Lecture 10: Support Vector Machines

SVM, Kernel Trick, Linear Separability

Ute Schmid

Cognitive Systems, Applied Computer Science, University of Bamberg
www.uni-bamberg.de/cogsys

Last change: February 17, 2021

Motivation

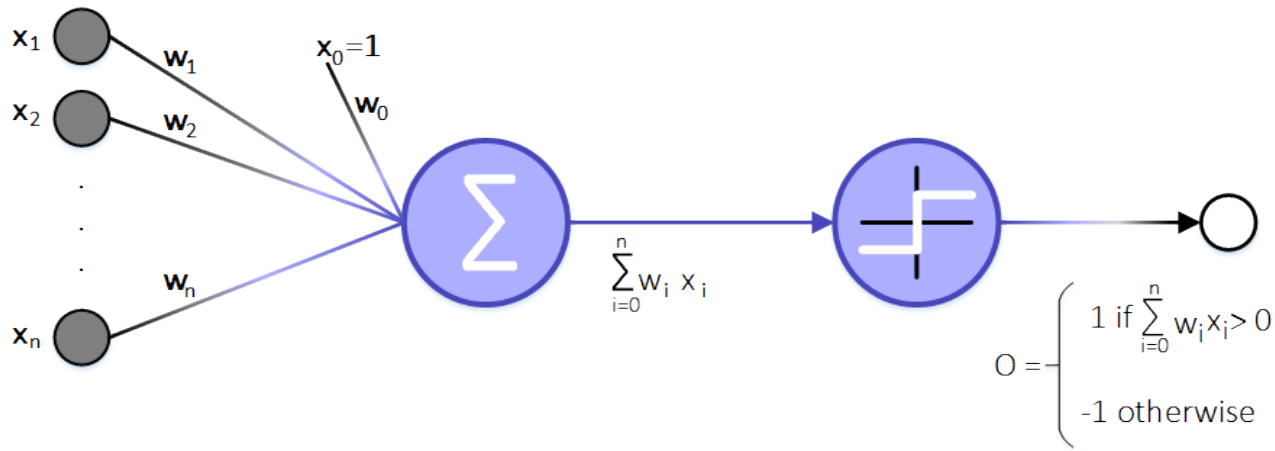
- linear classifiers (e.g. perceptrons): efficiently trainable, but low capacity and only for non-symbolic instances
- non-linear classifiers (e.g. NNs with hidden layers): high capacity, but high time complexity, local optima, overfitting, only numerical data
- idea of kernel methods: non-linearly embed instance space in high dimensional feature space where mapped training data is linearly separable
  - high capacity
  - applicable for symbolical data
  - time efficient training (polynomial with sample size)
  - global optimum
  - prevents overfitting

  (given a suitable kernel)

Outline

- Motivation
- Perceptron Revised
- Support Vector Machines
- Soft Margin
- The Kernel Trick
- Parameter Optimization
- Class Learning

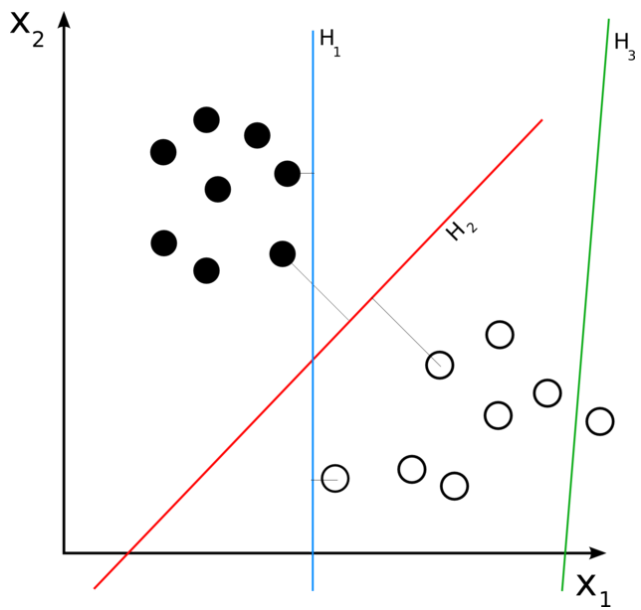
Perceptrons Revised



- real-valued input-vector $\vec{x} = \langle x_0, ..., x_n \rangle$
- calculates the linear combination of these inputs
  - weighted by $\vec{w} = \langle w_0, ..., w_n \rangle$
  - x_0 is always 1
  - $\vec{w} \cdot \vec{x} = \sum_{i=0}^{n} w_i x_i = w_0 x_0 + w_1 x_1 + ... + w_n x_n$
- outputs 1 if the result is greater than 0, otherwise -1

