for training
  - 10000 used for testing
  - 23 categories (corresponding to different diseases)
  - 15561 distinct terms after stemming and removal

Reuters example

```
<REUTERS TOPICS="YES" LEWISSPLIT="TRAIN" CGISPLIT="TRAINING-SET" OLDID="5548" NEWID="5">
<DATE>26-FEB-1987 15:10:44.60</DATE>
<TOPICS><D>grain</D><D>wheat</D><D>corn</D><D>barley</D><D>oat</D><D>sorghum</D></TOPICS>
<PLACES><D>usa</D></PLACES>
<PEOPLE></PEOPLE>
<ORGS></ORGS>
<EXCHANGES></EXCHANGES>
<COMPANIES></COMPANIES>
<UNKNOWN>reuteu f BC-average-prices   02-26 0095</UNKNOWN>
<TEXT>
<TITLE>NATIONAL AVERAGE PRICES FOR FARMER-OWNED RESERVE</TITLE>
<DATELINE>    WASHINGTON, Feb 26 - </DATELINE>
<BODY>
        The U.S. Agriculture Departmentreported the farmer-owned reserve
        national five-day average price through February 25 as follows
        (Dlrs/Bu-Sorghum Cwt) - Natl   Loan Release   Call Avge   Rate-X  Level   Price  Price
        Wheat   2.55   2.40      IV     4.65     --     V     4.65     --        VI     4.45     --
        Corn      1.35   1.92      IV     3.15   3.15  V     3.25     --         X         –
        1986 Rates.         Natl   Loan        Release   Call        Avge   Rate-X  Level   Price  Price
        Oats     1.24   0.99       V    1.65    --
        Barley   n.a.   1.56      IV    2.55   2.55                        V    2.65    --
         Sorghum  2.34   3.25-Y    IV    5.36   5.36                        V    5.54    --
        Reserves I, II and III have matured. Level IV reflectsgrain entered after Oct 6, 1981
        for feedgrain and after July23, 1981 for wheat. Level V wheat/barley after 5/14/82,
        corn/sorghum after 7/1/82. Level VI covers wheat entered afterJanuary 19, 1984.
        X-1986 rates. Y-dlrs per CWT (100 lbs).n.a.-not available. Reuter
</BODY>
</TEXT>
</REUTERS>
```

tags

text

Training & Test

- Chosen performance measure is *Precision/Recall-Breakeven*: point at which precision and recall are equal

$$P = \frac{|D_{rel} \cap D_{ret}|}{|D_{ret}|}$$

$$R = \frac{|D_{rel} \cap D_{ret}|}{|D_{rel}|}$$

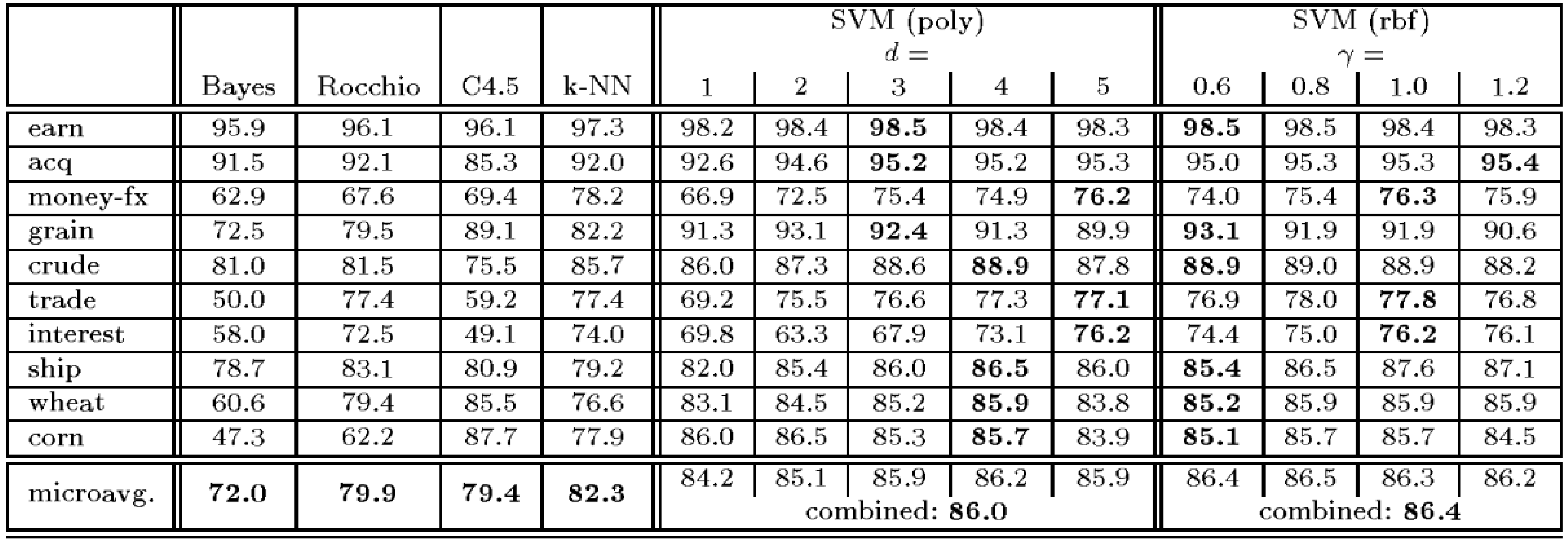
where:
$\mathrm{D}_{\mathrm{rel}}$: set of relevant documents
$\mathrm{D}_{\mathrm{ret}}$: set of retrieved documents

- In binary classification, precision P and recall R can be interpreted as probabilities
- There is a trade-off between high-precision and high-recall rates
- Training with the selected training sets on several polynomial and RBF support vector machines

Testing

- Testing with the already selected test set
- Also, comparison with other 4 methods
  - Naive Bayes classifier
  - Relevance feedback algorithm
  - K-nearest neighbours
  - Decision trees (C4.5)

Results on Reuters dataset



	Bayes	Rocchio	C4.5	k-NN	SVM (poly) $d =$					SVM (rbf) $\gamma =$			
					1	2	3	4	5	0.6	0.8	1.0	1.2
earn	95.9	96.1	96.1	97.3	98.2	98.4	**98.5**	98.4	98.3	**98.5**	98.5	98.4	98.3
acq	91.5	92.1	85.3	92.0	92.6	94.6	**95.2**	95.2	95.3	95.0	95.3	95.3	**95.4**
money-fx	62.9	67.6	69.4	78.2	66.9	72.5	75.4	74.9	**76.2**	74.0	75.4	**76.3**	75.9
grain	72.5	79.5	89.1	82.2	91.3	93.1	**92.4**	91.3	89.9	**93.1**	91.9	91.9	90.6
crude	81.0	81.5	75.5	85.7	86.0	87.3	88.6	**88.9**	87.8	**88.9**	89.0	88.9	88.2
trade	50.0	77.4	59.2	77.4	69.2	75.5	76.6	77.3	**77.1**	76.9	78.0	**77.8**	76.8
interest	58.0	72.5	49.1	74.0	69.8	63.3	67.9	73.1	**76.2**	74.4	75.0	**76.2**	76.1
ship	78.7	83.1	80.9	79.2	82.0	85.4	86.0	**86.5**	86.0	**85.4**	86.5	87.6	87.1
wheat	60.6	79.4	85.5	76.6	83.1	84.5	85.2	**85.9**	83.8	**85.2**	85.9	85.9	85.9
corn	47.3	62.2	87.7	77.9	86.0	86.5	85.3	**85.7**	83.9	**85.1**	85.7	85.7	84.5
microavg.	**72.0**	**79.9**	**79.4**	**82.3**	84.2	85.1	85.9	86.2	85.9	86.4	86.5	86.3	86.2
					combined: **86.0**					combined: **86.4**			

