Lecture 4
Support vector machine

Machine Learning

Andrey Filchenkov

27.09.2019

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 Learning".
- Slides are available online: **bit.ly/2lynu3L**

Lecture plan

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

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:

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

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

Machine learning. Lecture 4. SVM. 27.09.2019.

5
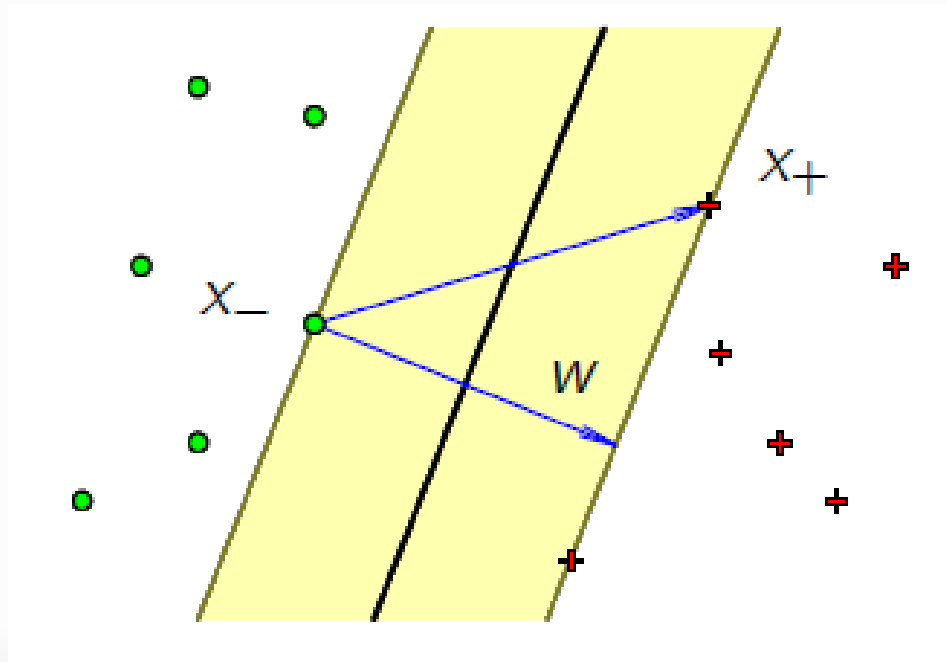
Separating stripe

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:

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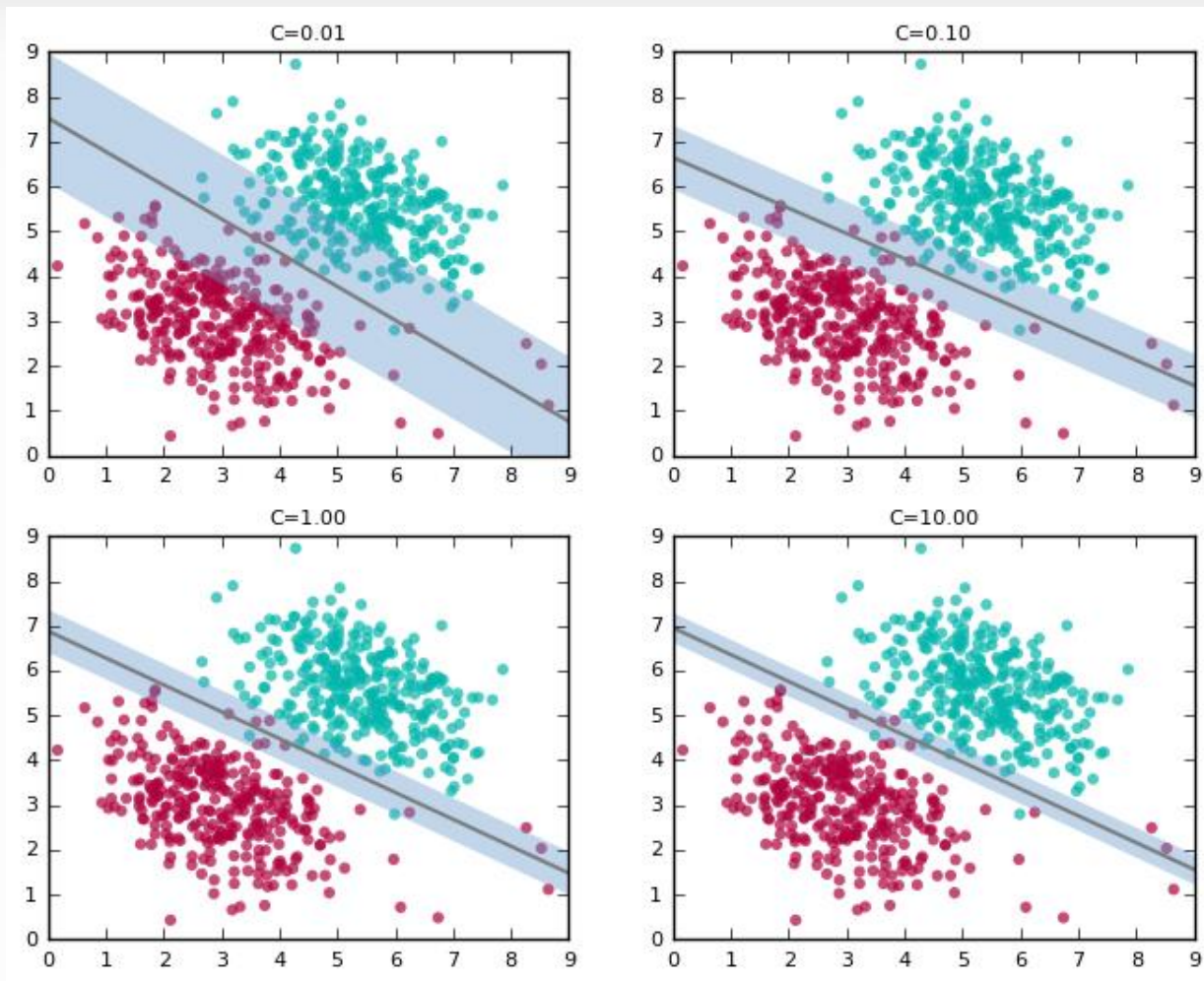
Equivalent unconditional optimization problem:

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

where $(x)_+ = (x + |x|)/2$.

This is the approximated empirical risk.

Machine learning. Lecture 4. SVM. 27.09.2019.

9

C effect

Machine learning. Lecture 4. SVM. 27.09.2019.

10

Non-linear programming problem

Mathematical programming problem:

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

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:

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

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

SVM problem

Lagrangian

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

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:

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

$i = 1, \dots, m.$

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 $\alpha_i \neq 0$.

Machine learning. Lecture 4. SVM. 27.09.2019.

13

Non-linear programming problem

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

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

Primal problem solution can be expressed with dual problem solution:

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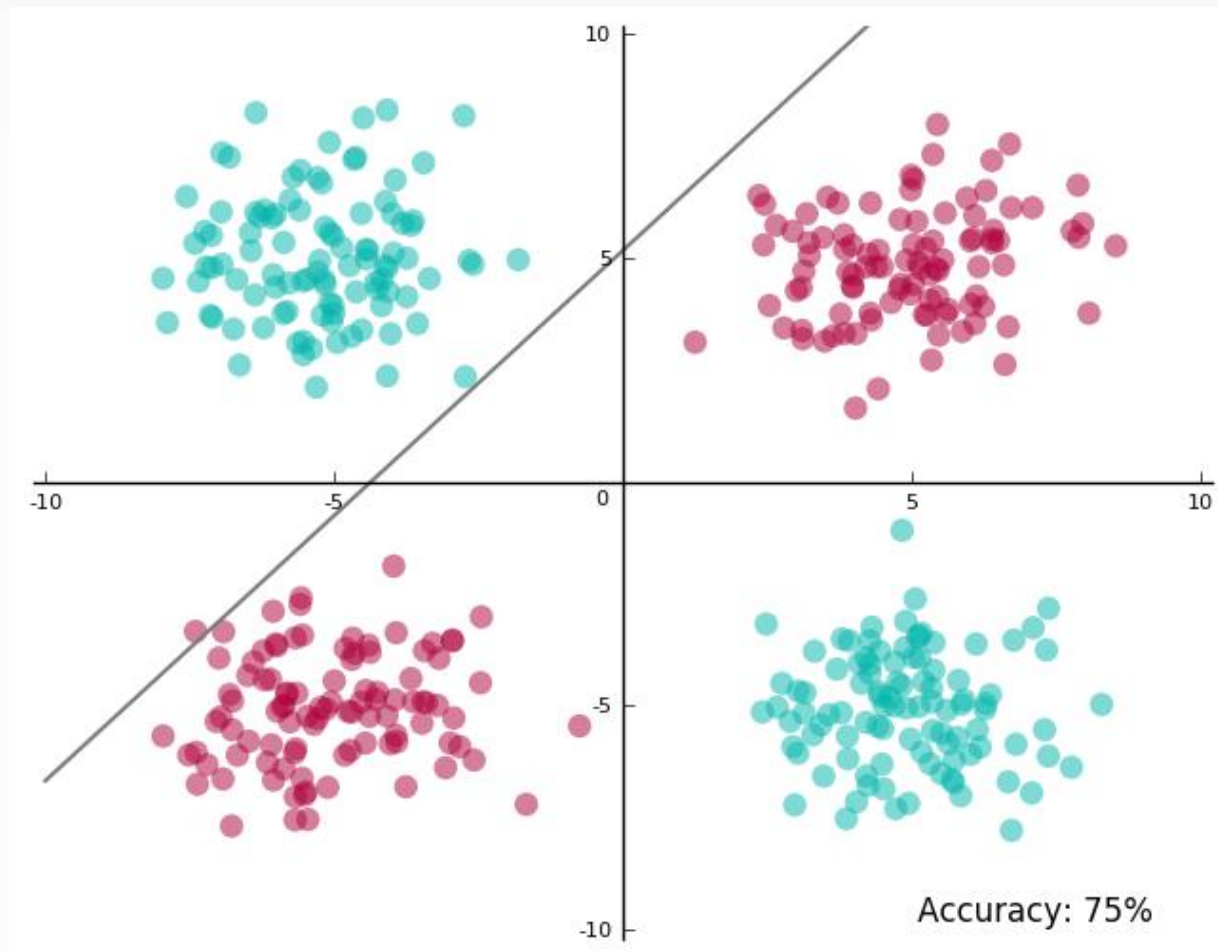
Linear classifier:

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

Lecture plan

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

Bad linearly inseparable case

Accuracy: 75%

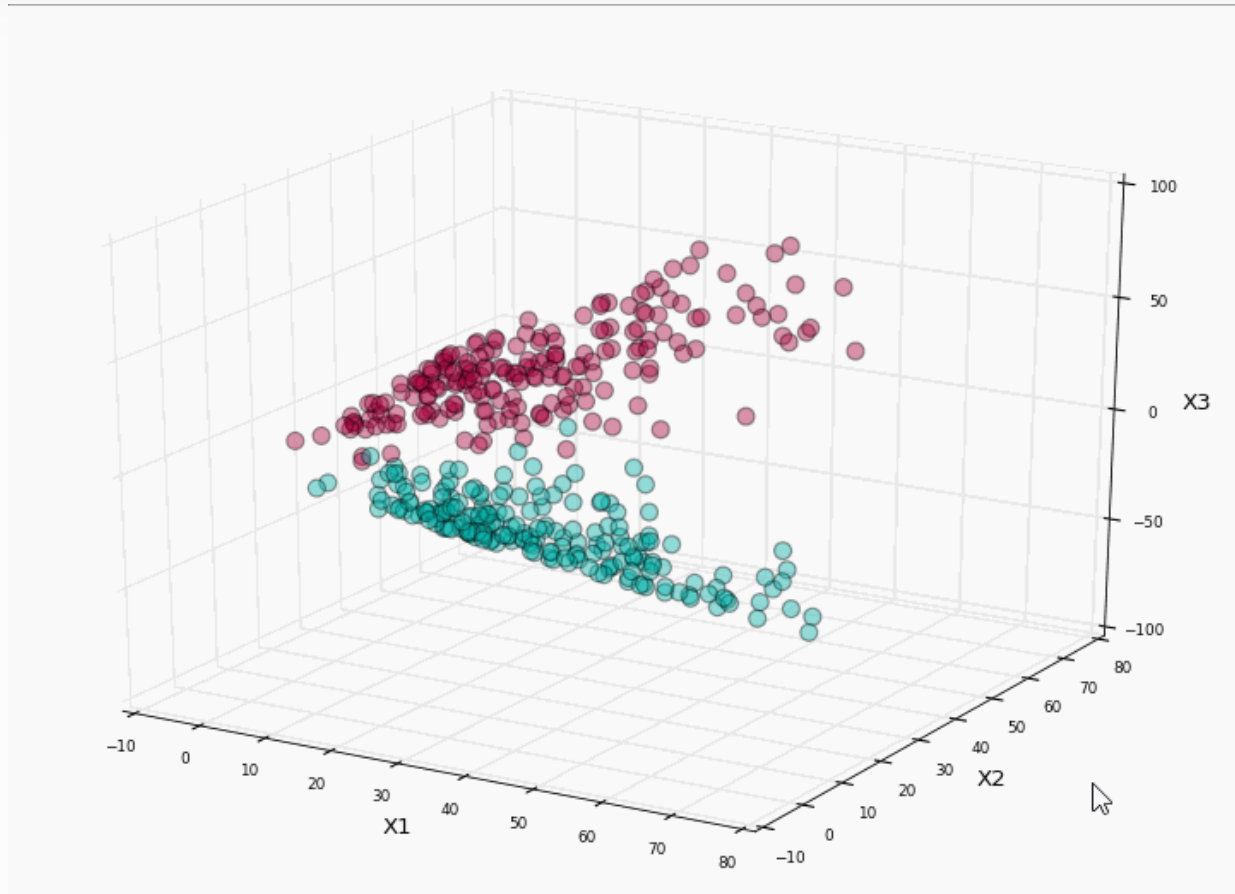Machine learning. Lecture 4. SVM. 27.09.2019.

