Machine Learning

MIRI Master

Lluís A. Belanche

belanche@cs.upc.edu

Soft Computing Research Group
Dept. de Ciències de la Computació (Computer Science)
Universitat Politècnica de Catalunya

Spring Semester 2016-2017

LECTURE 8: Learning with kernels (I) - The SVM

Learning with kernels (I): The SVM

Linear regression revisited

Problem: We wish to find a function $y(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + w_0$ which best models a data set $D = \{(\boldsymbol{x}_1, t_1), \ldots, (\boldsymbol{x}_N, t_N)\} \subset \mathbb{R}^d \times \mathbb{R}$

- We define $\boldsymbol{w} = (w_0, w_1, \ldots, w_d)^\mathsf{T}$ and $\boldsymbol{x} = (1, x_1, \ldots, x_d)^\mathsf{T}$

- Then we minimize the regularized (aka penalized) empirical error:

$$E^\lambda_{\text{emp}}(y) = \sum_{n=1}^{N} \Big(t_n - y(\boldsymbol{x}_n)\Big)^2 + \lambda \sum_{i=0}^{d} w_i^2 = \|\boldsymbol{t} - X\boldsymbol{w}\|^2 + \lambda \|\boldsymbol{w}\|^2$$

  The parameter $\lambda > 0$ defines a trade-off between the fit to the data and the complexity of the vector $\boldsymbol{w}$

Learning with kernels (I): The SVM

Linear regression revisited

Setting $\frac{\partial E^{\lambda}_{\text{emp}}(y)}{\partial \boldsymbol{w}} = 0$, we obtain the (regularized) normal equations:

$$-2X^{\mathsf{T}}(\boldsymbol{t} - X\boldsymbol{w}) + 2\lambda \boldsymbol{w} = 0$$

with solution

$$\widehat{\boldsymbol{w}} = (X^{\mathsf{T}}X + \lambda I_d)^{-1} X^{\mathsf{T}}\boldsymbol{t}$$

and therefore

$$y(\boldsymbol{x}) = \widehat{\boldsymbol{w}}^{\top}\boldsymbol{x}$$

Learning with kernels (I): The SVM

Linear regression revisited

It turns out that the regularized solution can be written as:

$$\widehat{\boldsymbol{w}} = \sum_{n=1}^{N} \alpha_n \boldsymbol{x}_n, \qquad \widehat{\boldsymbol{w}} = \begin{pmatrix} \widehat{w}_1 \\ \widehat{w}_2 \\ \vdots \\ \widehat{w}_d \end{pmatrix}$$

$$y(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_n (\boldsymbol{x}_n{}^\top \boldsymbol{x}), \qquad \boldsymbol{\alpha} = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{pmatrix}$$

The new vector of parameters is given by $\boldsymbol{\alpha} = (XX^\top + \lambda I_N)^{-1}\boldsymbol{t}$

Learning with kernels (I): The SVM

Linear regression revisited

So we have the **primal** and the **dual** forms for $y(\boldsymbol{x})$:

$$y(\boldsymbol{x}) = \widehat{\boldsymbol{w}}^\top \boldsymbol{x} \quad \text{and} \quad y(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_n (\boldsymbol{x}_n{}^\top \boldsymbol{x})$$

The dual form is usually more convenient when $d \gg N$:

- the primal requires the computation & inversion of $X^\mathsf{T} X + \lambda I_d$, requiring $O(Nd^2 + d^3)$ operations
- the dual requires the computation & inversion of $XX^\mathsf{T} + \lambda I_N$, requiring $O(dN^2 + N^3)$ operations

Learning with kernels (I): The SVM

Key aspects of kernel methods

How can we achieve non-linear regression?

A **feature map** is a function $\phi : \mathbb{R}^d \to \mathbb{R}^M$:

$$\phi(\boldsymbol{x}) = \Big(\phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{x}), \cdots, \phi_M(\boldsymbol{x})\Big)^\mathsf{T}$$

- $\phi(\boldsymbol{x})$ is called the **feature vector**
- $\{\phi(\boldsymbol{x}) : \boldsymbol{x} \in \mathbb{R}^d\}$ is the **feature space** (FS), and typically $M \gg d$.

Learning with kernels (I): The SVM

Key aspects of kernel methods

- Define $\Phi_{N\times M}$ the matrix of the $\phi(\boldsymbol{x}_n)$ as

$$\phi_{nm} = \phi_m(\boldsymbol{x}_n), n = 1, \ldots, N, m = 1, \ldots, M.$$

- Suppose we perform ridge regression on the Φ matrix

- The new regression function has the **primal** representation:

$$y(\boldsymbol{x}) = \widehat{\boldsymbol{w}}^\top \phi(\boldsymbol{x})$$

Note the primal now operates in feature space

Learning with kernels (I): The SVM

Key aspects of kernel methods

Given a feature map $\phi : \mathbb{R}^d \to \mathbb{R}^M$, we define its associated **kernel function** $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ as:

$$k(\boldsymbol{u}, \boldsymbol{v}) = \phi(\boldsymbol{u})^\top \phi(\boldsymbol{v}), \qquad \boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^d$$

- The (feature) space where k implicitly operates is $\mathbb{R}^M$
- For some feature maps, computing $k(\boldsymbol{u}, \boldsymbol{v})$ is independent of M (it depends only on d).

