УНИВЕРСИТЕТ ИТМО

ITMO ML LAB

Lecture 3.2
Support vector machine

Machine Learning Camp

Sergey Muravyov

January 29, 2022

Lecture plan

Linearly separable case

Linearly inseparable case

Kernel trick

Kernel selection and synthesis

Other problems

Main idea

If we say that the classifier should be linear, what is the best way to define it?

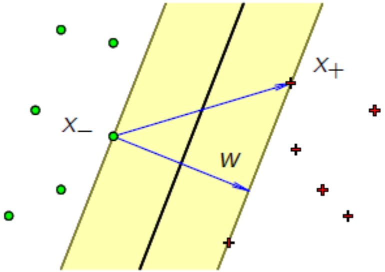
Main idea: search for a surface that is the most distant from the classes (large margin classification).

Linearly separable case

Key hypothesis: sample is linearly separable:

$$\exists w, w_0 : M_i(w, w_0) = y_i(\langle w, x_i \rangle - w_0) > 0, i = 1, \dots, |\mathcal{D}|.$$

Separating lines exist, therefore a line that has maximum distance from both the classes also exists.



Separating stripe

Normalize margin:

$$\min_i M_i(w, w_0) = 1.$$

Separating stripe equation:

$$\{x : -1 \leq \langle w, x \rangle - w_0 \leq 1\}.$$

Stripe width:

$$\frac{\langle x_+ - x_-, w \rangle}{\|w\|} = \frac{(\langle x_+, w \rangle - w_0) - (\langle x_-, w \rangle - w_0)}{\|w\|} = \frac{2}{\|w\|}.$$

It turns to be a minimization problem:

$$\begin{cases} \|w\|^2 \to \min_{w, w_0}; \\ M_i(w, w_0) \geq 1, \qquad i = 1, \ldots, |\mathcal{D}|. \end{cases}$$

Lecture plan

Linearly separable case

Linearly inseparable case

Kernel trick

Kernel selection and synthesis

Other problems

Linearly inseparable case

Key hypothesis: sample is not linearly separable:

$$\forall w, w_0 \exists x_d : M_d(w, w_0) = y_d(\langle w, x_d \rangle - w_0) < 0$$

There is no such separating line.

We can still try to find a line with smallest margins for each object.

In case of linearly inseparable sample:

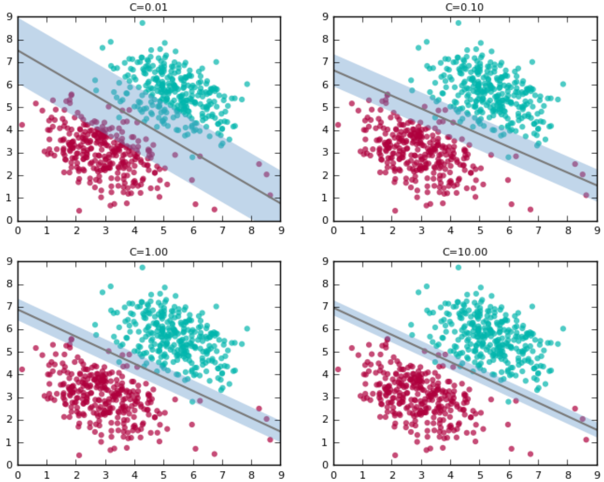
$$\begin{cases} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{|\mathcal{D}|}\xi_i \to \min_{w,w_0,\xi}; \\ M_i(w,w_0) \geq 1-\xi_i, i=1,\dots,|\mathcal{D}|; \\ \xi_i \geq 0, \qquad i=1,\dots,|\mathcal{D}|. \end{cases}$$

Equivalent unconditional optimization problem:

$$\sum_{i=1}^{|\mathcal{D}|}(1-M_i(w,w_0))_+ + \frac{1}{2C}\|w\|^2 \to \min_{w,w_0},$$

where $(x)_+ = max(0,x) = (x+|x|)/2$.
This is the approximated empirical risk.