16

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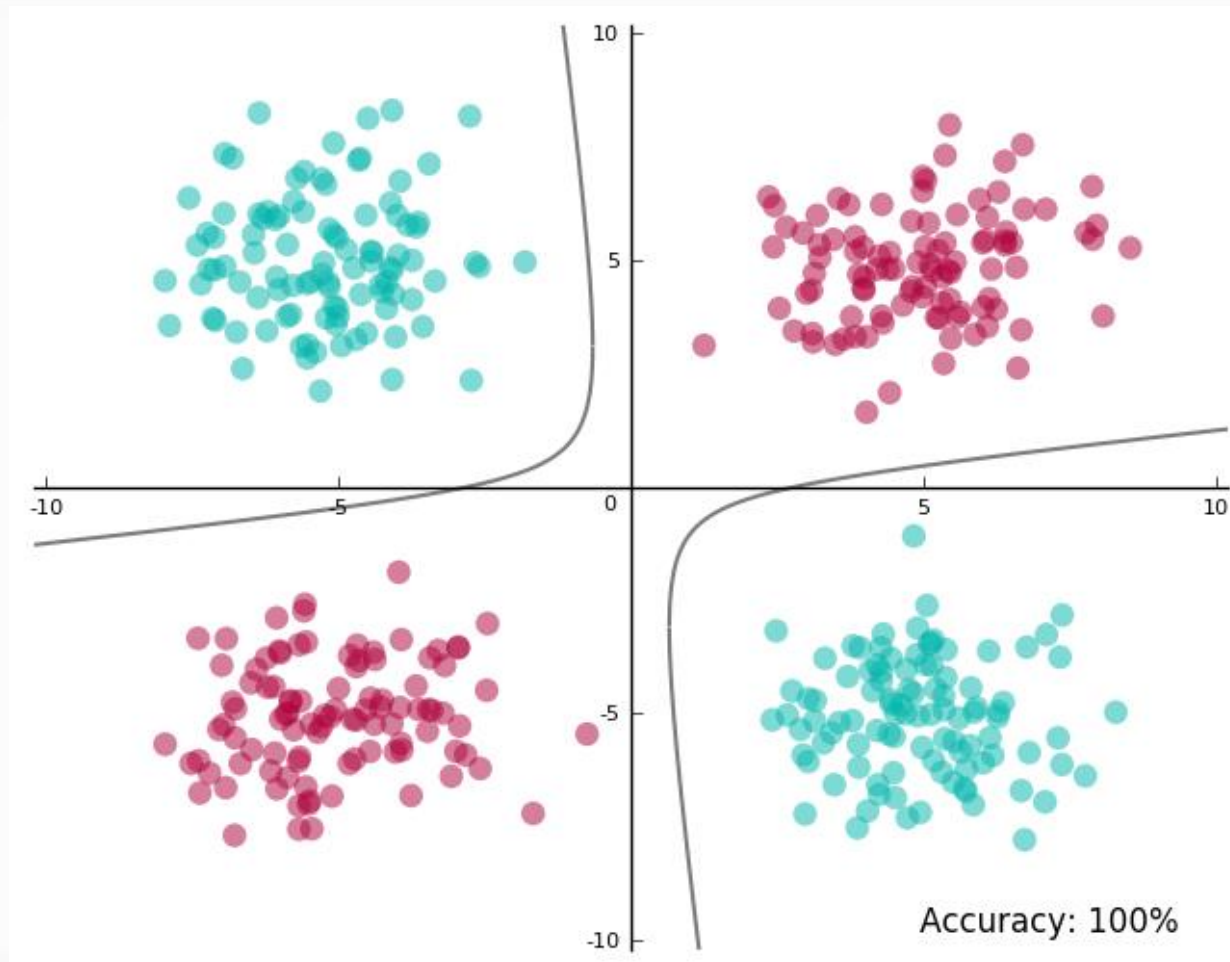
$$c_1 x_1 + \dots + c_n x_n + f_1(x_1, \dots, x_n) + \dots + f_k(x_1, \dots, x_n)$$

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.

Machine learning. Lecture 4. SVM. 27.09.2019.

17

Separability in higher dimension

Machine learning. Lecture 4. SVM. 27.09.2019.

18

How it looks in original space

Machine learning. Lecture 4. SVM. 27.09.2019.

19

Resulting separating surface

Machine learning. Lecture 4. SVM. 27.09.2019.

