Lecture 4

Support vector machine

Intellectual systems
(Machine Learning)

Andrey Filchenkov

27.09.2018

Lecture plan

- Linearly separable case
- Linearly inseparable case
- Kernel trick
- Kernel selection and synthesis
- Regularization for SVM

- The presentation is prepared with materials of the K.V. Vorontsov’s course “Machine Leaning”.
- Slides are available online: **goo.gl/BspjhF**

Lecture plan

- Linearly separable case
- Linearly inseparable case
- Kernel trick
- Kernel selection and synthesis
- Regularization for SVM

Basic idea

Basic idea: if we say that classifier is linear, what is the best way to define it?

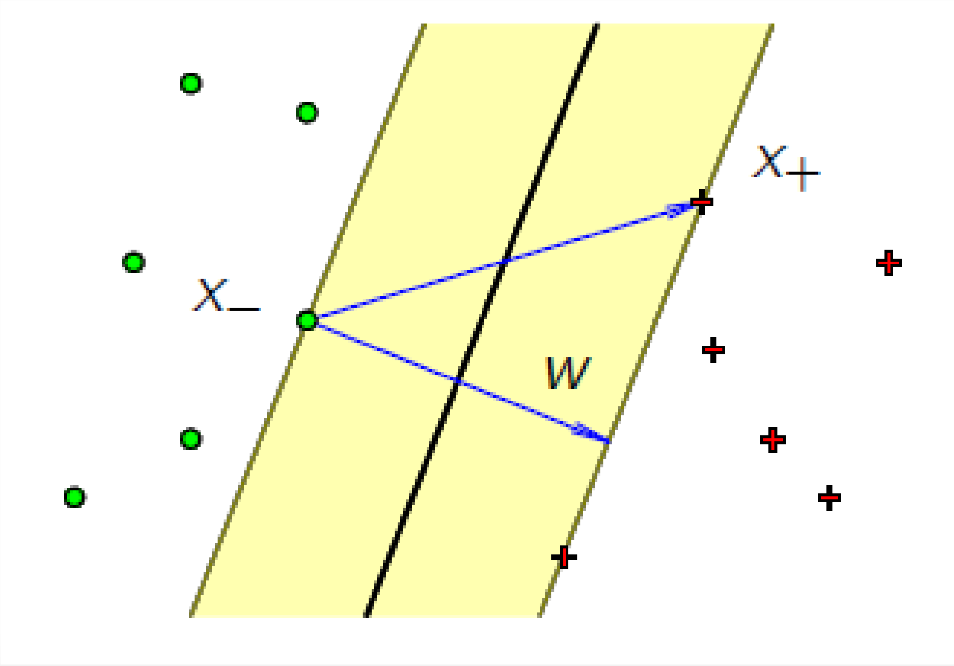
Main idea: search for a surface that is the most distant from the classes (large margin classification).

Linearly separable case

Key hypothesis: sample is linearly separable:

$$\exists w, w_0 \colon M_i(w, w_0) = y_i(\langle w, x_i \rangle - w_0) > 0, i = 1, \dots, \ell.$$

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:

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

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:

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

Lecture plan

- Linearly separable case
- Linearly inseparable case
- Kernel trick
- Kernel selection and synthesis
- Regularization for SVM

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.

Linearly inseparability

In case of linearly inseparable sample:

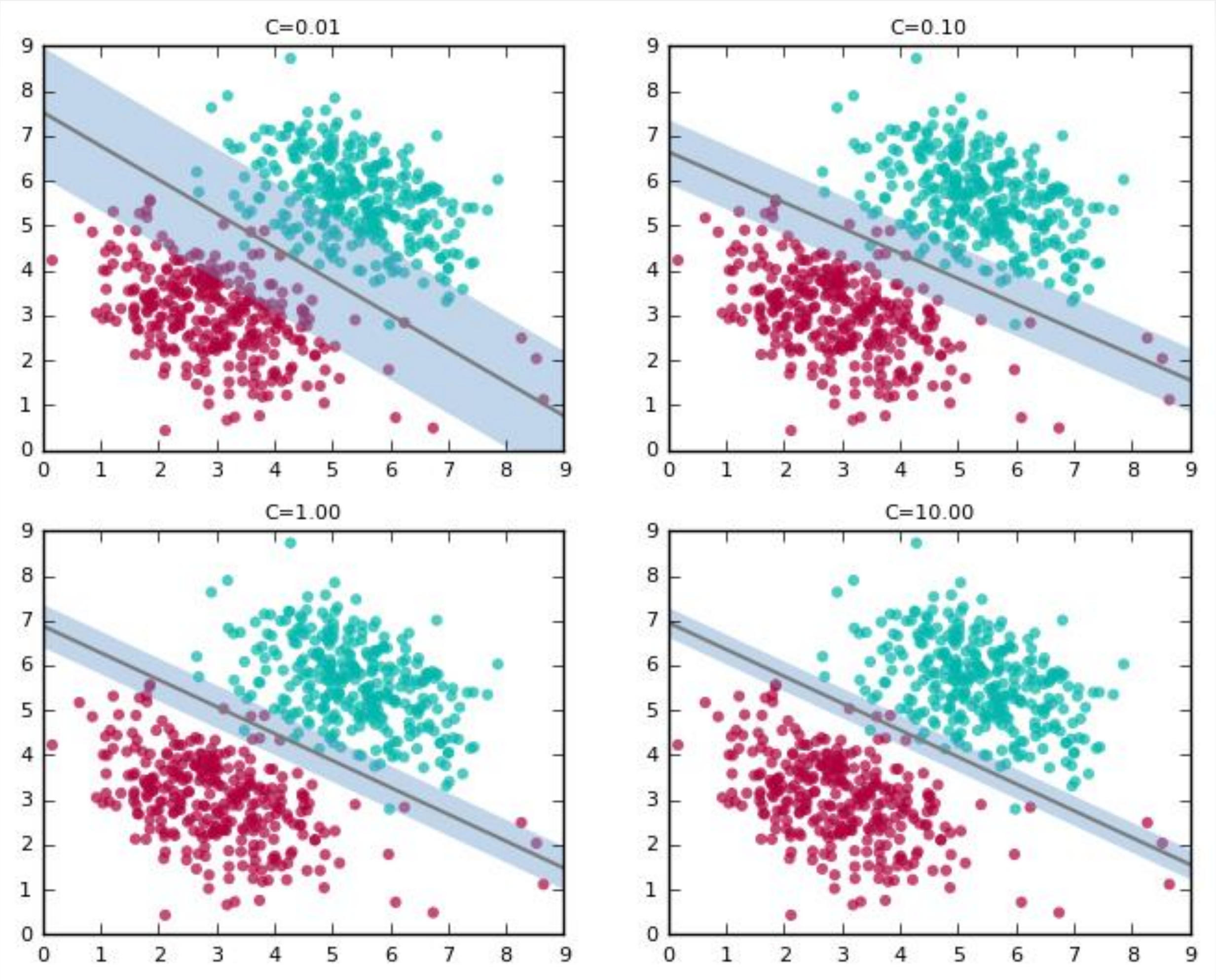
$$\begin{cases} \frac{1}{2}||w||^2 + C\sum_{i=1}^{\ell}\xi_i \to \min_{w,w_0,\xi}; \\ M_i(w,w_0) \geq 1 - \xi_i, i = 1,\dots,\ell; \\ \xi_i \geq 0, \qquad i = 1,\dots,\ell. \end{cases}$$

Equivalent unconditional optimization problem:

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

This is the approximated empirical risk.

C effect



Non-linear programming problem

Mathematical programming problem:

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

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:

$$\frac{\delta \mathcal{L}}{\delta x}(x^*; \mu, \lambda) = 0.$$

$$\begin{cases} g_i(x^*) \leq 0; \\ h_j(x^*) = 0; \\ \mu_i \geq 0; \\ \mu_i g_i(x^*) = 0. \end{cases} \qquad i = 1, \dots, m; j = 1, \dots, k.$$

SVM problem

Lagrangian

$$\mathcal{L}(w, w_0; \mu, \lambda) = \frac{1}{2}||w||^2 - \sum_{i=1}^{m} \mu_i (M_i(w, w_0) - 1) - \sum_{j=1}^{k} \xi_j (\mu_i + \lambda_i - C)$$

λ_i are variables, dual for constraints $M_i \geq 1 - \xi_i$;

μ_i are variables, dual for constraints $\xi_i \geq 0$.

Condition of minimum:

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

$i = 1, \dots, m.$

Support vectors

Object types:

1. $\lambda_i = 0; \mu_i = C; \xi_i = 0; M_i > 1$

peripheral objects.

2. $0 < \lambda_i < C; 0 < \mu_i < C; \xi_i = 0; M_i = 1$

support boundary objects.