Perceptrons' Representational Power

- a perceptron represents a **hyperplane decision surface** in the n-dimensional space of instances
- some sets of examples cannot be separated by any hyperplane, those that can be separated are called **linearly separable**



Alternate Representations for the Hyperplane

Basics

$\mathbf{x} = \langle x_1, \ldots, x_n \rangle$
$w = \langle w_1, \ldots, w_n \rangle$

Vector Representation

$$o(\mathbf{x}_i) = \begin{cases} 1 & \text{if } w\mathbf{x}_i - b > 0 \\ -1 & \text{otherwise} \end{cases}$$

Note: b corresponds to $-w_0$ of the perceptron

Alternate Representations for the Hyperplane

Basics

$\mathbf{x} = \langle x_1, \ldots, x_n \rangle$
$w = \langle w_1, \ldots, w_n \rangle$

Vector Representation

$$o(\mathbf{x}_i) = \begin{cases} 1 & \text{if } w\mathbf{x}_i - b > 0 \\ -1 & \text{otherwise} \end{cases}$$

Note: b corresponds to $-w_0$ of the perceptron

Constraint Representation

$w\mathbf{x}_i - b > 0$ for positive instances $\mathbf{x}_i$
$w\mathbf{x}_i - b < 0$ for negative instances $\mathbf{x}_i$

Alternate Representations for the Hyperplane

Basics

$\mathbf{x} = \langle x_1, \ldots, x_n \rangle$
$w = \langle w_1, \ldots, w_n \rangle$

Vector Representation

$$o(\mathbf{x}_i) = \begin{cases} 1 & \text{if } w\mathbf{x}_i - b > 0 \\ -1 & \text{otherwise} \end{cases}$$

Note: b corresponds to $-w_0$ of the perceptron

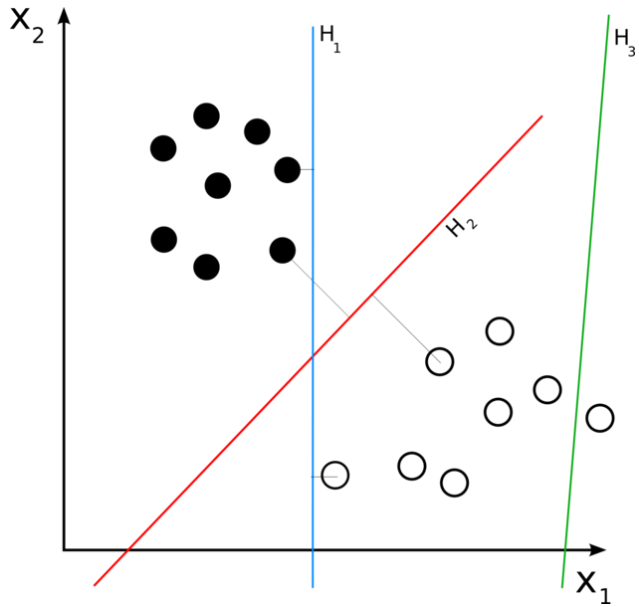
Constraint Representation

$w\mathbf{x}_i - b > 0$ for positive instances $\mathbf{x}_i$
$w\mathbf{x}_i - b < 0$ for negative instances $\mathbf{x}_i$

Unified Constraint Representation

$o(\mathbf{x}_i)(w\mathbf{x}_i - b) > 0$ for all instances $\mathbf{x}_i$

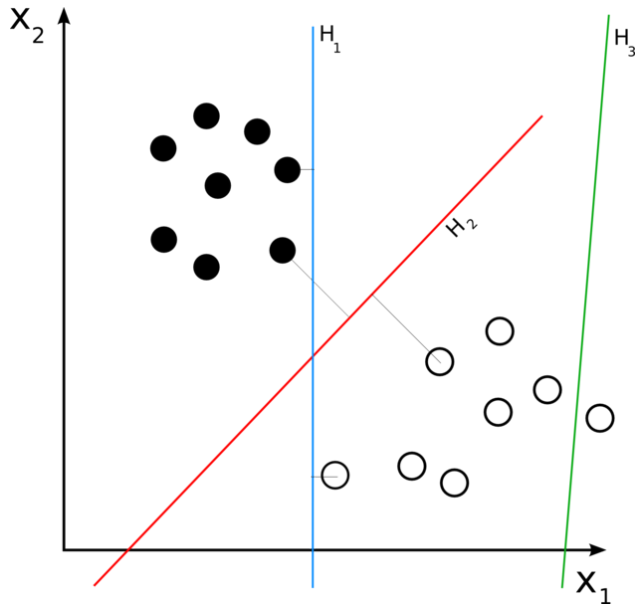
Initial Idea (Vladimir Vapnik)