Learning with kernels (I): The SVM

Key aspects of kernel methods

Since $\widehat{\boldsymbol{w}} = \sum_{n=1}^{N} \alpha_n \phi(\boldsymbol{x}_n)$, the new regression function has the **dual** representation:

$$y(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_n (\phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x})) = \sum_{n=1}^{N} \alpha_n k(\boldsymbol{x}_n, \boldsymbol{x})$$

The new vector of parameters is given by

$$\boldsymbol{\alpha} = (\mathbf{K} + \lambda I_N)^{-1} \boldsymbol{t}, \qquad \text{where } \mathbf{K} = \Big(k(\boldsymbol{x}_n, \boldsymbol{x}_m)\Big)$$

Kernel-Based Learning

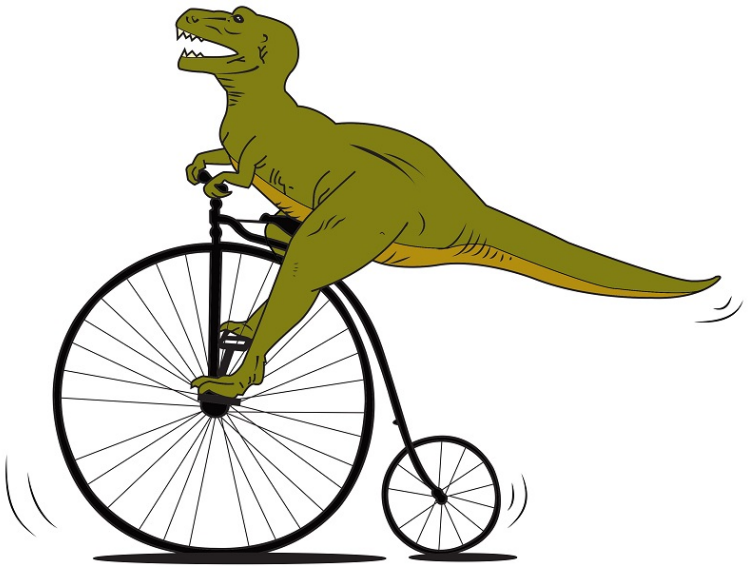
Example

What if we take the (simplest) choice $\phi(\boldsymbol{x}) = \boldsymbol{x}$?

$\Longrightarrow$ in this case $d = M$ and $k(\boldsymbol{u}, \boldsymbol{v}) = \langle \boldsymbol{u}, \boldsymbol{v} \rangle$

Then $\mathbf{K} = XX^{\mathsf{T}}$ and we are back to where we started

We thus have a "generalized linear model" (in a different sense than with GLMs)



Is this efficient or is it as out of date as a dinosaur riding a penny farthing?

Kernel-Based Learning

Key aspects of kernel methods

- A feature map is of the general form $\phi : \mathcal{X} \to \mathcal{H}$. The associated kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is $k(\boldsymbol{u}, \boldsymbol{v}) = \langle \phi(\boldsymbol{u}), \phi(\boldsymbol{v}) \rangle_{\mathcal{H}}, \ \boldsymbol{u}, \boldsymbol{v} \in \mathcal{X}$

- $\mathcal{X}$ can be any space, $\mathcal{H}$ is any **Hilbert space**:

  - An abstract complete vector space possessing the structure of an inner product

  - Examples would be $\mathbb{R}^M$ or the l_2 space of square-summable sequences

---

In our previous discussion, $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{H} = \mathbb{R}^M$

Learning with kernels (I): The SVM

Key aspects of kernel methods

Kernel-based methods consist of two ingredients:

1. The kernel function (this is non-trivial)

2. The algorithm taking kernels as input

   - Data items are embedded into a vector space (feature space FS)
   - Linear relations are sought among the elements of the FS
   - The coordinates of these images are not needed: only their pairwise inner products
   - These inner products can sometimes be computed efficiently and implicitly in the input space (kernel function)
   - The solution vector is expressed as a linear combination of the kernel centered at the data