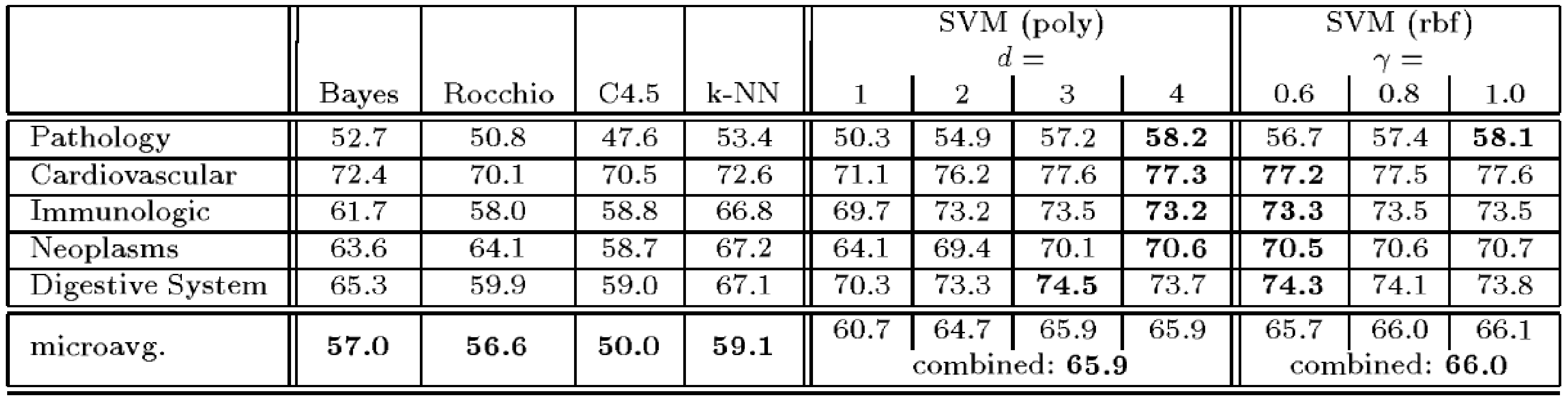
classes

competing methods

Polynomial SVM

RBF SVM

Results on Ohsumed dataset



					SVM (poly) $d =$				SVM (rbf) $\gamma =$		
	Bayes	Rocchio	C4.5	k-NN	1	2	3	4	0.6	0.8	1.0
Pathology	52.7	50.8	47.6	53.4	50.3	54.9	57.2	**58.2**	56.7	57.4	**58.1**
Cardiovascular	72.4	70.1	70.5	72.6	71.1	76.2	77.6	**77.3**	**77.2**	77.5	77.6
Immunologic	61.7	58.0	58.8	66.8	69.7	73.2	73.5	**73.2**	**73.3**	73.5	73.5
Neoplasms	63.6	64.1	58.7	67.2	64.1	69.4	70.1	**70.6**	**70.5**	70.6	70.7
Digestive System	65.3	59.9	59.0	67.1	70.3	73.3	**74.5**	73.7	**74.3**	74.1	73.8
microavg.	**57.0**	**56.6**	**50.0**	**59.1**	60.7	64.7	65.9	65.9	65.7	66.0	66.1
					combined: **65.9**				combined: **66.0**		

classes

competing methods

Polynomial SVM

RBF SVM

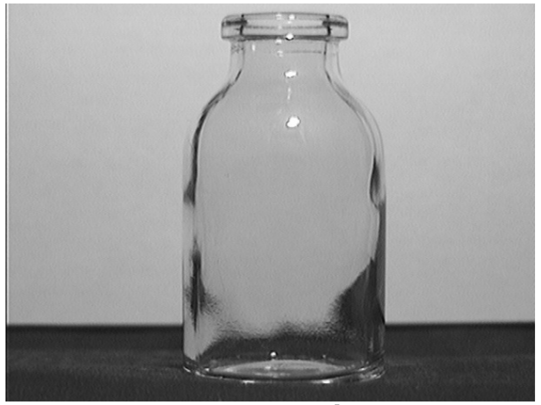
Comments

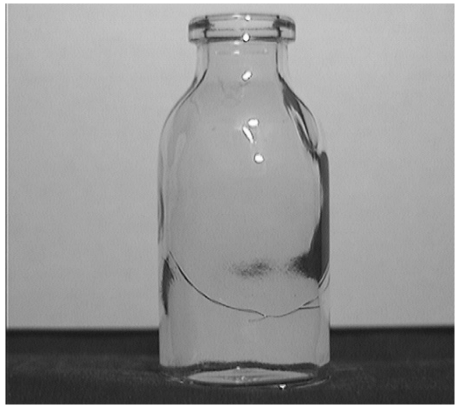
- SVMs perform very well in text categorization
- They show better results than competing methods
- Are less prone to overfitting, even in the presence of a overwhelming number of features
- 10 years ago, they were the best solution for text classification

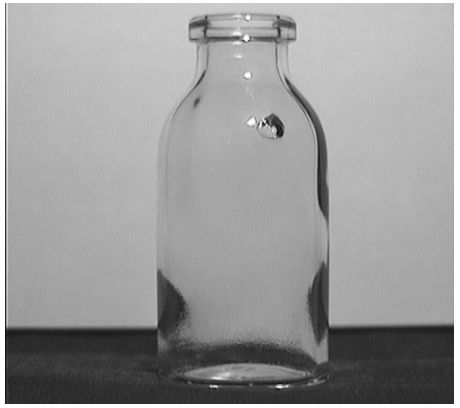
Another concrete example:
Vial Defects Classification

Identify mint bottles from

those damaged or defected


good


crack


bubble

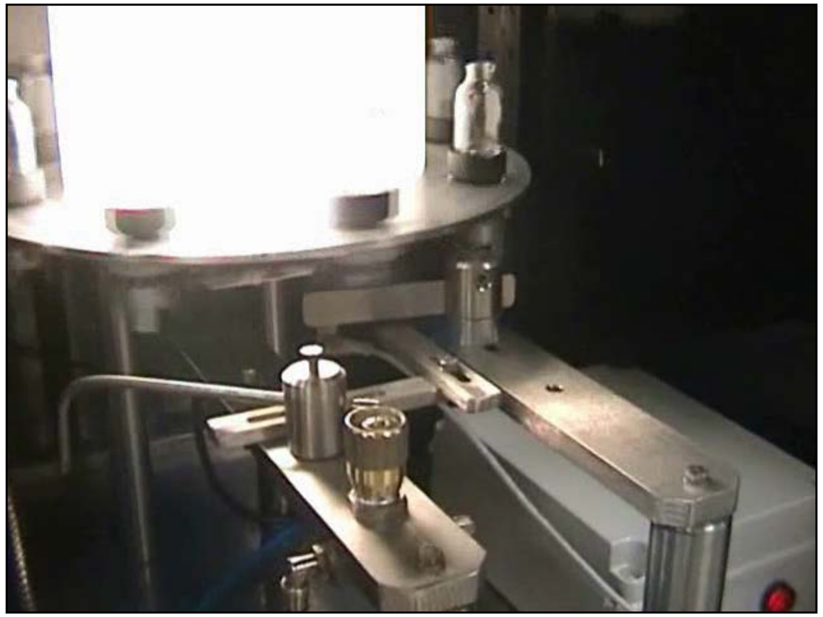
Challenges

- Molded glass do not consistently deflect light
- Various types of defects must be considered
- Defect can be subtle
- Sterilized environment imposes additional constraints on sensors and illumination placement (vials are for antibiotics)