C effect



Non-linear programming problem

Mathematical programming problem:

$$\begin{cases} f(x) \to \min_x \\ g_i(x) \le 0, \\ h_j(x) = 0. \end{cases} \qquad i = 1, \dots, m; j = 1, \dots, k.$$

Lagrangian:

$$\mathcal{L}(x; \mu, \lambda) = f(x) + \sum_{i=1}^{m} \mu_i g_i(x) + \sum_{j=1}^{k} \lambda_j h_j(x)$$

Karush—Kuhn—Tucker conditions of minimum:

$$\frac{\delta \mathcal{L}}{\delta x}(x^*; \mu, \lambda) = 0. \qquad \begin{cases} g_i(x^*) \le 0; \\ h_j(x^*) = 0; \\ \mu_i \ge 0; \\ \mu_i g_i(x^*) = 0. \end{cases} \qquad i = 1, \dots, m; j = 1, \dots, k.$$

SVM problem

Lagrangian

$$\mathcal{L}(w, w_0, \xi; \alpha, \beta) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{|\mathcal{D}|} \alpha_i (M_i(w, w_0) - 1) - \sum_{j=1}^{|\mathcal{D}|} \xi_j (\alpha_j + \beta_j - C)$$

α_i are variables, dual for constraints $M_i \geq 1 - \xi_i$;
β_i are variables, dual for constraints $\xi_i \geq 0$.

Condition of minimum:

$$\begin{cases} \frac{\delta \mathcal{L}}{\delta w} = 0; \frac{\delta \mathcal{L}}{\delta w_0} = 0; \frac{\delta \mathcal{L}}{\delta \xi} = 0; \\ \xi_i \geq 0; \alpha_i \geq 0; \beta_i \geq 0; \\ \alpha_i = 0 \text{ или } M_i(w, w_0) = 1 - \xi_i; \\ \beta_i = 0 \text{ или } \xi_i = 0; \end{cases}$$

$$i = 1, \ldots, |\mathcal{D}|.$$

Support vectors

Object types:

1. $\alpha_i = 0; \beta_i = C; \xi_i = 0; M_i > 1$
   peripheral objects.
2. $0 < \alpha_i < C; 0 < \beta_i < C; \xi_i = 0; M_i = 1$
   support boundary objects.
3. $\alpha_i = C; \beta_i = 0; \xi_i > 0; M_i < 1$
   support-disturbers.

Object x_i is **support object** if $\alpha_i \neq 0$.

Non-linear programming problem

$$-\mathcal{L}(\alpha) = -\sum_{i=1}^{|\mathcal{D}|} \alpha_i + \frac{1}{2}\sum_{i=1}^{|\mathcal{D}|}\sum_{j=1}^{|\mathcal{D}|} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \to \min_{\alpha}$$

$$\begin{cases} 0 \leq \alpha_i \leq C, i = 1, \dots, l; \\ \sum_{i=1}^{l} \alpha_i y_i = 0. \end{cases}$$

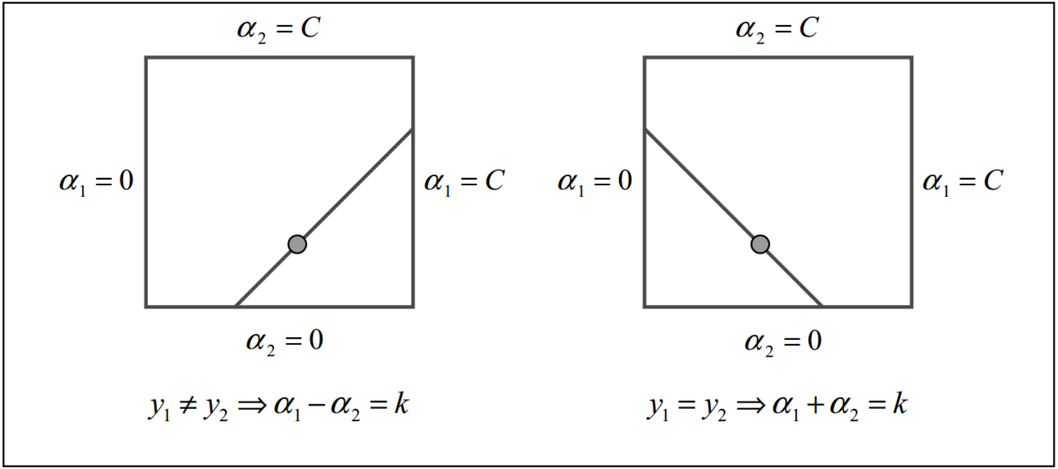
Primal problem solution can be expressed with dual problem solution:

$$\begin{cases} w = \sum_{i=1}^{|\mathcal{D}|} \alpha_i y_i x_i; \\ w_0 = \langle w, x_i \rangle - y_i. \end{cases} \quad \forall i : \alpha_i > 0, M_i = 1.$$