3. $\lambda_i = C; \mu_i = 0; \xi_i > 0; M_i < 1$

support-disturbers.

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

Non-linear programming problem

$$-\mathcal{L}(\lambda) = -\sum_{i=1}^{\ell}\lambda_i + \frac{1}{2}\sum_{i=1}^{\ell}\sum_{j=1}^{\ell}\lambda_i\lambda_j y_i y_j\langle x_i, x_j\rangle \to \min_{\lambda}$$

$$\begin{cases} 0 \le \lambda_i \le C; \\ \sum_{j=1}^{\ell}\lambda_i y_i = 0. \end{cases}$$

Primal problem solution can be expressed with dual problem solution:

$$\begin{cases} w = \sum_{i=1}^{\ell}\lambda_i y_i x_i\,; \\ w_0 = \langle w, x_i\rangle - y_i. \end{cases} \quad \forall i\colon \lambda_i > 0, M_i = 1.$$

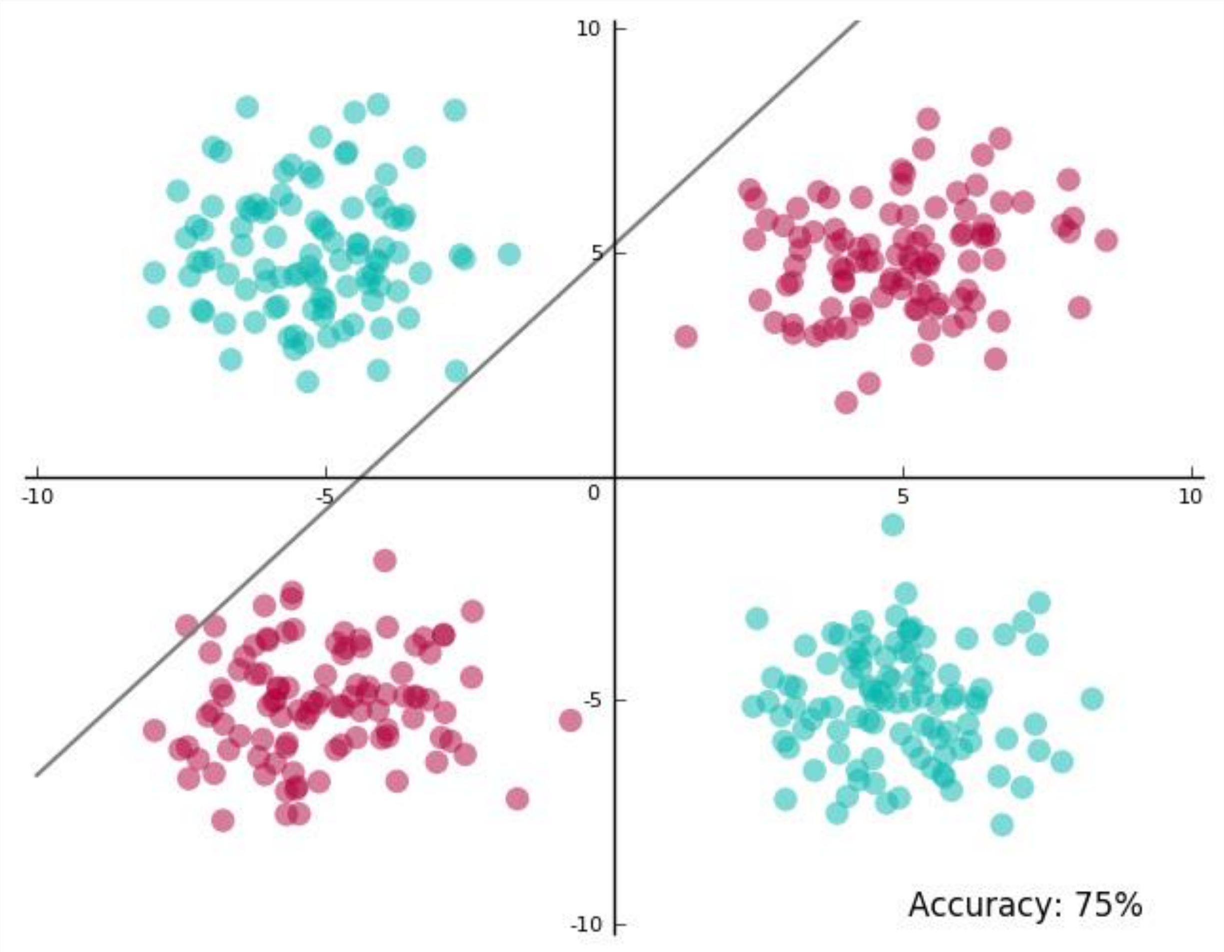
Linear classifier:

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

Lecture plan

- Linearly separable case
- Linearly inseparable case
- Kernel trick
- Kernel selection and synthesis
- Regularization for SVM

Bad linearly inseparable case



Kernel trick

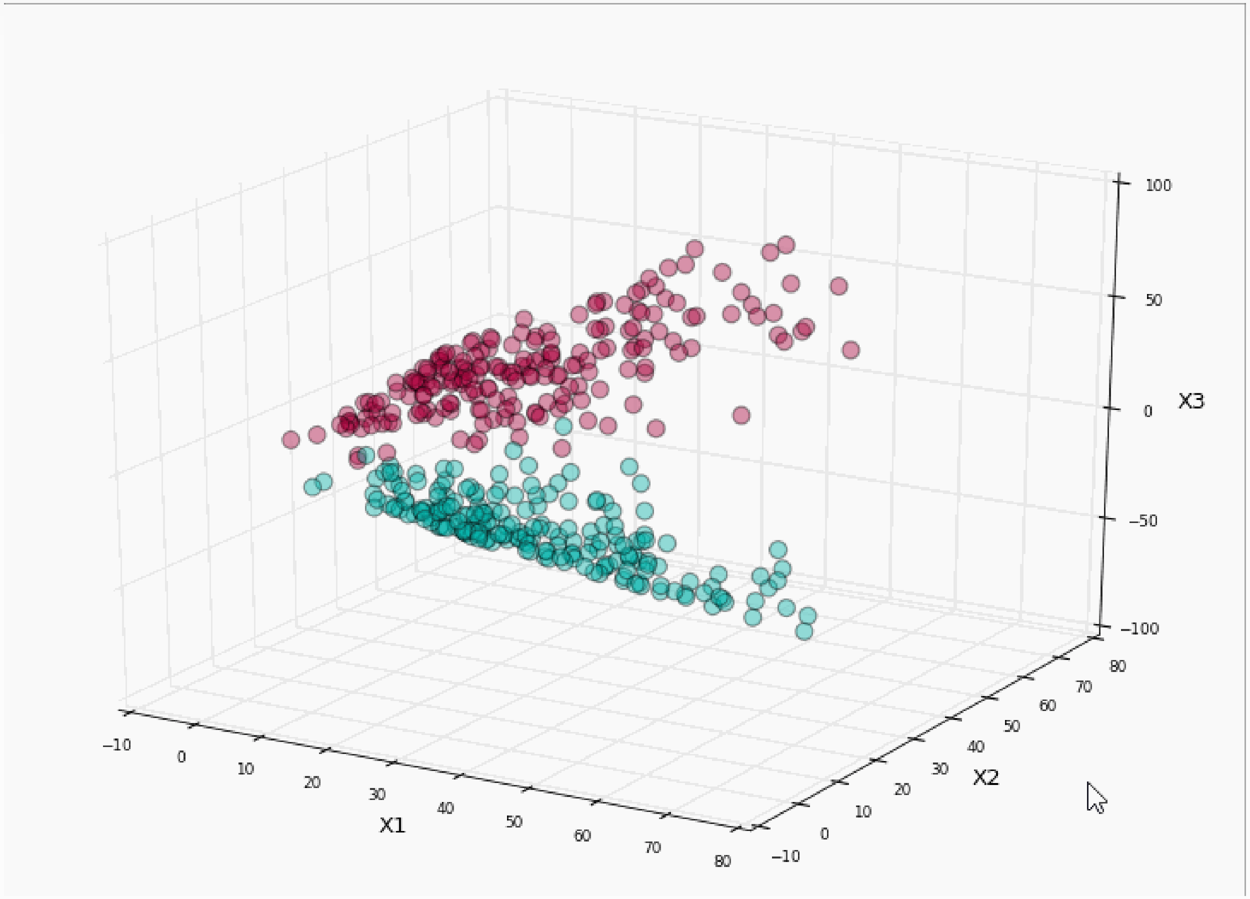
Main idea: find a mapping to a higher-dimensional space, such that the points in new space will be linearly separable.