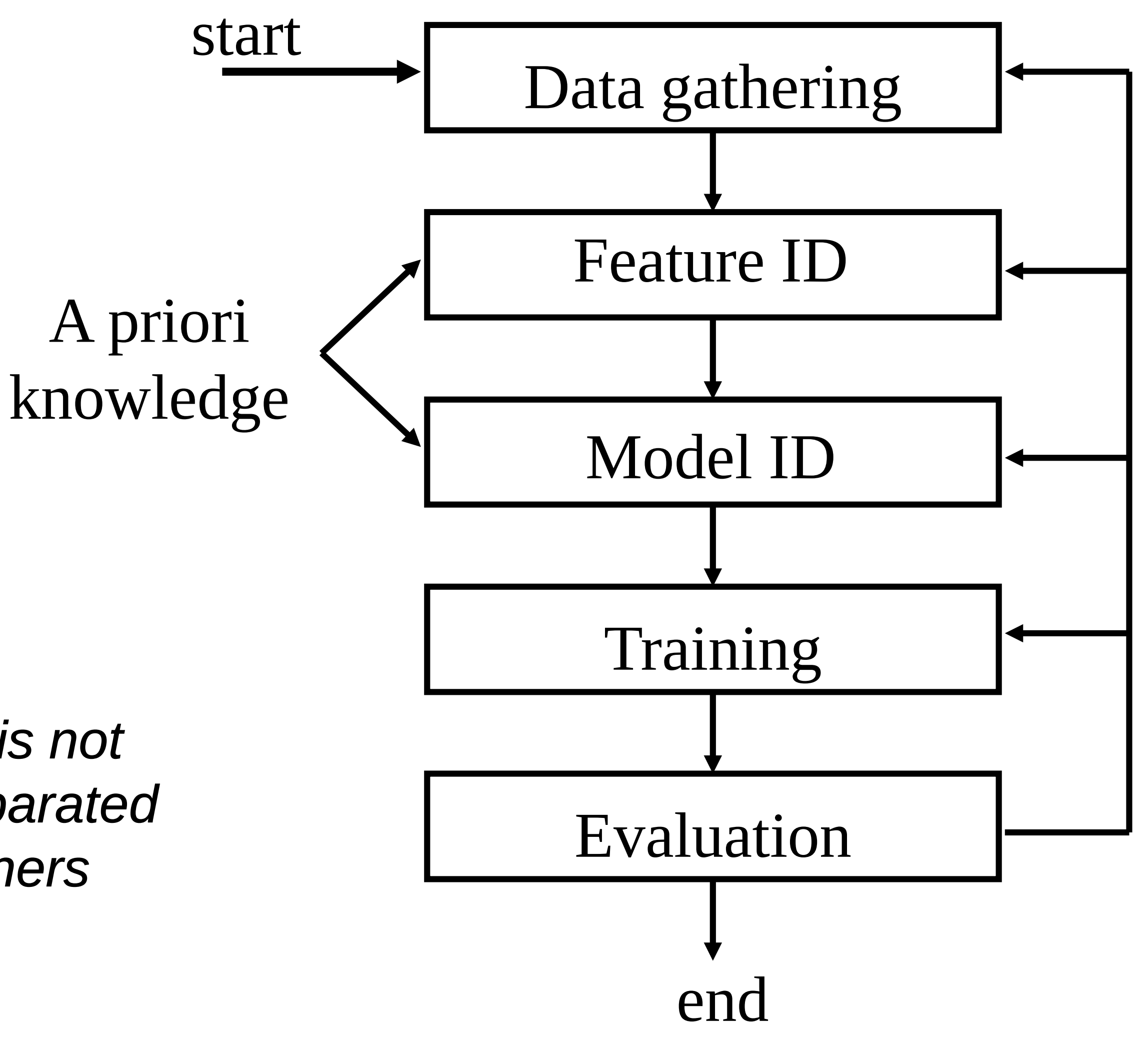
Approach

- Construction of a specific classifier for each type of defect:
  - Cracks
  - Bubbles
  - Splinters
  - others…

- Different views to be considered
  - Front
  - Shoulder
  - Bottom
  - Top



The usual classification pipeline



Each phase is not necessary separated from the others

Data gathering

- No samples of defects were readily available
- Defected samples were relatively rare
- Collection and acquisition of samples were a slow, manual, error-prone, labor-intensive task
- The contractor had no internal policy on vial rejection (a 0.2 mm bubble is a defect?)

→ All those factors made the acquisition phase the longer and most painful of all. *Do not underrate it!*

Data gathering phases

Phase	Samples per class	Defect classes	Views for vial	Acquisition method	Duration (weeks)
1	~30	3	1	Manual	2
2	~100	4	1	Semi-automatic	2
3	~400	4	~24	Automatic	4

How much data is “sufficient” and “representative”?

Feature selection

- The following features has been selected for this first batch of experiments:
  - raw image
  - gradient image
  - gabor transform
  - wavelet transform
  - Fourier transform
  - Harris operator
  - sift descriptors
  - spin images
  - histograms
  - laplacian and LoG
  - morphological operators

- Initial selection guided from domain knowledge
- Secondary selection based on early tests

The model ID and modus operandi

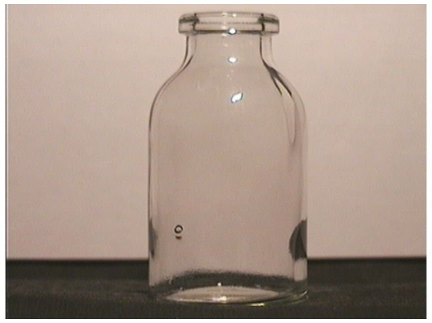
- Support Vector Machines

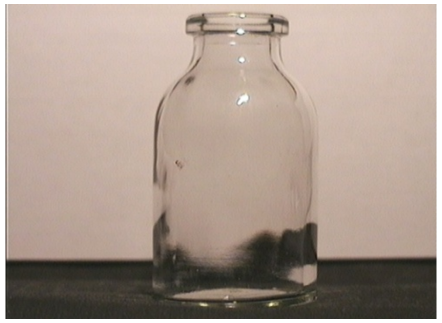
- Construction of a specific classifier for each type of defects to recognize

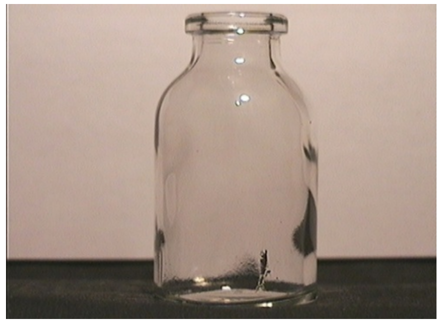
- Validation with a *leave-k-out* tests.