Hypothesis' Language

w, b such that $o(\mathbf{x}_i) \cdot (w\mathbf{x}_i - b) > 0$

Initial Idea (Vladimir Vapnik)



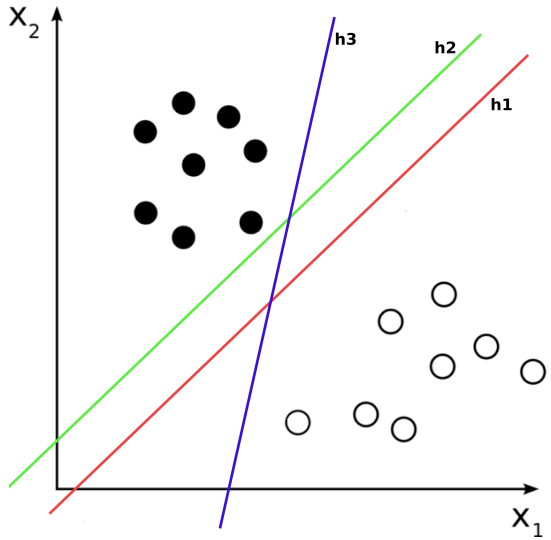
Hypothesis' Language

w, b such that $o(\mathbf{x}_i) \cdot (w\mathbf{x}_i - b) > 0$

Issue

The hyperplane not unique!

The Best Hyperplane

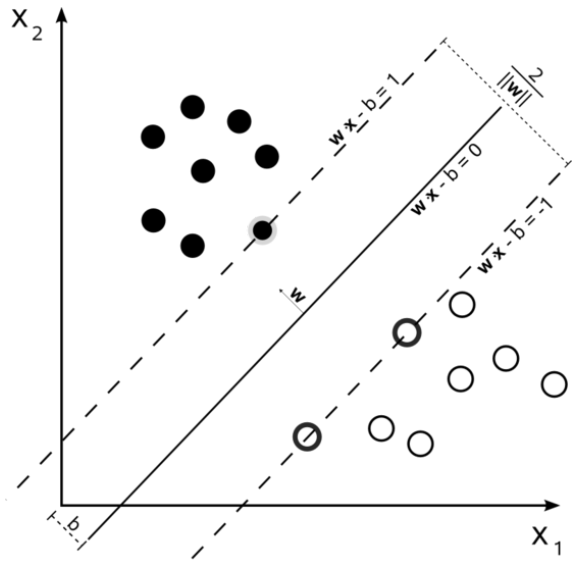


Question

Which of these hyperplanes is the best?

1. **h1** is the best hyperplane
2. **h2** is the best hyperplane
3. **h3** is the best hyperplane
4. **h1** and **h2** are both best hyperplanes
5. **h2** and **h3** are both best hyperplanes
6. More information is needed to answer the question

The Best Hyperplane



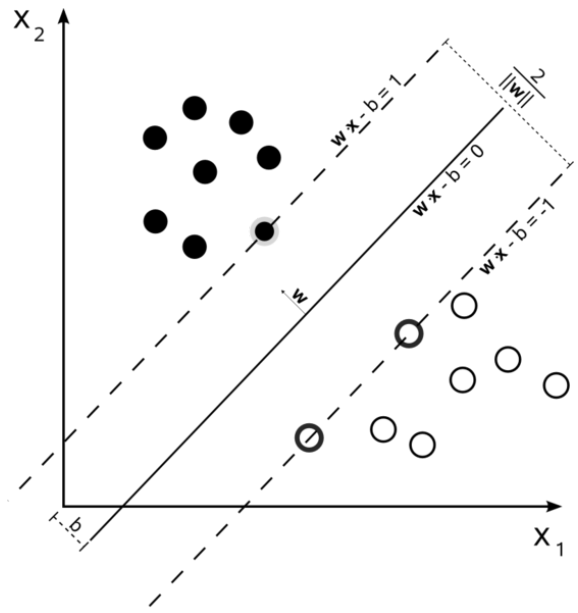
Best Hyperplane

The hyperplane separating all instances while being as far apart from the instances as possible.