Learning with kernels (I): The SVM

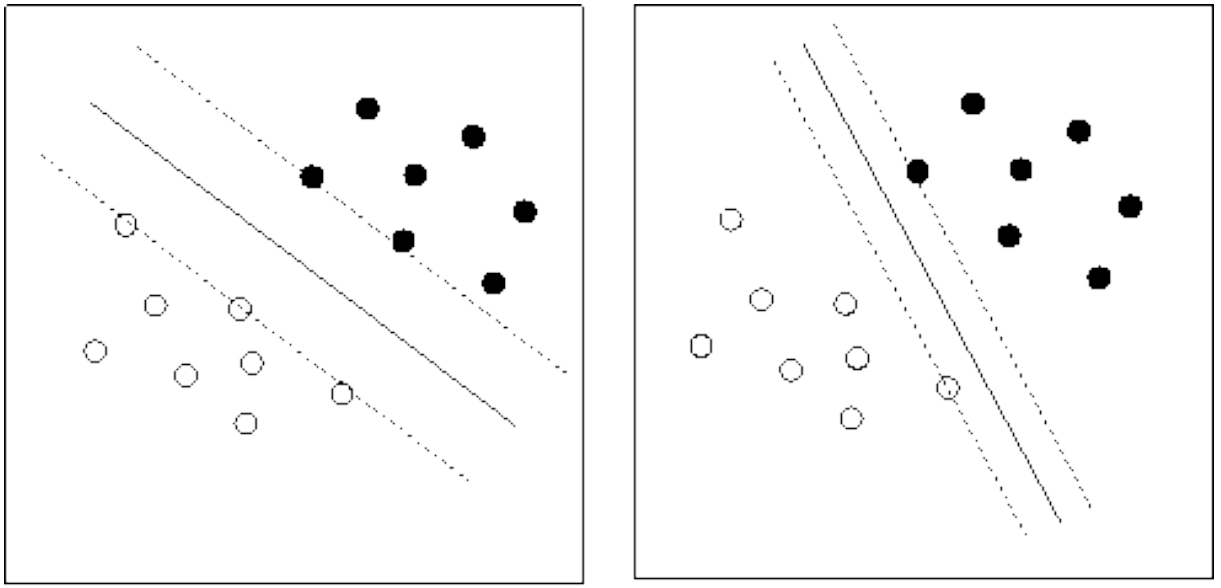
Key aspects of kernel methods

Many (classical and new) learning algorithms can be "kernelized":

- The Support Vector Machine (SVM) and the Relevance Vector Machine (RVM)
- Fisher Discriminant Analysis (KFDA), Principal Components Analysis (KPCA), Canonical Correlation Analysis (KCCA), ...
- Kernel (regularized) linear regression
- Kernel k-means, kernel kNN
- (less known or very recent): PLS, Parzen Windows, logistic regression, statistical tests, ...

Support Vector Machines

Preliminaries



(a) Larger margin (b) Smaller margin

Which solution is more likely to lead to better generalization?

Support Vector Machines

Preliminaries

Working Hypothesis (intuition):

The larger the margin, the better the generalization

- Criterion for building a two-class classifier:

  Maximize the **width of the margin** between the classes

- **margin** = empty area around the decision boundary, defined by the distance to the nearest training examples

  These examples will be called the **support vectors**

Goal: find the hyperplane (linear boundary) with the **largest margin**

Support Vector Machines

Formalisation

We have a data set $D = \{(\boldsymbol{x}_1, t_1), \ldots, (\boldsymbol{x}_N, t_N)\}$, with $\boldsymbol{x}_n \in \mathbb{R}^d$ and $t_n \in \{-1, +1\}$, describing a two class problem

We wish to find a function $y(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + w_0$ which best models D

- We would like to find $\boldsymbol{w}, w_0$ such that:

  $\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + w_0 > 0$, when $t_n = +1$
  $\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + w_0 < 0$, when $t_n = -1$

- In short, $t_n(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + w_0) > 0$, or $t_n\, y(\boldsymbol{x}_n) > 0, \qquad 1 \leq n \leq N$

Support Vector Machines

Formalisation

- The quantity $t_n\, y(\boldsymbol{x}_n)$ is the **functional** margin of $\boldsymbol{x}_n$ (there will be an "error" whenever $t_n\, y(\boldsymbol{x}_n) < 0$)

- Define the **loss** $L(t_n, \langle \boldsymbol{w}, \boldsymbol{x}_n \rangle) = \text{máx}(1 - t_n\, y(\boldsymbol{x}_n), 0)$

- Given the plane $\pi : y(\boldsymbol{x}) = 0$ or $\pi : \langle \boldsymbol{w}, \boldsymbol{x} \rangle + w_0 = 0$, the distance $d(\boldsymbol{x}, \pi) = \frac{|y(\boldsymbol{x})|}{\|\boldsymbol{w}\|}$ is called the **geometrical margin** of $\boldsymbol{x}$

- The optimal separating hyperplane (OSH) is the one that maximizes the geometrical margin for linearly separable data:

$$\underset{\boldsymbol{w},b}{\text{máx}} \left\{ \underset{1 \le n \le N}{\text{mín}}\, d(\boldsymbol{x}_n, \pi) \right\} \qquad \text{subject to } t_n\, (\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + w_0) > 0 \ (1 \le n \le N)$$

Support Vector Machines

Formalisation

- Rescaling $\boldsymbol{w}, b$ such that $|\langle \boldsymbol{w}, \boldsymbol{x} \rangle + w_0| = 1$ for the points closest to the hyperplane, we obtain $|\langle \boldsymbol{w}, \boldsymbol{x} \rangle + w_0| \geq 1$

- The **support vectors** (SV) are the $\boldsymbol{x}_n$ such that $|\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + w_0| = 1$