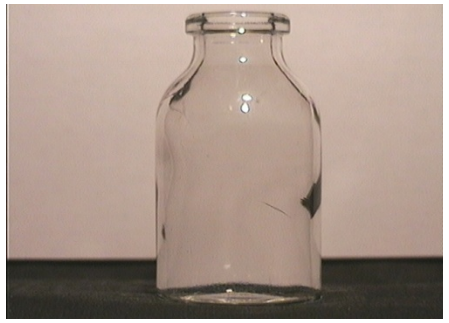
Results from Phase 1 (June ’05)

- Using ~40 manually acquired samples per class
- Defect centered in the vial and generally well visible
- Using normalized raw gray levels as features


bubble


hit

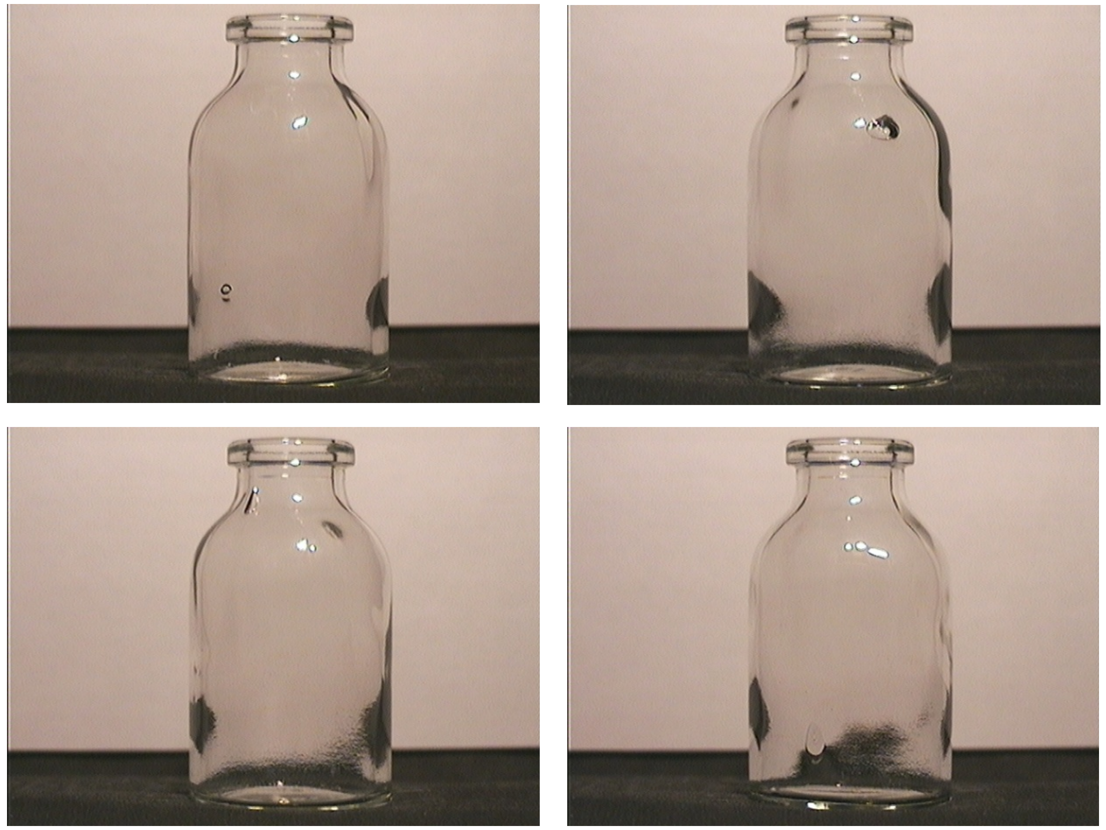

(big) hit


crease

Results from Phase 1: bubbles

- 60 good samples
- 30 bad samples

- Leave-1-out: 84.44 %

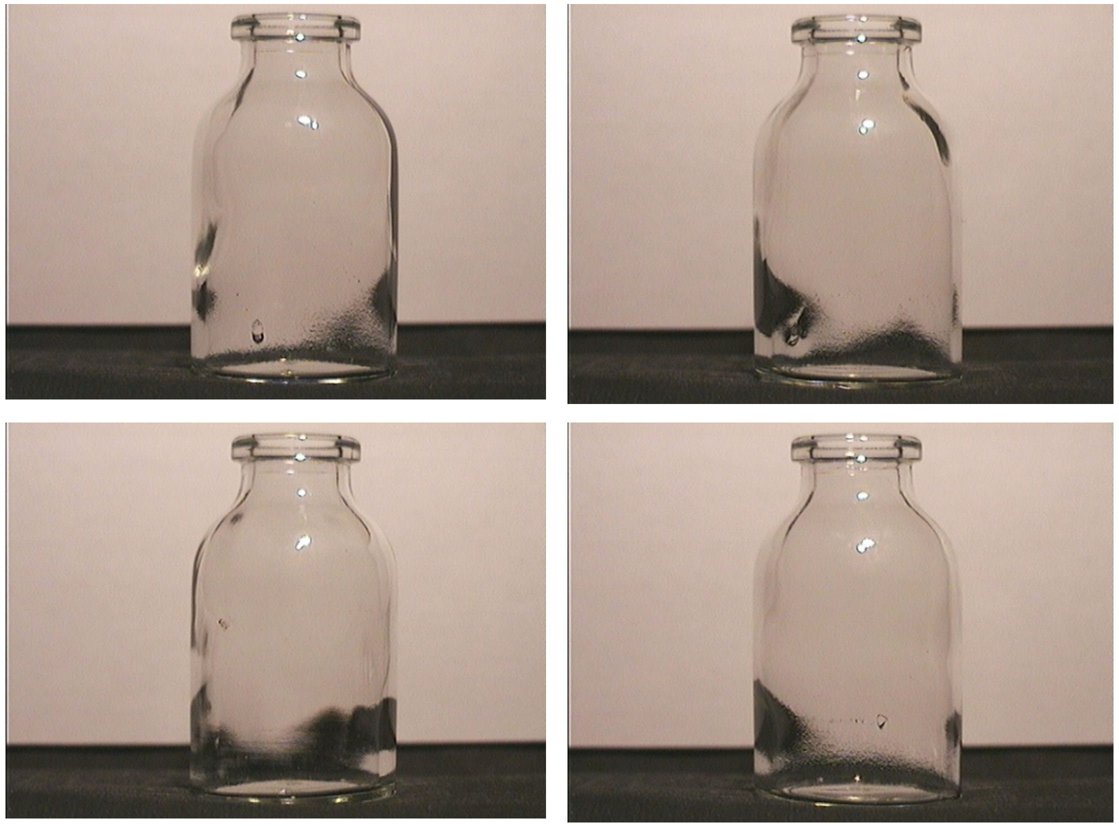


78

Results from Phase 1: hits

- 60 good samples