Idea basis: let separating surface can be well approximated by a sum of functions depending on $x_1, \dots, x_n$:

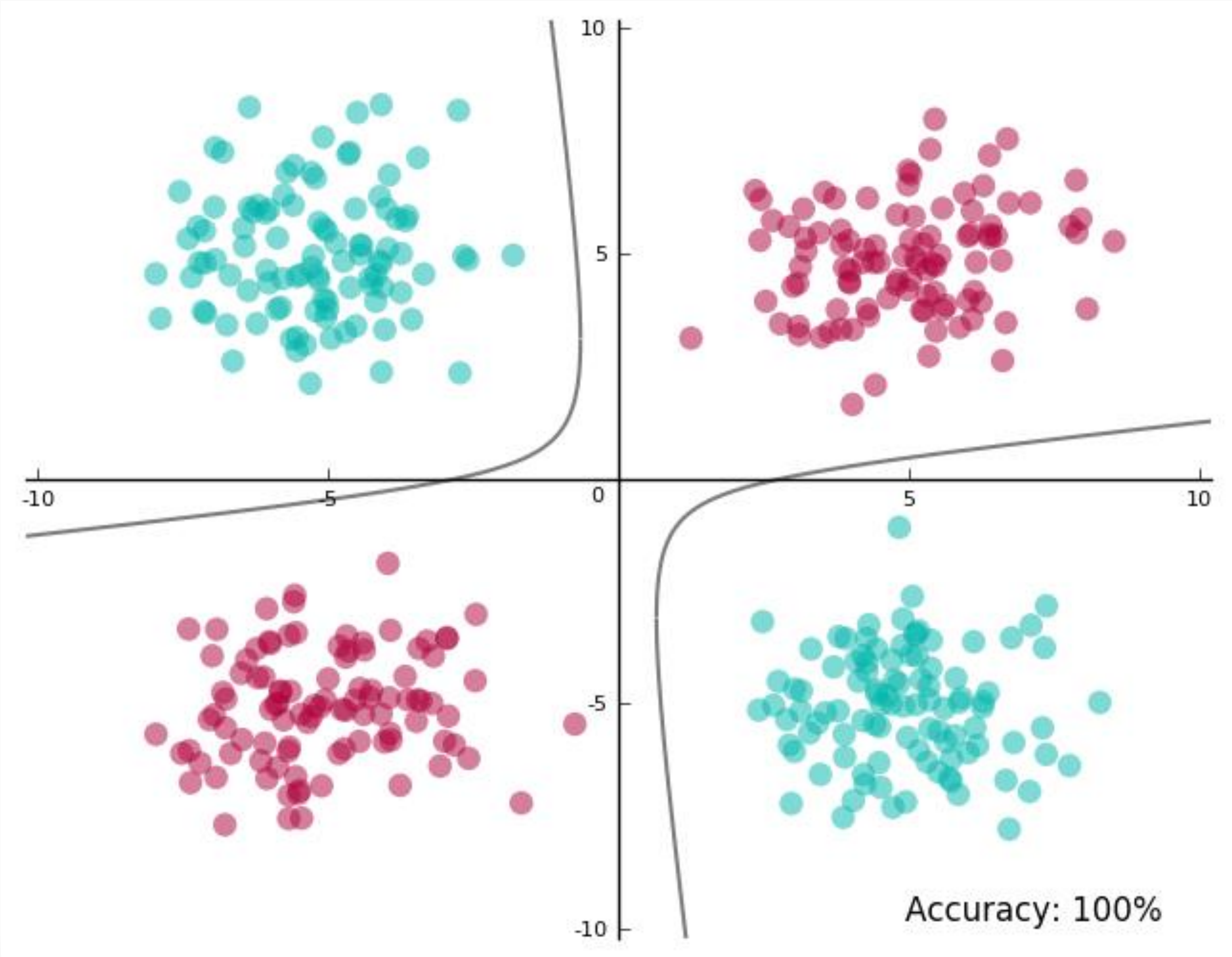
$$c_1 x_1 + \cdots + c_n x_n + f_1(x_1, \dots, x_n) + \cdots + f_k(x_1, \dots, x_n)$$

If we add features $f_1(x_1, \dots, x_n)$, …, $f_k(x_1, \dots, x_n)$, 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.

