

Lecture 4

Support vector machine

Intellectual systems
(Machine Learning)

Andrey Filchenkov

03.10.2015

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".

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?

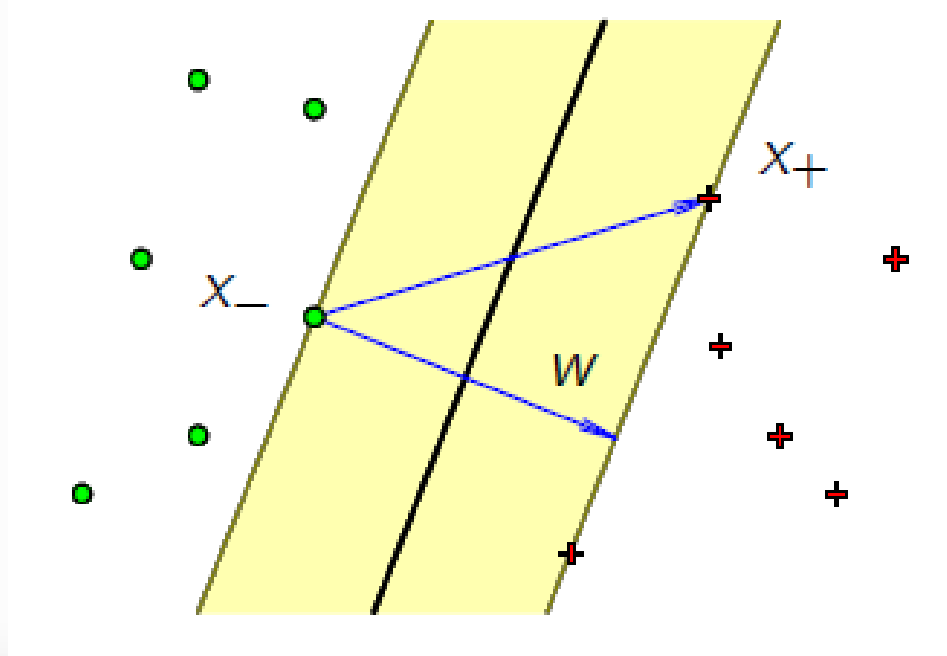
Main idea: search for a surface that is the most distant from the classes (large margin classification).

Linearly separable case

Key hypothesis: sample is linearly separable:

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

Separating lines exist, therefore such 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:

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

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 lines, but we can still try to find a line with the smallest margins for each object.

Linearly inseparability

In case of linearly inseparable sample:

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

Equivalent problem of unconditional optimization problem:

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

This is approximated empirical risk.

Non-linear programming problem

Mathematical programming problem:

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

Lagrangian:

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

Karush – Kuhn – Tucker conditions:

$$\begin{cases} \frac{\delta \mathcal{L}}{\delta x}(x^*; \mu, \kappa) = 0. \\ g_i(x^*) \leq 0; \\ h_j(x^*) = 0; \\ \mu_i \geq 0; \\ \mu_i g_i(x^*) = 0. \end{cases} \quad i = 1, \dots, m; j = 1, \dots, k.$$

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:

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

$i = 1, \dots, m$.

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

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

$$\begin{cases} 0 \leq \lambda_i \leq C; \\ \sum_{j=1}^{\ell} \lambda_j y_j = 0. \end{cases}$$

Primal problem solution can be expressed with dual problem solution:

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

Linear classifier:

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

Lecture plan

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

Kernel trick

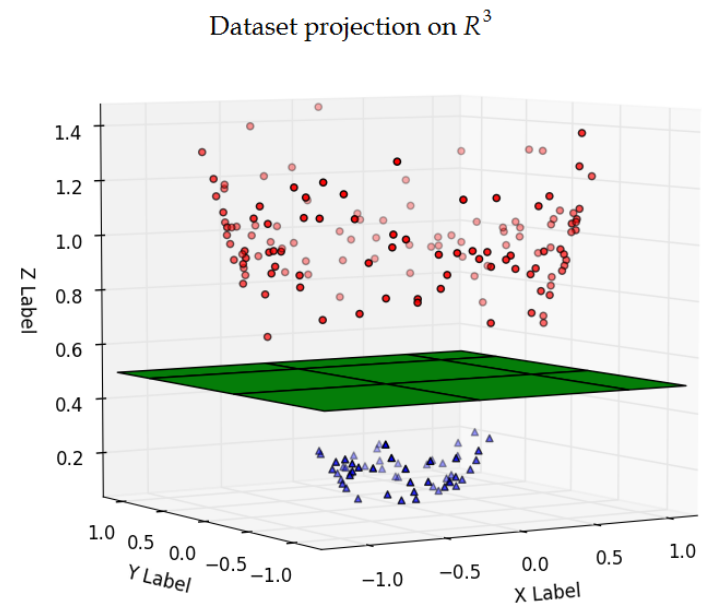
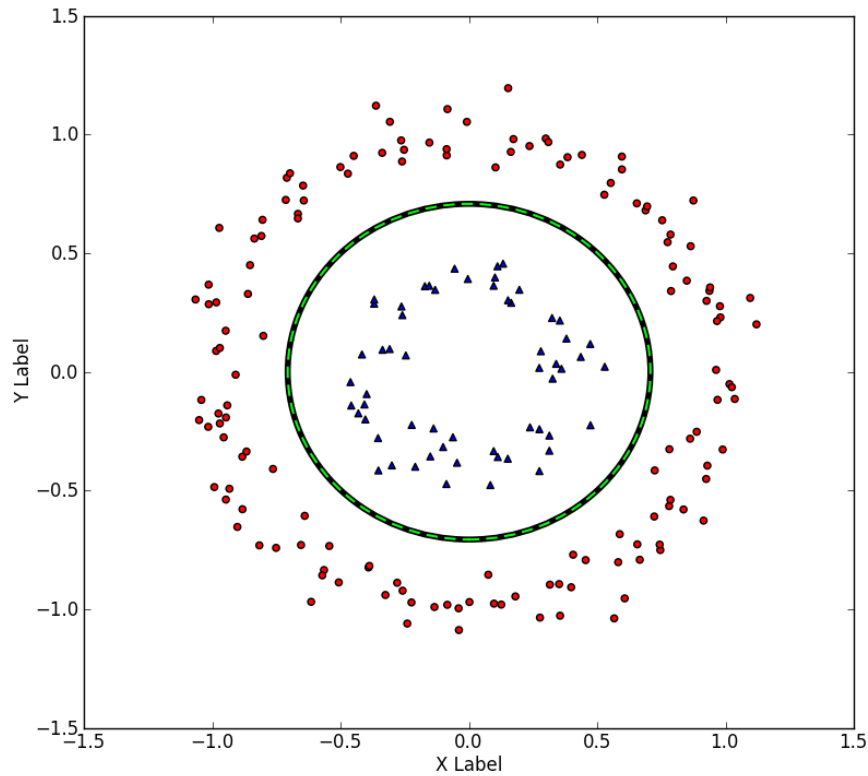
Main idea: find such a mapping to higher-dimensional space, that the points in new space will be linearly separable.

Idea basis: let separating surface can be well approximated by sum of functions depend 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 $f_1(x_1, \dots, x_n), \dots, f_k(x_1, \dots, x_n)$, than we would have new space $x_1, \dots, x_n, x_{n+1}, \dots, x_{n+k}$, in which it points would be linearly separable.

Example



Why kernels?

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

Main problem is to find kernel which maps our 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 – 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} :

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

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

Kernel examples

Quadratic:

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

Polynomial with monomial degree = 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;
- $\forall \psi: X \rightarrow \mathbb{R} \ K(x, x') = \psi(x)\psi(x')$ 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;
- $\forall \phi: X \rightarrow X$ if K_0 is kernel, than $K(x, x') = K_0(\phi(x), \phi(x'))$ is kernel;
- if $s: X \times X \rightarrow \mathbb{R}$ 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:

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

And SVM equation is:

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

Quadratic penalty conditions

Let $w \in \mathbb{R}^n$ is described with n -dimensional Gaussian distribution:

$$p(w; \sigma) = \frac{1}{(2\pi\sigma)^{n/2}} \exp\left(-\frac{\|w\|^2}{2\sigma}\right),$$

(weights are independent, their expectations are equal to zeros, their variances are the same and equal to σ).

It leads to quadratic penalty:

$$\frac{1}{2\sigma} \|w\|^2 + \text{const}(w).$$

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:

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

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