- 70 bad samples

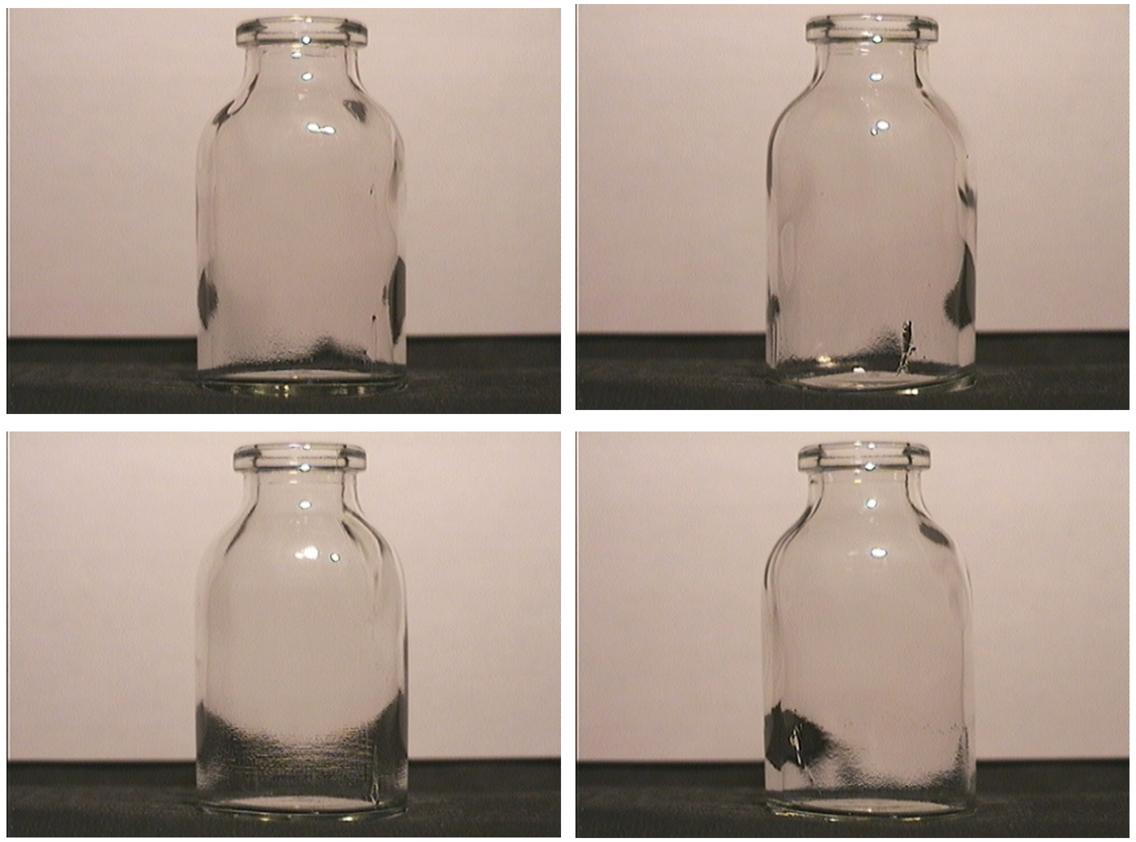
- Leave-1-out: 96.92 %



Results from Phase 1: hits (bigger)

- 60 good samples
- 40 bad samples

- Leave-1-out: 89.00 %

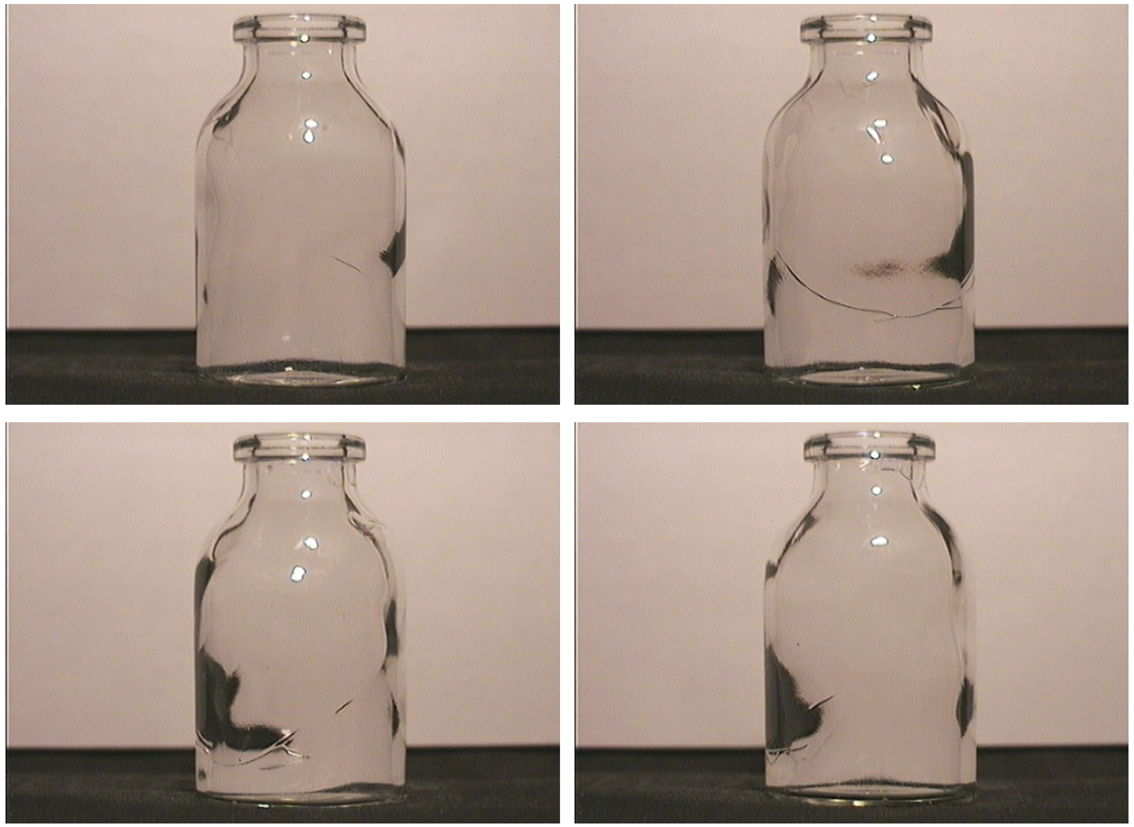


80

Results from Phase 1: creases

- 60 good samples
- 65 bad samples

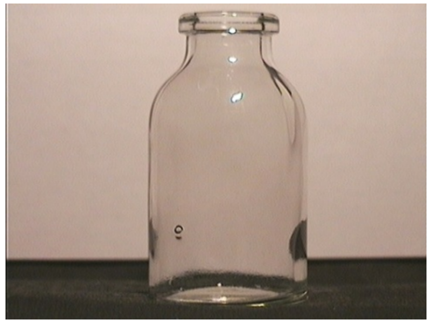
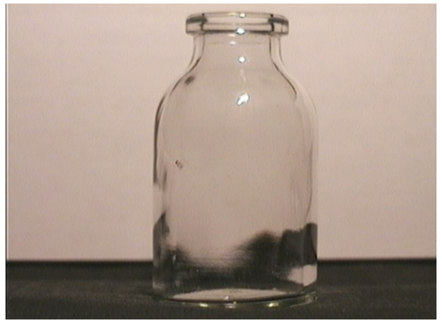
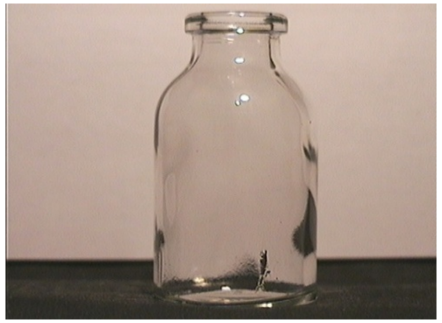
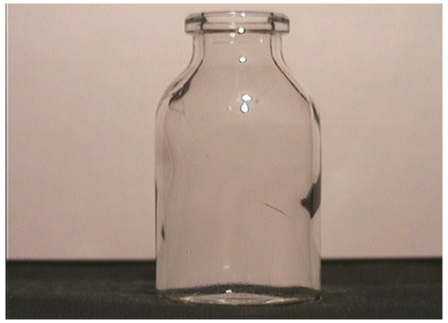
- Leave-1-out: 96.00 %