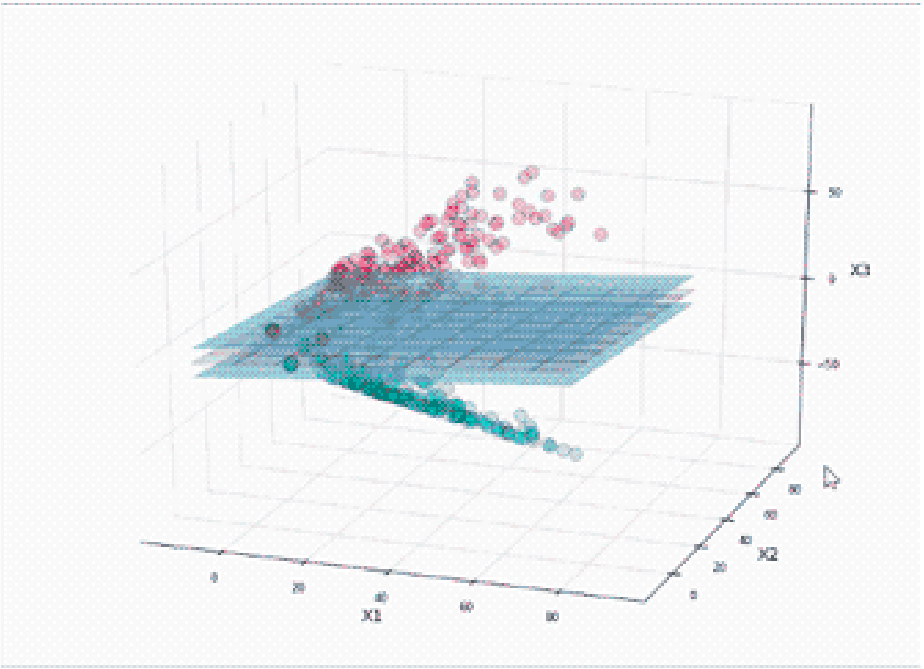
Separability in higher dimension

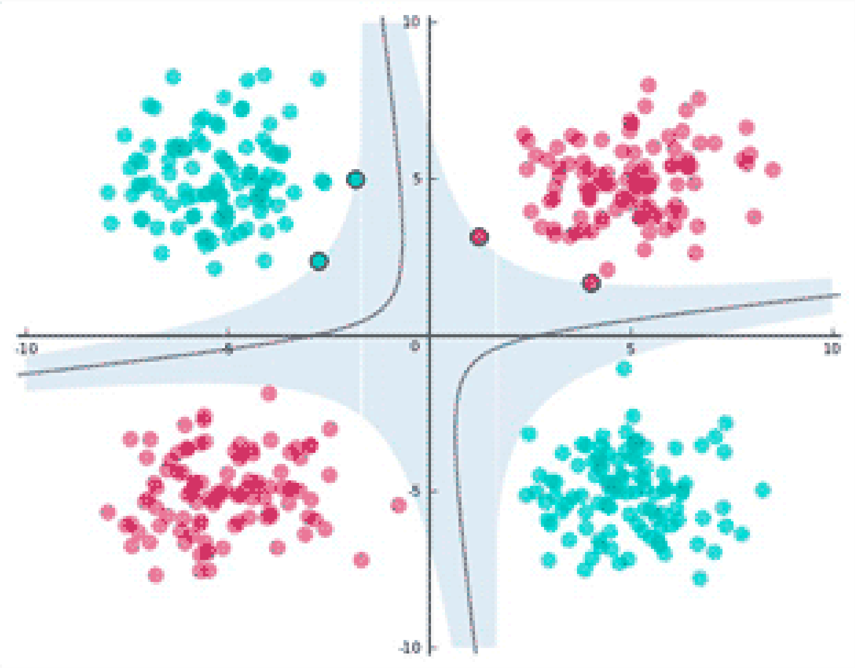


How it looks in original space



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
- Regularization for SVM

Kernels

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

Theorem (Mercer)

Function $K(x, x')$ is kernel iff it is symmetrical, $K(x, x') = K(x', x)$, and non-negatively defined on $\mathbb{R}$:

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

for any function $g: X \rightarrow \mathbb{R}$.

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

$$K(x, x') = \int_X s(x, z)(x', z)dz \text{ is kernel.}$$

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
- Regularization for SVM

Regularization (reminder)

Key hypothesis: *w* “swings” during overfitting

Main idea: clip *w* norm.

Add regularization penalty for weights norm:

$$Q_\tau\left(a_w, T^\ell\right) = Q\left(a_w, T^\ell\right) + \frac{\tau}{2}\|w\|^2 \to \min_w.$$

And SVM equation is:

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

Other penalties

Relevance vector machine:

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

LASSO SVM:

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

Support feature machine:

$$\sum_{i=1}^{n} R_\mu(w_i),$$

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