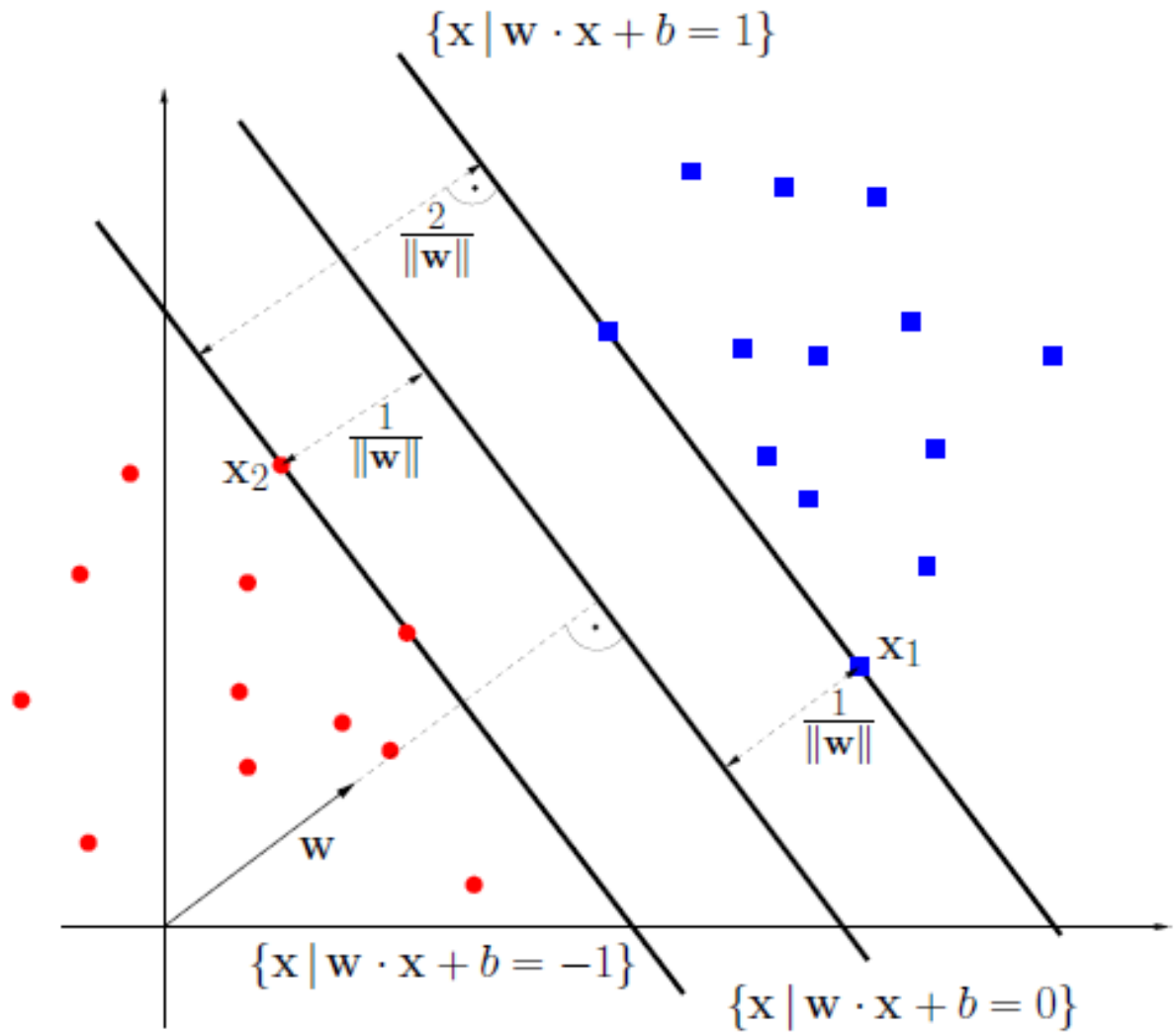
- The new loss is máx$(1 - t_n\, y(\boldsymbol{x}_n), 0) = (1 - t_n\, y(\boldsymbol{x}_n))_+$ (**hinge loss**)

- The **margin** of the OSH is twice the distance to any SV:

  $\text{margin(OSH)} = 2\, d(\boldsymbol{x}_{\text{SV}}, \text{OSH}) = \dfrac{2}{\|\boldsymbol{w}\|}$, since $|y(\boldsymbol{x}_{\text{SV}})| = 1$

Support Vector Machines

Geometrical view



Support Vector Machines

A look on what's to come

1. We find the **canonical OSH** by solving

$$\underset{\boldsymbol{w},b}{\text{máx}} \left\{ \frac{2}{\|\boldsymbol{w}\|} \quad / \quad t_n \left(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + w_0 \right) \geq 1, \qquad 1 \leq n \leq N \right\}$$

2. The solution will be $\boldsymbol{w} = \sum_{n=1}^{N} t_n \alpha_n \boldsymbol{x}_n$, with $\alpha_n \geq 0$ (the **dual** form)

3. A fraction of the training $\boldsymbol{x}_n$ will have $\alpha_n = 0$ (**sparsity**); the $\boldsymbol{x}_n$ for which $\alpha_n > 0$ will coincide with the **support vectors**

4. The **discriminant function** will be written

$$y_{\text{SVM}}(\boldsymbol{x}) = \text{sgn}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + w_0) = \text{sgn} \left(\sum_{n=1}^{N} t_n \alpha_n \langle \boldsymbol{x}, \boldsymbol{x}_n \rangle + w_0 \right)$$

Support Vector Machines

Using the VC dimension for two-class classification

For a two-class classifier, the **VC dimension** is the maximum number k of points that can be separated in all possible 2^k ways (*shattered*) by using functions representable by the classifier.

- Note it is *sufficient* that one set of k points exists that can be shattered for the VC dimension to be at least k
- If the VC dimension of a class is k, this means there is at least one set of k points that can be shattered by members of the class. It does not mean that every set of k points can be shattered.
- If no set of $k+1$ points can be shattered by members of the class, then the VC dimension of the class is less than $k+1$.