Linear classifier:

$$a(x) = \operatorname{sign}\left(\sum_{i=1}^{|\mathcal{D}|} \alpha_i y_i \langle x_i, x \rangle - w_0 \right).$$

Sequential Minimal Optimization (SMO)



Lecture plan

Linearly separable case

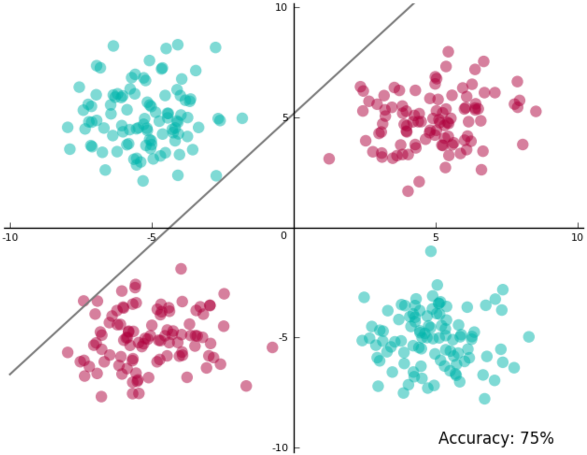
Linearly inseparable case

Kernel trick

Kernel selection and synthesis

Other problems

Bad linearly inseparable case



Kernel trick

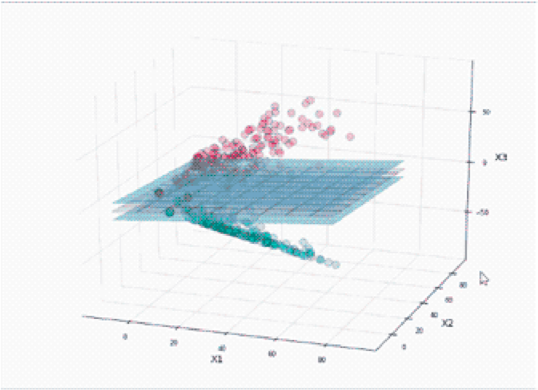
Main idea:

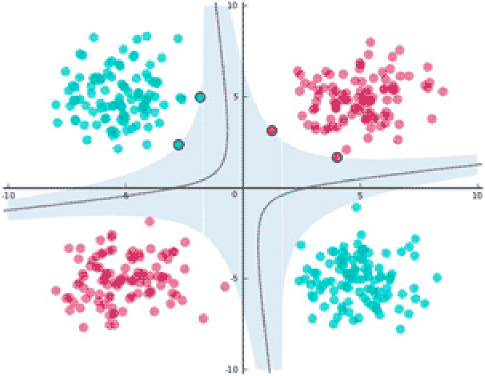
- Find an **implicit** mapping to a higher-dimensional space, such that the points in new space will be linearly separable.
- Calculating the kernel is faster than dot product with explicit conversion.

Explicit Conversion Example:

- Let separating surface can be well approximated by a sum of functions depending on $x_1, \dots, x_n : c_1x_1 + \dots + c_nx_n + f_1(x_1, \dots, x_n) + \dots + f_k(x_1, \dots, x_n)$.
- If we add features $f_1(x_1, \dots, x_n), \dots, f_k(x_1, \dots, x_n)$, then we will have new space over variables $x_1, \dots, x_n, x_{n+1}, \dots, x_{n+k}$, points of which will be linearly separable.

Resulting separating surface





Why kernels?

We can build distance-based classifier for support objects (vectors). Using a kernel function is equal to using a certain mapping.

The main problem is to find a kernel, which maps initial space into linearly separable.

Lecture plan

Linearly separable case

Linearly inseparable case

Kernel trick

Kernel selection and synthesis

Other problems

Kernel

Function $K : X \times X \to \mathbb{R}$ is **kernel function** if it can be represented as $K(x, x') = \langle \psi(x), \psi(x') \rangle$ with a mapping $\psi : X \to H$, where H is a space with a scalar product.