Corridor

Define a Hyperplane Corridor such that $|(w\mathbf{x}_i) - b| = 1$ for the $\mathbf{x}_i$ closest to the hyperplane. The distance between the two hyperplanes is $\frac{2}{\|w\|}$.

Transforming a Hyperplane



- moving parallel:
  $c(\mathbf{x}_i) \cdot (w\mathbf{x}_i - (b + \nu)) = 0$
- scaleing:
  $\beta \cdot c(\mathbf{x}_i) \cdot (w\mathbf{x}_i - b) = 0$

Initial Idea (cont'd)

Support Vector Machine

A hyperplane $w\mathbf{x}_i - b$ such that

- $\frac{2}{\|w\|}$ is maximal and
- $c(\mathbf{x}_i)(w\mathbf{x}_i - b) \geq 1$ for all instances $\mathbf{x}_i$.

Initial Idea (cont'd)

Support Vector Machine

A hyperplane $w\mathbf{x}_i - b$ such that

- $\frac{2}{\|w\|}$ is maximal and
- $c(\mathbf{x}_i)(w\mathbf{x}_i - b) \geq 1$ for all instances $\mathbf{x}_i$.

Numerical Optimization

- Swarm Algorithms
- Genetic Algorithms
- Analytic Solutions

Alternate Problem Formulation

Idee

Minimize $\frac{1}{2}\|w\|^2$ such that $c(\mathbf{x}_i)(w\mathbf{x}_i - b) \geq 1$ for all $\mathbf{x}_i$:

$$\frac{1}{2}\|w\|^2 - \sum_i \alpha_i \left(1 - c\left(\mathbf{x}_i\right)\left(w\mathbf{x}_i - b\right)\right)$$

Alternate Problem Formulation

Idee

Minimize $\frac{1}{2}\|w\|^2$ such that $c(\mathbf{x}_i)(w\mathbf{x}_i - b) \geq 1$ for all $\mathbf{x}_i$:

$$\frac{1}{2}\|w\|^2 - \sum_i \alpha_i \left(1 - c\left(\mathbf{x}_i\right)\left(w\mathbf{x}_i - b\right)\right)$$

Support Vectors

The solution can be expressed as

$$w = \sum_{i=1}^{n} \alpha_i c(\mathbf{x}_i)\mathbf{x}_i$$

Only few α_i are not zero. The corresponding $\mathbf{x}_i$ are called the support vectors. The support vectors satisfy $c(\mathbf{x}_i)(w\mathbf{x}_i - b) = 1$, thus lie on the two parallel hyperplanes.

Soft Margins (Corinna Cortes and Vladimir Vapnik)

- data may contain noise
- constraint is relaxed:
  $c(\mathbf{x}_i)(w\mathbf{x}_i - b) \geq 1 - \xi_i$
- relaxation is penalized
  (violate the margin constraint as seldom as possible)
- ξ_i are called slack variables

Soft Margins (Corinna Cortes and Vladimir Vapnik)

- data may contain noise
- constraint is relaxed:
  $c(\mathbf{x}_i)(w\mathbf{x}_i - b) \geq 1 - \xi_i$
- relaxation is penalized
  (violate the margin constraint as seldom as possible)
- ξ_i are called slack variables