Results from Phase 1

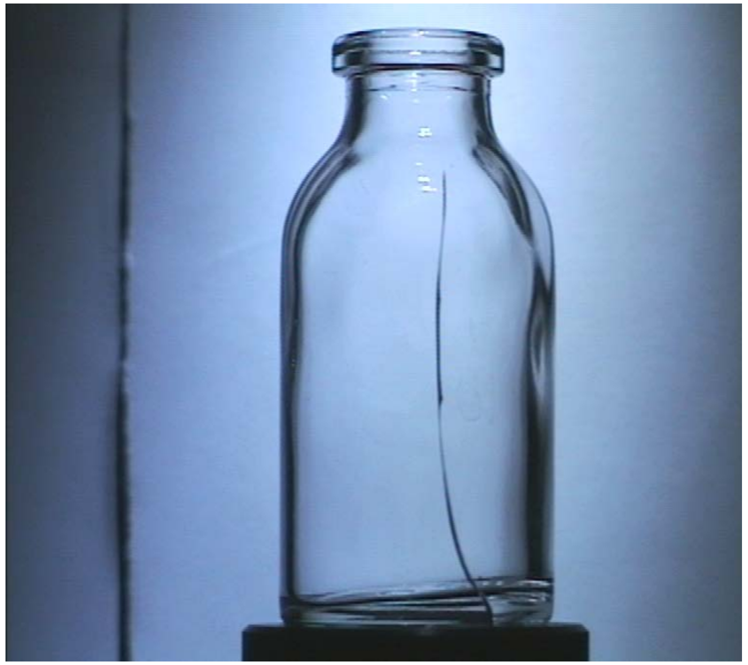
- Using ~40 manually acquired samples per class for training
- Defect centered in the vial and generally well visible
- Using normalized raw gray levels as features

Bubbles	Hits	Big hits	Creases
84.44%	96.92%	89.00%	96.00%

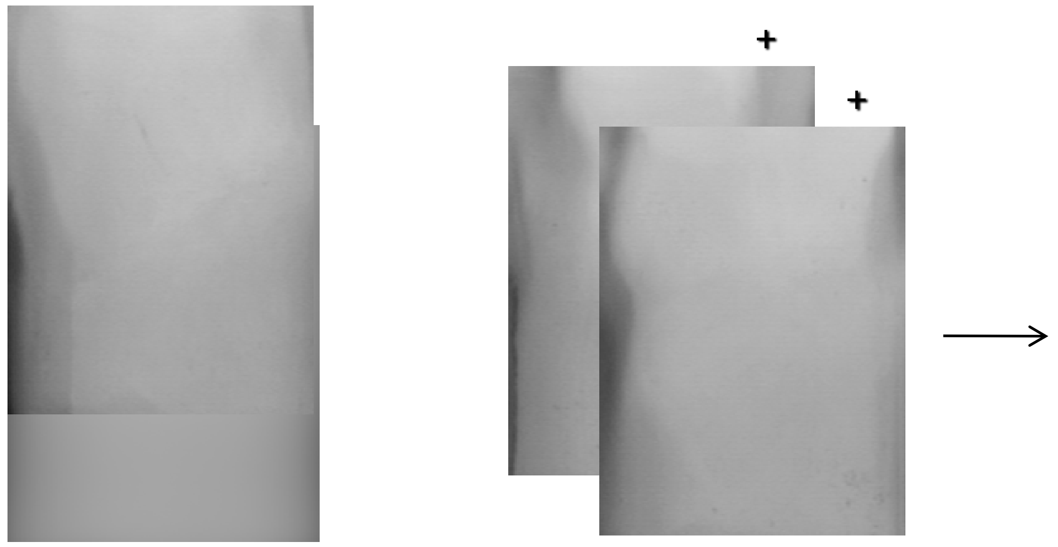





Results from Phase 2

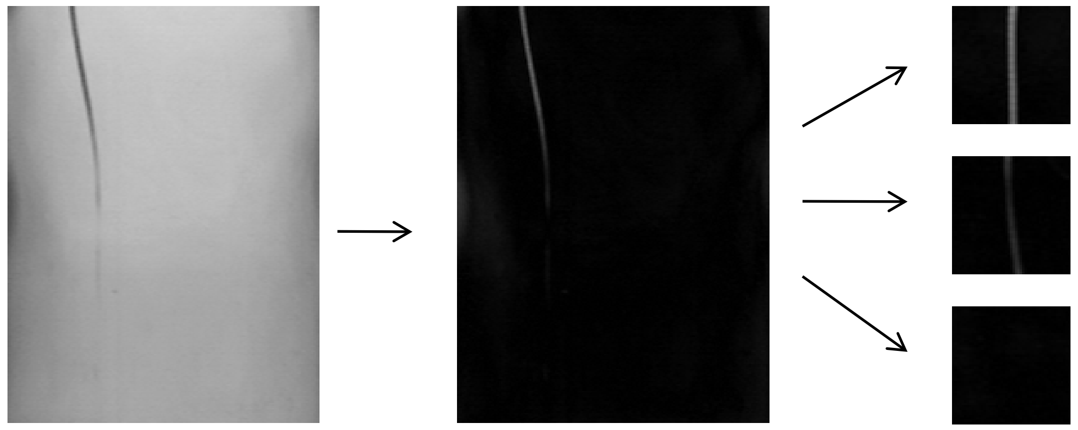
- Focus on the two most critical defects
  - Cracks
  - Bubbles
- ~100 samples for training, acquired semi-automatically
- Better illumination
- Now working on sub-windows



Feature extraction



Background modeling



Background subtraction

Sub-windows extraction

(manual task)

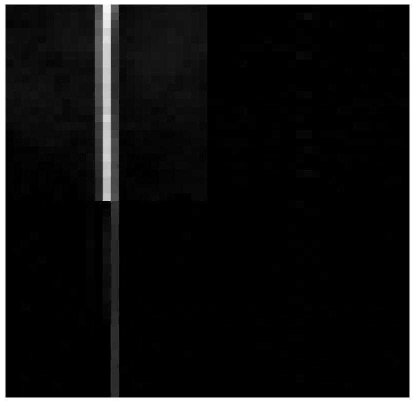
Results from Phase 2: cracks

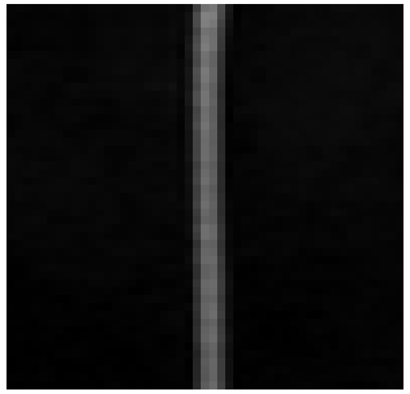
Gray level	98.54
Sobel	96.70
Harris	94.64
Log	97.69
Wavelet: Haar	98.76
Wavelet: Db2	98.22
Fourier	98.58
Dct	97.69