Theorem (Mercer)

Function $K(x, x')$ is a kernel if it's:

- Symmetrical: $K(x, x') = K(x', x)$
- Non-negatively defined on $\mathbb{R}$ for any function $g : X \to \mathbb{R}$:

$$\int_X \int_X K(x, x') g(x) g(x') dx dx' \geq 0$$

Kernel examples

Quadratic:

$$K(x, x') = \langle x, x' \rangle^2$$

Polynomial with monomial degree equal to d:

$$K(x, x') = \langle x, x' \rangle^d$$

Polynomial with monomial degree $\leq d$:

$$K(x, x') = (\langle x, x' \rangle + 1)^d$$

Neural nets:

$$K(x, x') = \sigma(\langle x, x' \rangle)$$

Radial-basis:

$$K(x, x') = \exp(-\beta \|x - x'\|^2)$$

Kernel synthesis

- $K(x, x') = \langle x, x' \rangle$ is kernel;
- Constant $K(x, x') = 1$ is kernel;
- $K(x, x') = K_1(x, x')K_2(x, x')$ is kernel;
- $\forall \psi : X \to \mathbb{R} K(x, x') = \psi(x)\psi(x')$ is kernel;
- $K(x, x') = \alpha_1 K_1(x, x') + \alpha_2 K_2(x, x')$ with $\alpha_1, \alpha_2 > 0$ is kernel;
- $\forall \phi : X \to X$ if K_0 is kernel, then $K(x, x') = K_0(\phi(x), \phi(x'))$ is kernel;
- if $s : X \times X \to \mathbb{R}$ is symmetric and integrable, then $K(x, x') = \int_X s(x, z)s(x', z)dz$ is kernel.

SVM discussion

Advantages:

- Convex quadratic programming problem has a single solution
- Any separating surface
- Small number of support object used for learning

Disadvantages:

- Sensitive to noise
- No common rules for kernel function choice
- The constant C should be chosen
- No feature selection

Lecture plan

Linearly separable case

Linearly inseparable case

Kernel trick

Kernel selection and synthesis

Other problems

Other penalties

Relevance vector machine:

$$\frac{1}{2}\sum_{i=1}^{|\mathcal{D}|}\left(\ln\lambda_i + \frac{\alpha_i^2}{\lambda_i}\right)$$

LASSO SVM:

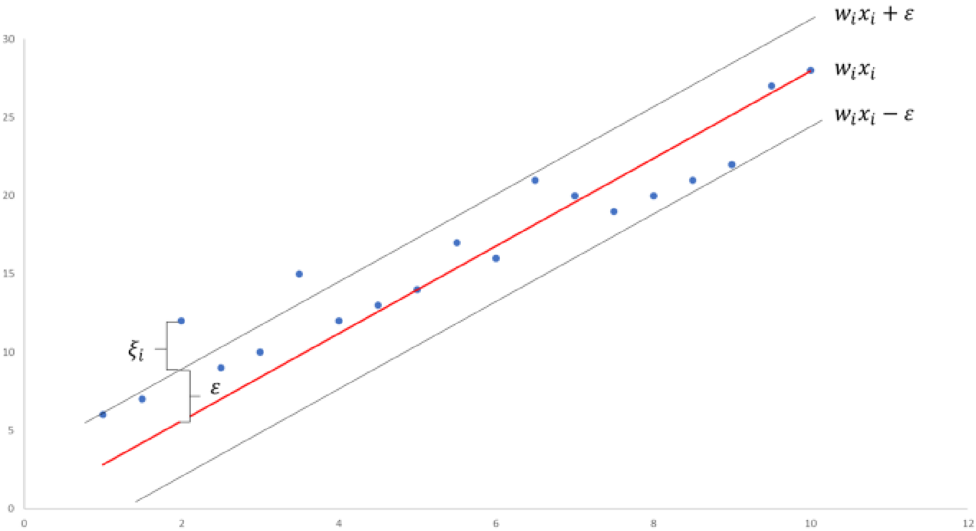
$$\mu\sum_{i=1}^{|\mathcal{D}|}|w_i|$$

Support feature machine:

$$\sum_{i=1}^{|\mathcal{D}|}R_\mu(w_i),$$

where μ — selectivity parameter, $R_\mu = \begin{cases} 2\mu|w_i|, & \text{if } |w_i| < \mu, \\ \mu^2 + w_i^2, & \text{otherwise.} \end{cases}$