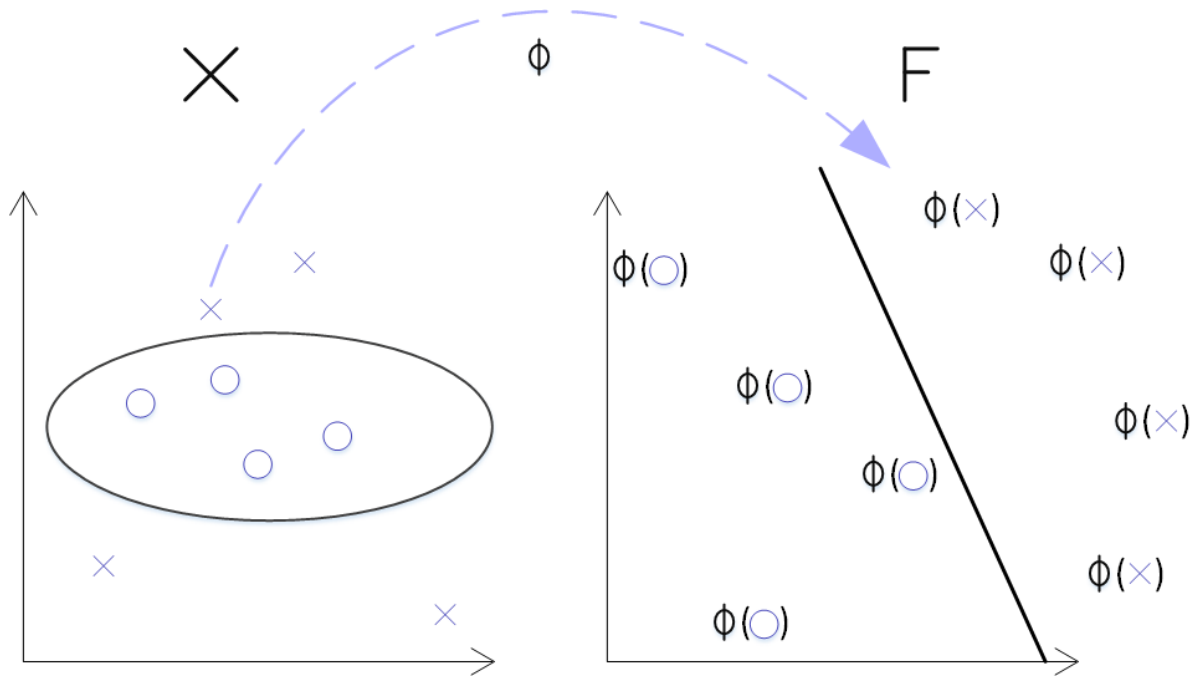
Soft Margin SVM

A hyperplane $w\mathbf{x}_i - b$ such that

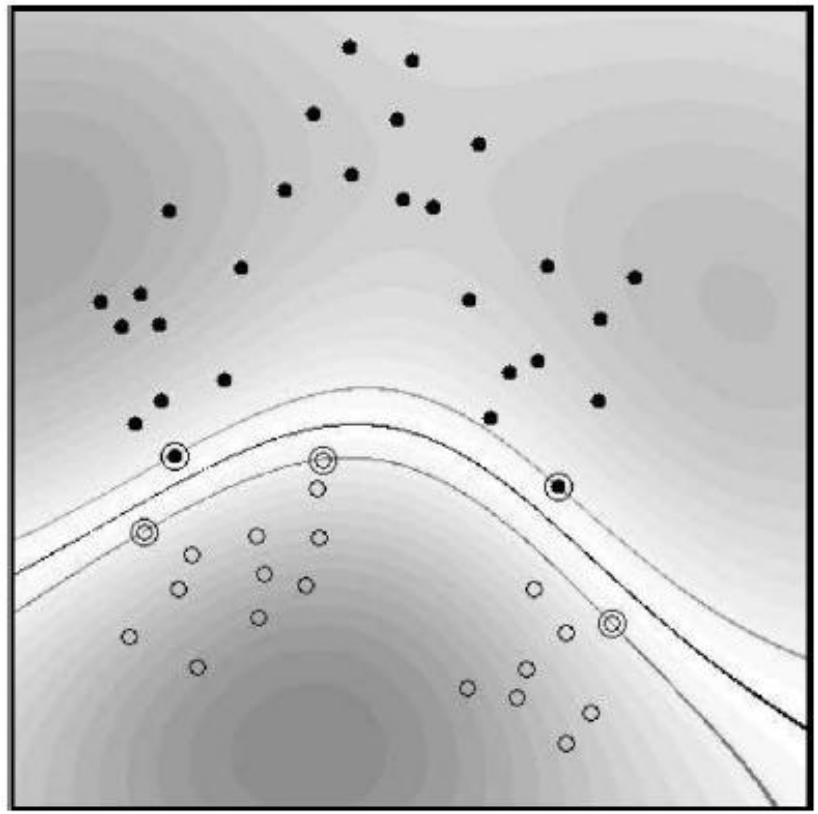
- $\frac{1}{2}\|w\|^2 - C\sum_{i=0}^{n}\xi_i$ is minimal (for large C margins become "harder") and
- $c(\mathbf{x}_i)(w\mathbf{x}_i - b) \geq 1 - \xi_i$ for all $\mathbf{x}_i$.