Cracks: best case analysis

- 98.76% with the Haar wavelet

Matching	98.76	
Positives	232 (41.21)	*correctly classified non-defects*
Negatives	324 (57.55)	*correctly classified defects*
False positives	4 (0.71)	*misclassified defects*
False negatives	3 (0.53)	*misclassified non-defects*





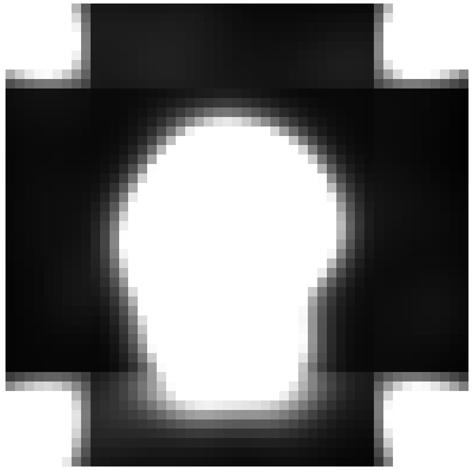
Results from Phase 2: bubbles

Gray level	94.56
Sobel	96.60
Harris	98.64
Log	90.48
Wavelet: Haar	95.24
Wavelet: Db2	94.92
Fourier	83.67
Dct	80.27

Bubbles: best case analysis

- 98.64% with the Harris operator

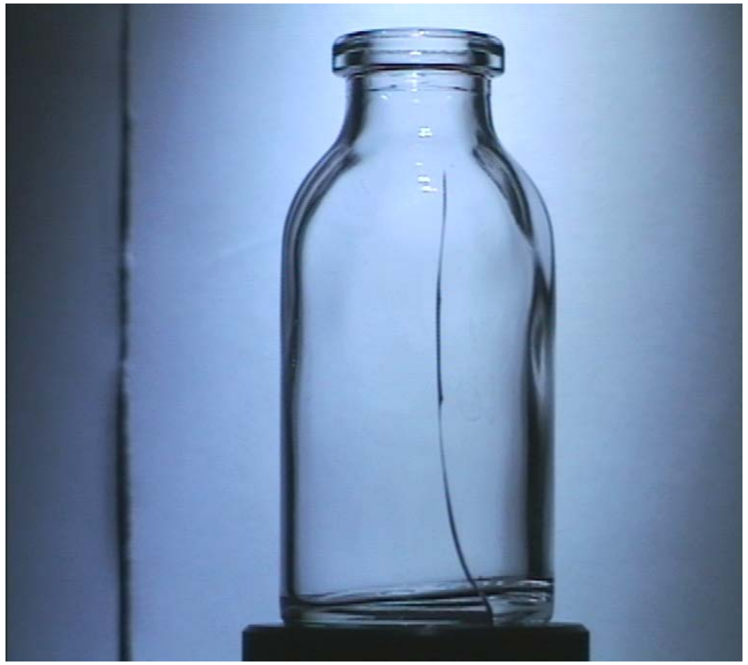
Matching	98.64	
Positives	102 (69.39)	*correctly classified non-defects*
Negatives	42 (28.57)	*correctly classified defects*
False positives	1 (0.68)	*misclassified defects*
False negatives	2 (1.36)	*misclassified non-defects*



Results from Phase 2

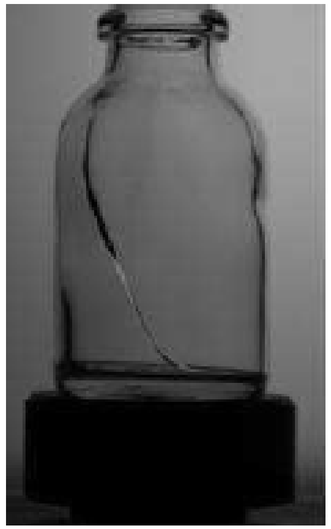
- ~100 samples for training
- Sub-windows
- Background subtraction

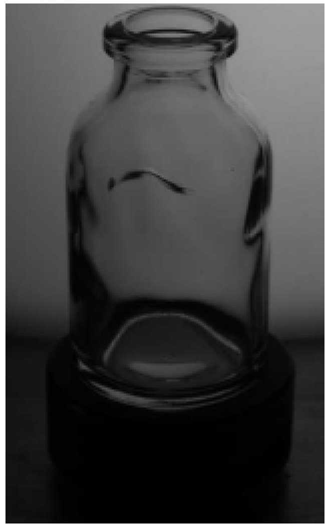
Cracks	Bubbles
98.76%	98.64%
Haar	Harris



Results from Phase 3

- ~400 samples per class for training, acquired automatically
- Features used
  - Histograms (ldg, gradient)
  - Morphological operators
- Working on sub-windows
- Additional (shoulder) view



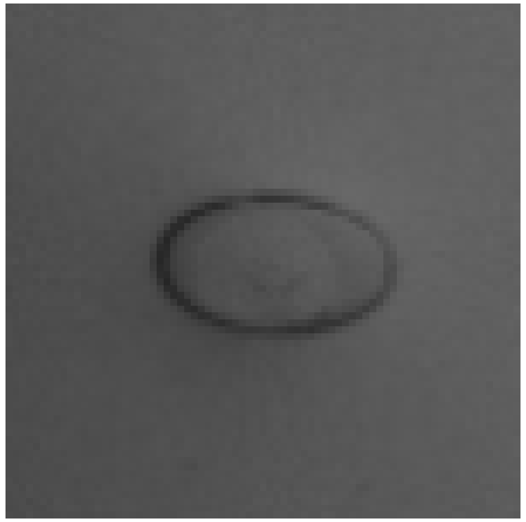


Big bubbles: best case analysis

- 95.76% with morphological operator on sub-windows

- Matching 95.76%

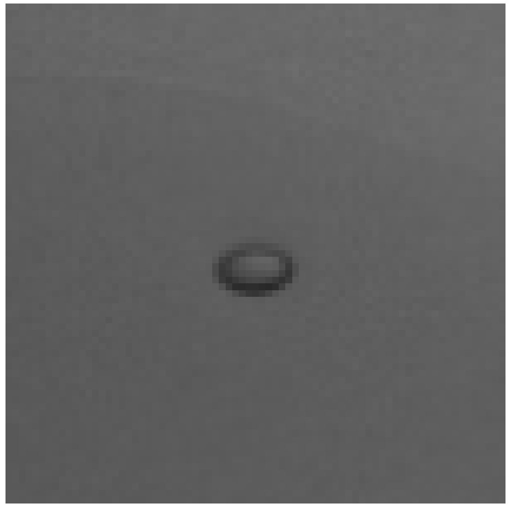
Positives	253 (48.75)	*correctly classified non-defects*
Negatives	244 (47.01)	*correctly classified defects*
False positives	17 (3.28)	*misclassified defects*
False negatives	5 (0.96)	*misclassified non-defects*