Support Vector Machines

Using the VC dimension for two-class classification

Theorem (Vapnik and Chervonenkis, 1974). Let D be an i.i.d data sample of size N and $\mathcal{Y}$ a class of parametric binary classifiers. Let ϑ denote the VC dimension of $\mathcal{Y}$. Take $y \in \mathcal{Y}$ with empirical error $R_{\text{emp}}(y)$ on D. For all $\eta > 0$ it holds true that, with probability at least $1 - \eta$, the true error of y is bounded by:

$$R(y) \leq R_{\text{emp}}(y) + H(N, \vartheta, \eta)$$

where

$$H(N, \vartheta, \eta) = \sqrt{\frac{\vartheta(\ln(2N/\vartheta) + 1) - \ln(\eta/4)}{N}}$$

Support Vector Machines

More than an intuition

- Separating hyperplanes in $\mathbb{R}^d$ have VC dimension $d + 1$
- When we use a feature map into a very high dimension $M \in (\mathbb{N} \cup \{\infty\})$, VC dimension will grow accordingly
- If we bound the margin of the hyperplanes, we limit VC dimension (therefore, we have an explicit control on complexity)

Support Vector Machines

More than an intuition

Theorem. Consider canonical hyperplanes $y(\boldsymbol{x}) = \text{sgn}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + w_0)$ w.r.t. a set of training patterns X (where $x_n \in \mathbb{R}^d$).

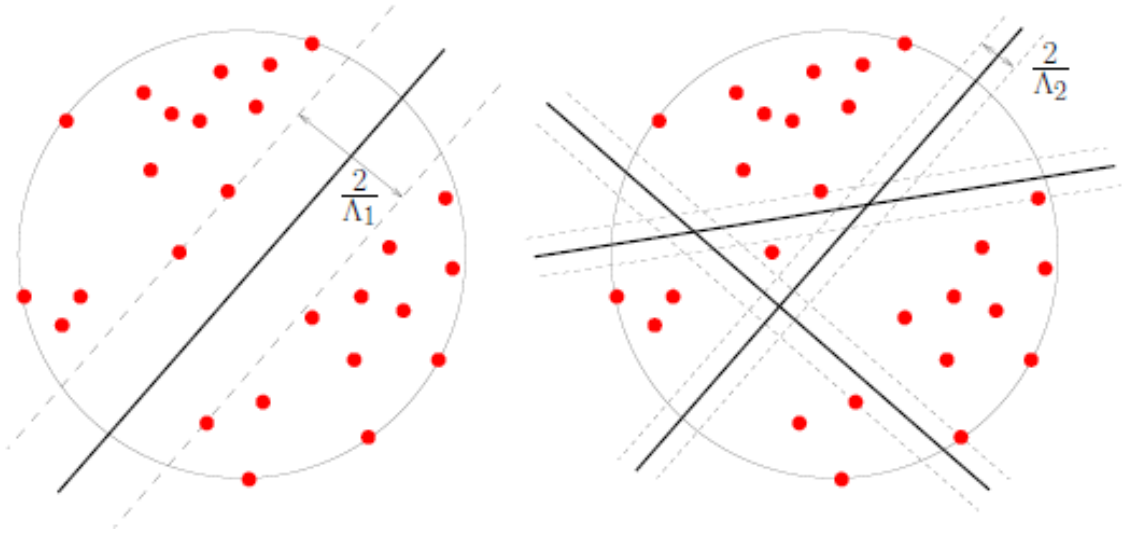
The subclass of linear classifiers with margin $m \geq m_0$ has VC dimension k bounded by

$$k \leq \text{mín} \left(\left\lceil \frac{R^2}{m_0^2} \right\rceil, d \right) + 1$$

where R is the radius of the smallest sphere centered at the origin containing X.

Support Vector Machines

More than an intuition



- Left: hyperplanes with a large margin have reduced chances to separate the data (the VC dimension is small)

- Right: smaller margins allow more separating hyperplanes (the VC dimension is large)

Support Vector Machines

Resolution

$$\textbf{minimize} \qquad \frac{1}{2}\|\boldsymbol{w}\|^2$$

$$\textbf{subject to} \quad t_n\left(\langle \boldsymbol{w}, \boldsymbol{x}_n\rangle + w_0\right) \geq 1, \qquad 1 \leq n \leq N$$

This is solved (numerically) by QP techniques:

- Quadratic (therefore convex) function subject to linear constraints
- Unique solution (or set of equivalent ones); therefore, NO LOCAL MINIMA

Support Vector Machines

Margin violations

- In practice, we allow small margin violations ε_n, for each $\boldsymbol{x}_n$:

$$\textbf{minimize} \qquad \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{n=1}^{N}\varepsilon_n$$

$$\textbf{subject to} \quad t_n\left(\langle \boldsymbol{w}, \boldsymbol{x}_n\rangle + w_0\right) \geq 1 - \varepsilon_n, \qquad \varepsilon_n \geq 0, 1 \leq n \leq N$$

- The ε_n are **slack** variables, leading to a **soft margin** ($\varepsilon_n > 0$ implying the functional margin $t_n y(\boldsymbol{x}_n) < 1$)

Support Vector Machines

Lagrangian form (primal)

- We first construct the Lagrangian:

$$\mathcal{L} = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{n=1}^{N} \alpha_n \Big\{ t_n \left(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + w_0 \right) - 1 + \varepsilon_n \Big\} + C \sum_{n=1}^{N} \varepsilon_n - \sum_{n=1}^{N} \mu_n \varepsilon_n$$

- The $\alpha_n, \mu_n \geq 0$ are the *Lagrange multipliers* (auxiliary variables to cope with the constraints)

- The solution is the saddle point of $\mathcal{L}$:

  1. the minimum of $\mathcal{L}$ is taken with respect to $\boldsymbol{w}, b$

  2. the maximum of $\mathcal{L}$ is taken with respect to the α_n

Support Vector Machines

Lagrangian form

The gradient of $\mathcal{L}$ with respect to $\boldsymbol{w}, b$ must vanish:

$$\frac{\delta \mathcal{L}}{\delta b} = \sum_{n=1}^{N} \alpha_n t_n = 0, \quad \frac{\delta \mathcal{L}}{\delta \boldsymbol{w}} = \boldsymbol{w} - \sum_{n=1}^{N} \alpha_n t_n \boldsymbol{x}_n = 0, \quad \frac{\delta \mathcal{L}}{\delta \varepsilon_n} = C - \alpha_n - \mu_n = 0$$

In addition, the so-called **KKT complementarity conditions** hold:

$$\alpha_n \Big(t_n \left(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + w_0 \right) - 1 + \varepsilon_n \Big) = 0, \ 1 \leq n \leq N$$

Support Vector Machines

Dual formulation

The Lagrangian $\mathcal{L}$ is convex; its optimization is equivalent to the maximization of its **dual problem** $\mathcal{L}_D$:

---

maximize $$\mathcal{L}_D = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m t_n t_m \langle \boldsymbol{x}_n, \boldsymbol{x}_m \rangle$$

subject to $0 \leq \alpha_n \leq C, \quad 1 \leq n \leq N,$ **and** $\sum_{n=1}^{N} \alpha_n t_n = 0$

- Note how neither $\mu_n, \varepsilon_n, \boldsymbol{w}, w_0$ appear in the dual form
- Maximization is only with respect to the α_n

Support Vector Machines

Dual formulation

A closer look at the KKT complementarity conditions:

- $\alpha_n = 0$ implies $t_n\, y(\boldsymbol{x}_n) > 1$ and $\varepsilon_n = 0$ ($\boldsymbol{x}_n$ is **not a SV**)
- $\alpha_n \in (0, C)$ implies $t_n\, y(\boldsymbol{x}_n) = 1$ and $\varepsilon_n = 0$ ($\boldsymbol{x}_n$ is a **non-bound SV**)
- $\alpha_n = C$ implies $t_n\ y(\boldsymbol{x}_n) < 1$ and $\varepsilon_n > 0$ ($\boldsymbol{x}_n$ is a **bound SV**)

Support Vector Machines

Result

- The discriminant function is:

$$y_{\mathsf{SVM}}(\boldsymbol{x}) = \mathsf{sgn}\left(\sum_{n=1}^{N} \alpha_n t_n \, \langle \boldsymbol{x}_n, \boldsymbol{x} \rangle + w_0\right)$$

  with $\alpha_n > 0$ *only* for the support vectors (for the rest, $\alpha_n = 0$).

- By setting $C = \infty$ we are in the *hard* margin case

  $\Rightarrow C$ represents a tradeoff between **margin width** and **training error**

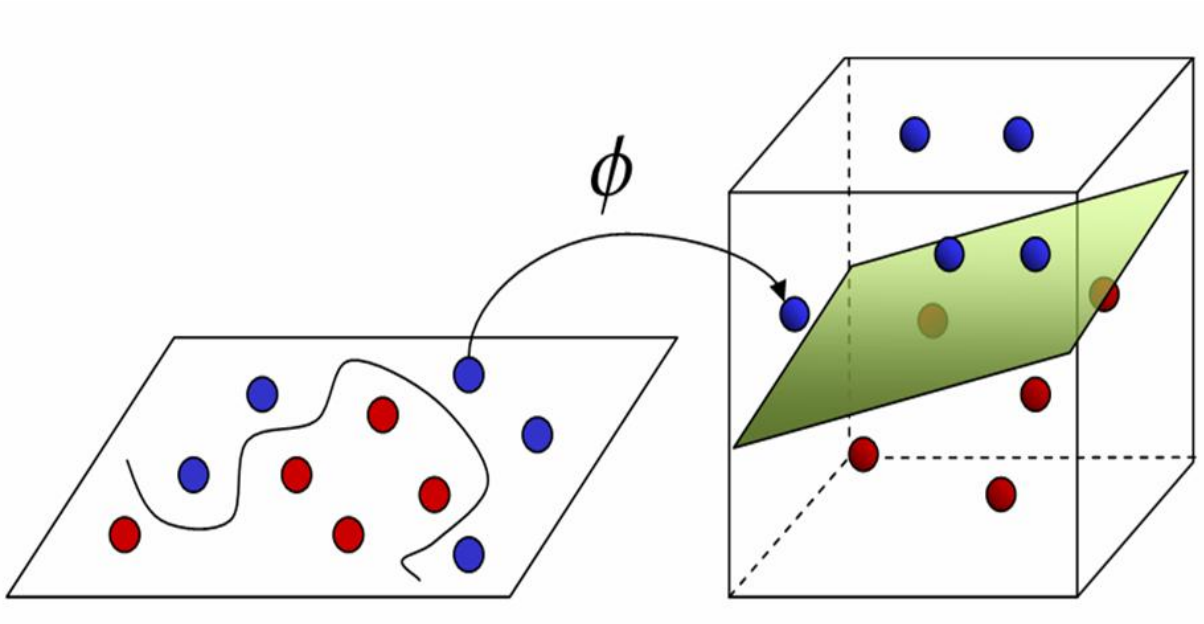
- $w_0 = -\left(\sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n t_n \, \alpha_m t_m \, \langle \boldsymbol{x}_n, \boldsymbol{x}_m \rangle\right) / \left(\sum_{n=1}^{N} \alpha_n\right)$