20
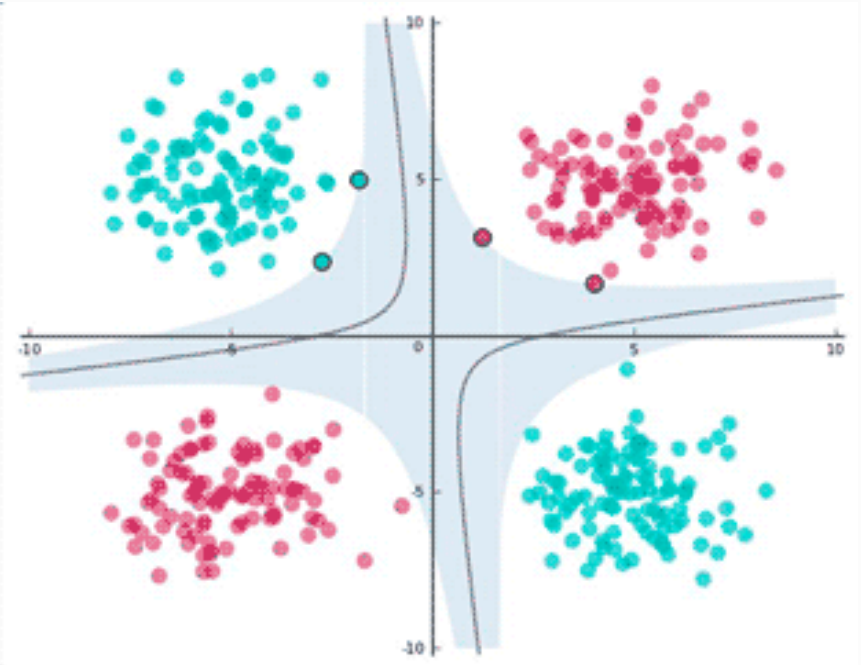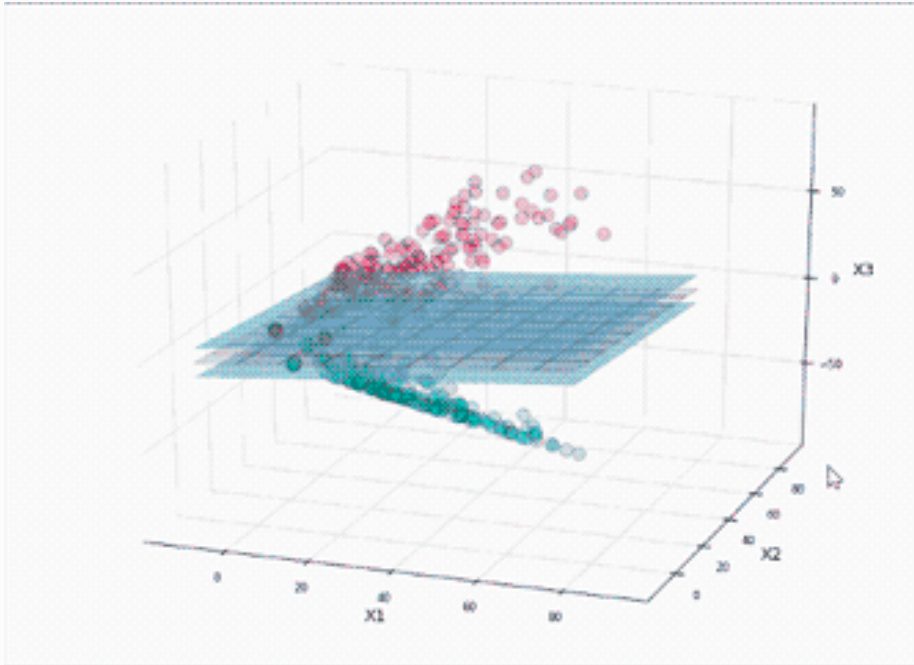
Why kernels?

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.

Machine learning. Lecture 4. SVM. 27.09.2019.

21

Lecture plan

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

Kernels

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 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 $g: X \to \mathbb{R}$.

Machine learning. Lecture 4. SVM. 27.09.2019.

23

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

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

Machine learning. Lecture 4. SVM. 27.09.2019.

24

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

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

Machine learning. Lecture 4. SVM. 27.09.2019.

25

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

Machine learning. Lecture 4. SVM. 27.09.2019.

26

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:

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

And SVM equation is:

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

Machine learning. Lecture 4. SVM. 27.09.2019.

28

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

Machine learning. Lecture 4. SVM. 27.09.2019.

29