Small bubbles: best case analysis

- 91.07% with morphological operator on sub-windows

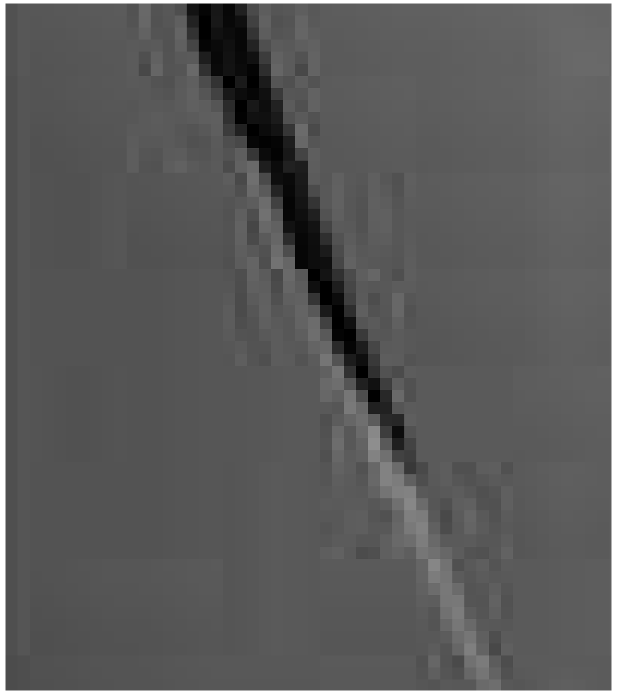
Matching	91.07%	
Positives	215 (47.99)	*correctly classified non-defects*
Negatives	193 (43.08)	*correctly classified defects*
False positives	9 (2.01)	*misclassified defects*
False negatives	31 (6.92)	*misclassified non-defects*



Cracks, front: best case analysis

- 91.91% with the gradient histogram

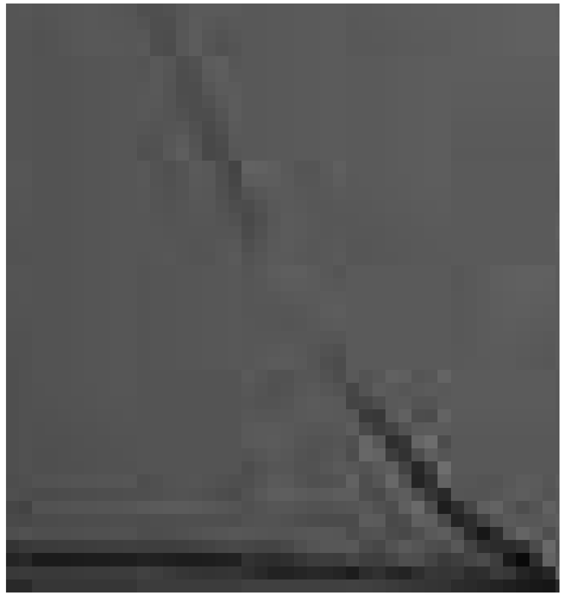
Matching	91.91%	
Positives	286 (52.57)	*correctly classified non-defects*
Negatives	214 (39.34)	*correctly classified defects*
False positives	14 (2.57)	*misclassified defects*
False negatives	30 (5.51)	*misclassified non-defects*



Cracks, shoulder: best case analysis

- 99.75% with morphological operator on sub-windows

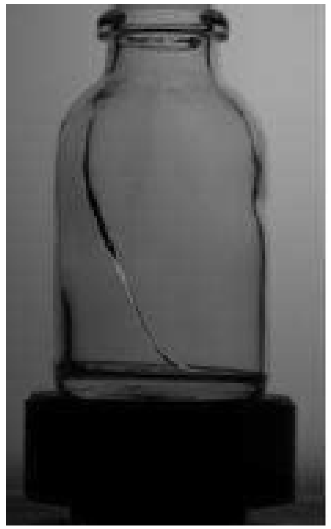
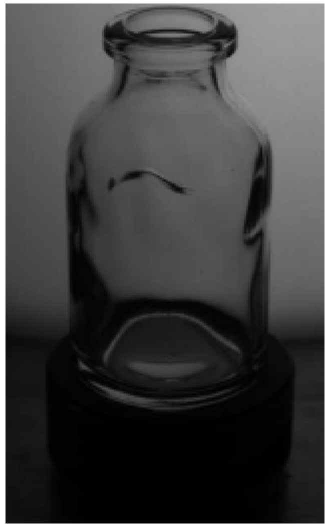
Matching	99.75%	
Positives	197 (49.75)	*correctly classified non-defects*
Negatives	198 (50.00)	*correctly classified defects*
False positives	1 (0.25)	*misclassified defects*
False negatives	0 (0.00)	*misclassified non-defects*



Results from Phase 3

- ~400 samples per class for training, acquired automatically

Cracks		Bubbles	
Front	Shoulder	Big	Small
91.91%	99.75%	95.76%	91.07%
GradHist	Hist	Morph	Morph

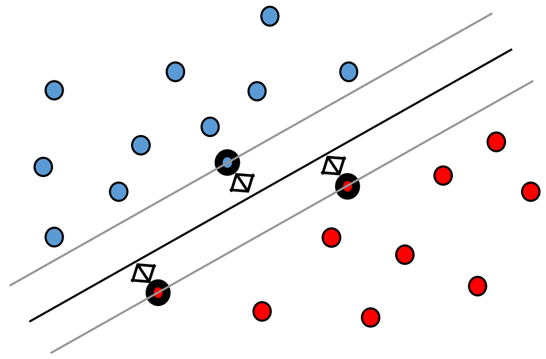



Comments

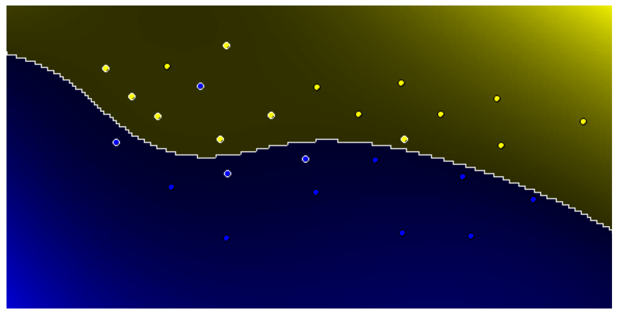
- Obtained results were satisfactory and in line with the customer requirements
- The acquisition phase is crucial (garbage in, garbage out)
- Support vector number was not monitored

Summary

- Support Vector Machines, two key ideas
  - Margin maximization
  - Kernel trick
- Training
  - a quadratic optimization problem
  - always convergent to the optimal solution
  - solvable in polynomial time
- Free parameters
  - The cost C
  - The kernel type
  - The kernel parameters



$$K(x_i, x_j) = \varphi(x_i) \cdot \varphi(x_j)$$



Advantages

- The SVM theory is an elegant and highly principled
- SVMs have a simple geometric interpretation
- The use of kernels provides a efficient solution to non-linear classification and dispels the ”curse of dimensionality”
- Convergence to the solution is guaranteed
- Support vectors give a compact representation of the entire dataset; their number is a measure of the generalization performance

Disadvantages

- Kernel and parameter choice is crucial
- Training can sometimes be tricky and time consuming
- Training is not incremental, the whole dataset must be processed for every new addition
- There are no optimized extension to multi-class problems: a problem with N classes requires N classifiers

References

An introduction to Support Vector Machines

(and other kernel-based learning methods)

N. Cristianini and J. Shawe-Taylor

Cambridge University Press (2000)

Support Vector and Kernel Machines

Tutorial presented at ICML, Nello Cristianini

A tutorial on support vector machines for pattern recognition

C. Burges, Data Mining and Knowledge Discovery, Vol 2(2), June 1998

http://www.kernel-machines.org/

http://www.support-vector.net/