Support Vector Machines

General feature maps

Recall the idea of mapping input data into some Hilbert space (called the *feature space*) via a non-linear mapping $\phi : \mathcal{X} \to \mathcal{H}$

The associated kernel function is $k(\boldsymbol{u}, \boldsymbol{v}) = \langle \phi(\boldsymbol{u}), \phi(\boldsymbol{v}) \rangle, \ \boldsymbol{u}, \boldsymbol{v} \in \mathcal{X}$



Support Vector Machines

SVM training ... back to the OSH

- We now substitute $\boldsymbol{x}_n$ by $\phi(\boldsymbol{x}_n)$, then build the OSH in $\mathcal{H}$

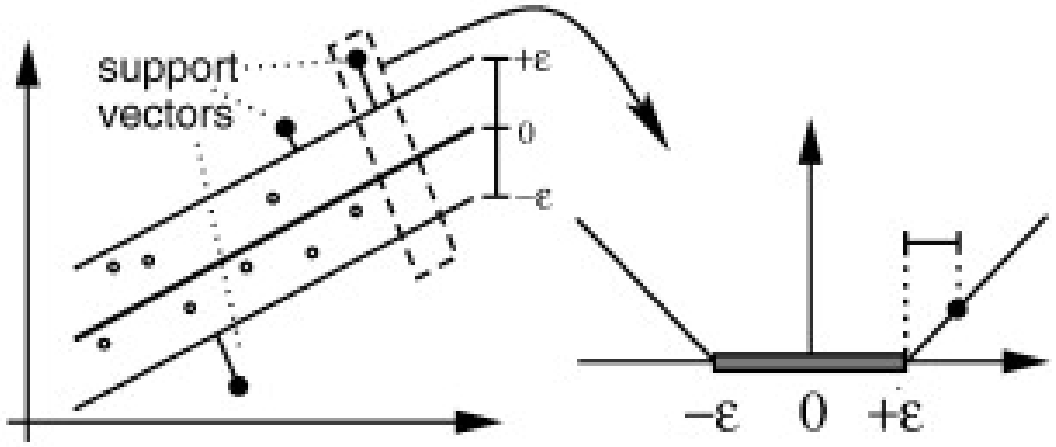
- The discriminant function becomes:

$$y_{\text{SVM}}(\boldsymbol{x}) = \text{sgn}\left(\sum_{n=1}^{N} \alpha_n t_n k(\boldsymbol{x}, \boldsymbol{x}_n) + w_0\right)$$

- The dual of the new QP problem is formulated exactly as before, replacing $\langle \boldsymbol{x}_n, \boldsymbol{x}_m \rangle$ with $k(\boldsymbol{x}_n, \boldsymbol{x}_m)$

Support Vector Machines

SVMs for regression



We choose the loss: $|t - y(\boldsymbol{x})|_\epsilon = \text{máx}(|t - y(\boldsymbol{x})| - \epsilon, 0)$

Support Vector Machines

SVMs for regression

---

minimize $$\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{n=1}^{N}(\epsilon_n + \epsilon_n^*)$$

subject to

$$\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + w_0 - t_n \leq \varepsilon + \epsilon_n \quad (1)$$

$$t_n - \langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + w_0 \leq \varepsilon + \epsilon_n^* \quad (2)$$