Support Vector Regression (SVR)



Learning Using Privileged Information (LUPI)

Original problem

$$\begin{cases} \min \frac{1}{2} \langle w, w \rangle + C \sum \xi_i \\ M_i(w, w_0) \geq 1 - \xi_i \end{cases}$$

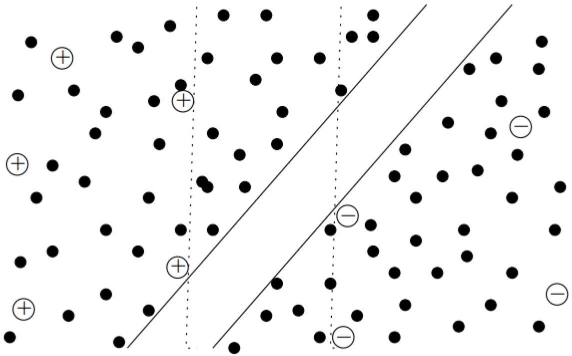
Modification for privileged data x^*

$$\begin{cases} \min \frac{1}{2} \left(\langle w, w \rangle + \gamma \langle w^*, w^* \rangle \right) + C \sum \left(\langle w^*, x_i^* \rangle - w_0^* \right) \\ y_i \left(\langle w, x_i \rangle - w_0 \right) \geq 1 - \left(\langle w^*, x_i^* \rangle - w_0^* \right) \\ \left(\langle w^*, x_i^* \rangle - w_0^* \right) \geq 0 \end{cases}$$

Semi-Supervised Support Vector Machine (S3VM)

The loss function contains the distance to unallocated objects:

$$\sum_{i=1}^{l}(1 - M_i(w, w_0))_+ + \frac{1}{2C}\|w\|^2 + \sum_{j=1}^{u}(1 - |M_j(w, w_0)|)_+ \to \min_{w, w_0}$$



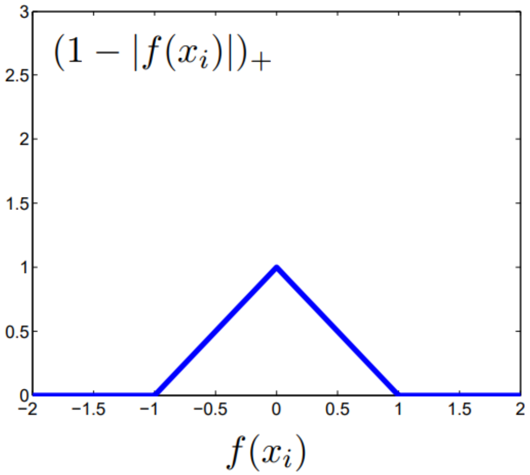
S3VM loss function (Hat loss)

Problem

- Using **hat loss** leads to non-convex optimization.

Ways to solve:

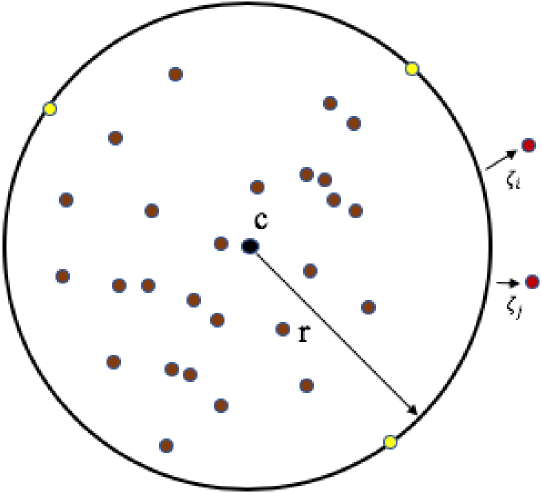
- Use fine-tuned non-convex optimization.
- Make hat loss smooth and apply gradient descent.
- Use upper bound and split into several convex optimization problems.



Search for anomalies and novelty (One-Class SVM)

$$\begin{cases} \min r^2 + C \sum \xi_i \\ \|\Phi(x_i) - c\|^2 \leq r^2 + \xi_i \\ \xi_i \geq 0 \end{cases}$$

Φ is higher dimensional space projection.



Thanks for your attention!

ITsMOre than a UNIVERSITY