The Kernel Trick
(Bernhard Boser, Isabelle Guyon and Vladimir Vapnik)

- the data may not be linearly separable at all
- the solution is to transform the feature space: $\mathbf{x}_i \rightarrow \Phi(\mathbf{x}_i)$
- for example $\Phi\left(\begin{pmatrix} a \\ b \end{pmatrix}\right) = \begin{pmatrix} a^2 \\ 2ab \\ b^2 \end{pmatrix}$



The Kernel Trick (cont'd)



- as every vector occurs only inside a dot product it is not necessary to give Φ explicitly
- every dot-product is replaced by an application of a non-linear kernel function $\mathcal{K}(x_i, x_j)$

The Kernel Trick (cont'd)

Common Kernels

- Polynomial: $\mathcal{K}(x_i, x_j) = (x_i \cdot x_j)^d$
- Gaussian Radial Basis Function (RBF): $\mathcal{K}(x_i, x_j) = e^{-\gamma \left\| x_i - x_j \right\|^2}$
- Hyperbolic Tangent: $\mathcal{K}(x_i, x_j) = \tanh(\kappa x_i \cdot x_j + c)$

The Kernel Trick (cont'd)

Common Kernels

- Polynomial: $\mathcal{K}(x_i, x_j) = (x_i \cdot x_j)^d$
- Gaussian Radial Basis Function (RBF): $\mathcal{K}(x_i, x_j) = e^{-\gamma \left\| x_i - x_j \right\|^2}$
- Hyperbolic Tangent: $\mathcal{K}(x_i, x_j) = \tanh(\kappa x_i \cdot x_j + c)$

Kernel SVM

A hyperplane $w\Phi(\mathbf{x}_i) - b$ such that

- $\frac{1}{2} \left\| w \right\|^2$ is minimal
- $c(\mathbf{x}_i)(w \cdot \Phi(\mathbf{x}_i) - b) \geq 1$ for all $\mathbf{x}_i$

new classifications are calculated without explicit reference to Φ and to w by

$$o(\mathbf{x}) = sign(\sum_{i=1}^{n} \alpha_i c(\mathbf{x}_i) \mathcal{K}(\mathbf{x}, \mathbf{x}_i) + b)$$

Parameter Optimization

- free parameters are:
  - the cost factor C: the higher the more accurate instances are classified (during training)
  - the kernel function $\mathcal{K}$
  - the parameters of the kernel
- normally powers of 2 are used for C: $C \in \{2^{-5}, \ldots, 2^{15}\}$
- the same holds for γ when using a Gaussian RBF: $\gamma \in \{2^{-15}, \ldots, 2^{3}\}$

Class Learning

- support vector machines are designed for concept learning
- there is ongoing research how to handle class learning
- a single SVM approach (called multi-class SVM) tries to solve the optimization problem directly
- other approaches use more than one SVM in a "divide and conquer" manner
  - 1-against-1,
  - 1-against-all,
  - error correcting output codes (ECOC)

Divide and Conquer

1-against-1

- for each pair of classes one SVM is learned
- only examples of the two classes are used
- final classification is assigned by vote

A similar meta learner is available (e.g. for perceptrons).

Divide and Conquer

1-against-1

- for each pair of classes one SVM is learned
- only examples of the two classes are used
- final classification is assigned by vote

1-against-all

- for each class one SVM is trained
- the concept is whether the example belongs to the class or not
- the class of the SVM with the highest output $(w\mathbf{x} - b)$ is assigned as final classification

A similar meta learner is available (e.g. for perceptrons).

Divide and Conquer (cont'd)

ECOCs

- for each class a bit-string s_i is created
- every bit-string is of length L
- the Hamming-Distance between all bit-strings is maximized
- $L \leq 2^{(k-1)} - 1$ for k classes
- for each bit an SVM is trained
- final classification is assigned to the class with the closest bit-string

Learning Terminology

Support Vector Machine

Supervised Learning	unsupervised learning

Approaches:

Concept / Classification	Policy Learning
symbolic	**statistical** / neuronal network
inductive	analytical

Learning Strategy:	**learning from examples**
Data:	**metric features**/any (with the right kernel)
Target Values:	**concept**