$$\epsilon_n, \epsilon_n^* \geq 0 \quad (3)$$

Support Vector Machines

SVMs for novelty detection

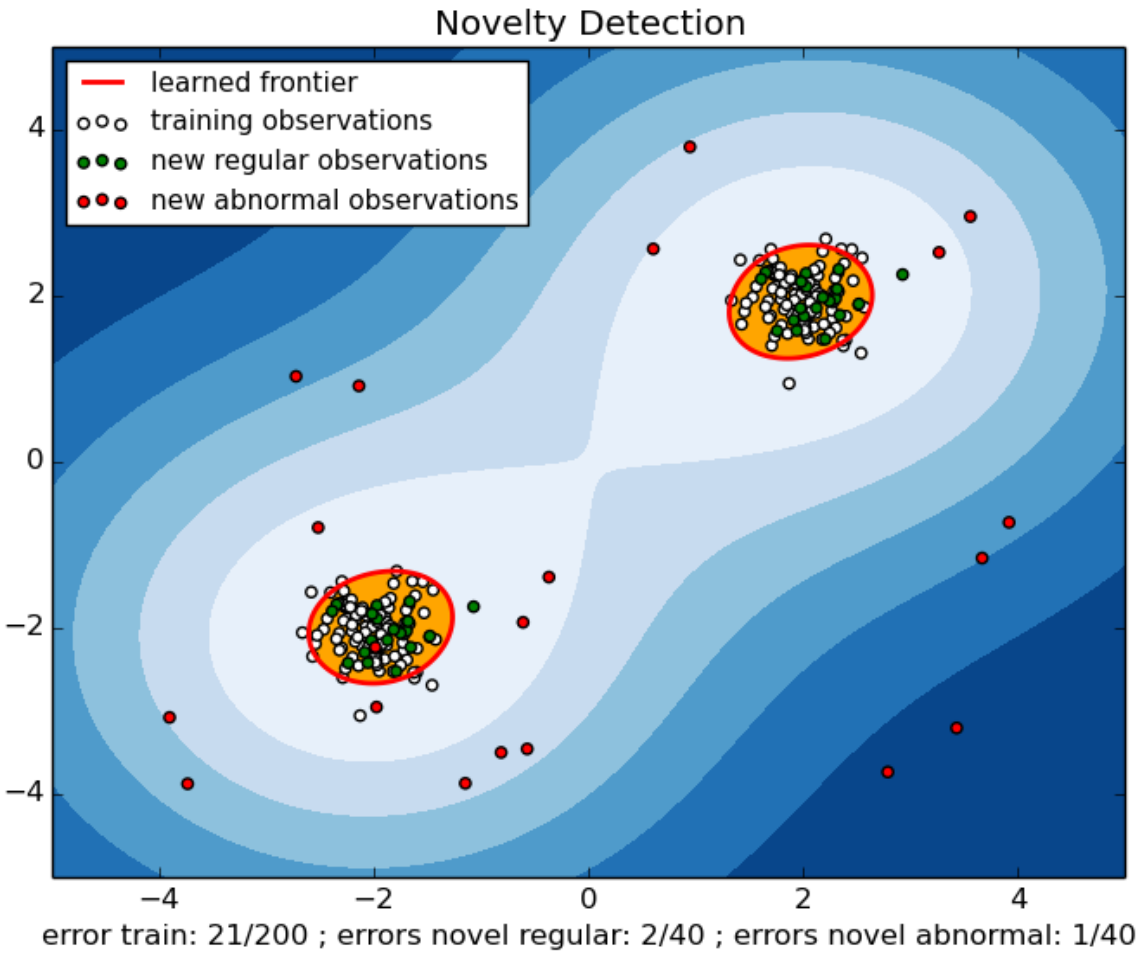
- You are given a dataset drawn from a pdf P; you want to estimate a "simple" subset S of input space s.t. the probability that a test point drawn from P lies outside of S equals some a priori specified $\nu \in (0, 1)$

- The SVM approach to this problem is by estimating a function f which is positive on S and negative on its complement

- The ν parameter characterizes both the fraction of SVs and the fraction of outliers

Support Vector Machines

SVMs for novelty detection



Support Vector Machines

SVMs for novelty detection

USPS dataset of handwritten digits: 9,298 digit images of size 16×16

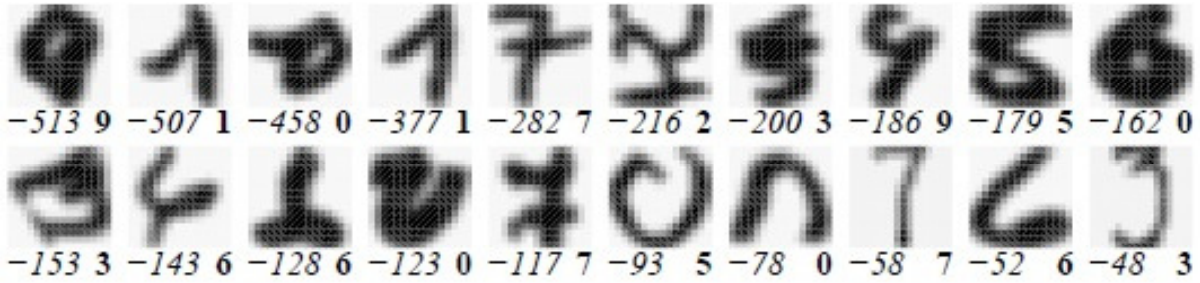


Figure 2: Outliers identified by the proposed algorithm, ranked by the negative output of the SVM (the argument of the sgn in the decision function). The outputs (for convenience in units of 10^{-5}) are written underneath each image in italics, the (alleged) class labels are given in bold face. Note that most of the examples are "difficult" in that they are either atypical or even mislabelled.

The 20 worst outliers for the USPS test set (here $\nu = 0{,}05$)

(from Schoelkopf et al, *Support Vector Method for Novelty Detection*, NIPS 2000)

Support Vector Machines

In conclusion ...

- (⇑) No local minima, no initial conditions, few parameters to set
- (⇑) Provide with a compact description of original data
- (⇑) Decision surfaces can be RBFs, polynomials, multilayer perceptrons, ...
- (⇑) Accept any data type as input
- (⇑) Explicit complexity control via C, ϵ and the kernel parameters
- (⇑) Less affected by large input dimensions than other methods
- (⇑) Excellent practical performance: OCR, text categorization, face detection, ...

- (⇓) Training requires solving a QP problem
- (⇓) Choice of best kernel is an open issue; **kernel design** is an active area of research
- (⇓) Sometimes fraction of SVs very high (indicating a poor model)
- (⇓) Performance usually depends on a careful choice of the parameters

Machine Learning

Syllabus

1. Introduction to Machine Learning

2. Theoretical issues (I): regression

3. Linear regression and beyond

4. Theoretical issues (II): classification

5. Generative classifiers

6. Discriminative classifiers

7. Clustering
8. Learning with kernels (I): The SVM
9. Learning with kernels (II): Kernel functions
10. Learning with kernels (III): Other kernel methods
11. Artificial neural networks (I): the MLP
12. Artificial neural networks (II): the RBF
13. Ensemble methods: Random Forests
14. Advanced topics and frontiers