

Kernel trick

Victor Kitov

v.v.kitov@yandex.ru

Table of Contents

1 Recap of SVM solution

2 Kernel SVM parameters

Making predictions

- 1 Solve dual task to find α_i^* , $i = 1, 2, \dots, N$

$$\begin{cases} \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \rightarrow \max_{\alpha} \\ \sum_{i=1}^N \alpha_i y_i = 0 \\ 0 \leq \alpha_i \leq C \end{cases}$$

- 2 Find optimal w_0 :

$$w_0 = \frac{1}{n_{\tilde{S}\tilde{V}}} \left(\sum_{j \in \tilde{S}\tilde{V}} y_j - \sum_{j \in \tilde{S}\tilde{V}} \sum_{i \in S\mathcal{V}} \alpha_i^* y_i \langle x_i, x_j \rangle \right)$$

- 3 Make prediction for new x :

$$\hat{y} = \text{sign}[w^T x + w_0] = \text{sign}\left[\sum_{i \in S\mathcal{V}} \alpha_i^* y_i \langle x_i, x \rangle + w_0\right]$$

Making predictions

- 1 Solve dual task to find α_i^* , $i = 1, 2, \dots, N$

$$\begin{cases} L_D = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \rightarrow \max_{\alpha} \\ \sum_{i=1}^N \alpha_i y_i = 0 \\ 0 \leq \alpha_i \leq C \end{cases}$$

- 2 Find optimal w_0 :

$$w_0 = \frac{1}{n_{\tilde{S}V}} \left(\sum_{j \in \tilde{S}V} y_j - \sum_{j \in \tilde{S}V} \sum_{i \in SV} \alpha_i^* y_i \langle \mathbf{x}_i, \mathbf{x}_j \rangle \right)$$

- 3 Make prediction for new \mathbf{x} :

$$\hat{y} = \text{sign}[w^T \mathbf{x} + w_0] = \text{sign} \left[\sum_{i \in SV} \alpha_i^* y_i \langle \mathbf{x}_i, \mathbf{x} \rangle + w_0 \right]$$

- On all steps we don't need exact feature representations, only scalar products $\langle \mathbf{x}, \mathbf{x}' \rangle$!

Kernel trick generalization

- 1 Solve dual task to find α_i^* , $i = 1, 2, \dots, N$

$$\begin{cases} L_D = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j K(x_i, x_j) \rightarrow \max_{\alpha} \\ \sum_{i=1}^N \alpha_i y_i = 0 \\ 0 \leq \alpha_i \leq C \end{cases}$$

- 2 Find optimal w_0 :

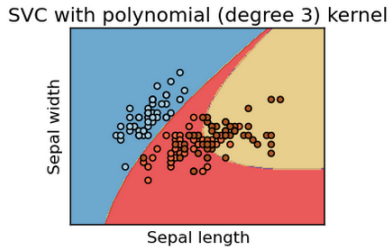
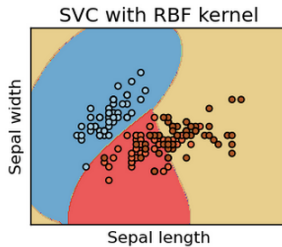
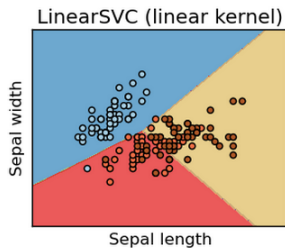
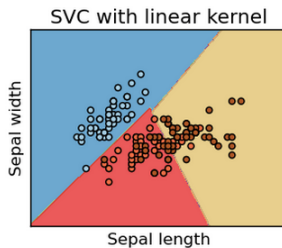
$$w_0 = \frac{1}{n_{\tilde{S}V}} \left(\sum_{j \in \tilde{S}V} y_j - \sum_{j \in \tilde{S}V} \sum_{i \in SV} \alpha_i^* y_i K(x_i, x_j) \right)$$

- 3 Make prediction for new x :

$$\hat{y} = \text{sign}[w^T x + w_0] = \text{sign}\left[\sum_{i \in SV} \alpha_i^* y_i K(x_i, x) + w_0\right]$$

- We replaced $\langle x, x' \rangle \rightarrow K(x, x')$ for $K(x, x') = \langle \phi(x), \phi(x') \rangle$ for some feature transformation $\phi(\cdot)$.

Kernel results



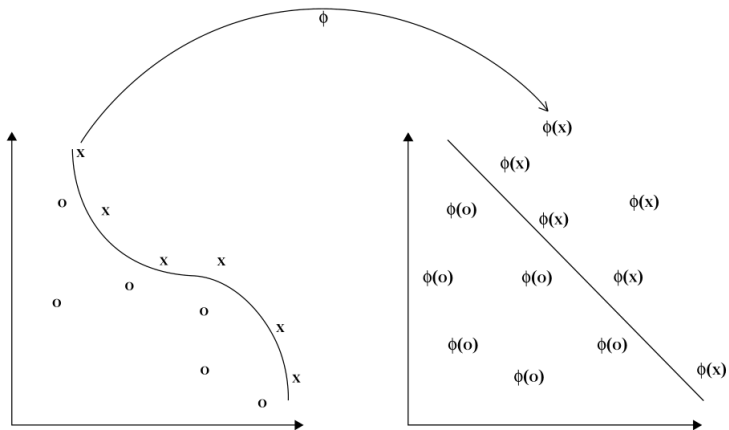
Kernel trick

Kernel trick

If solution depends from x_n only through pairwise scalar products $\langle x_i, x_j \rangle$ and $\langle x_i, x \rangle$, extend it by replacing $\langle x, z \rangle$ with Mercer kernel $K(x, z)$.

- $K(x, z)$ corresponds to ordinary scalar product $\langle \phi(x), \phi(z) \rangle$ after feature transformation $x \rightarrow \phi(x)$.
- Kernelizable algorithms: ridge regression, K-NN, K-means, PCA, SVM, ...
- Specific types of kernels:
 - $K(x, x') = K(x - x')$ - stationary kernels (invariant to translations)
 - $K(x, x') = K(\|x - x'\|)$ - radial basis functions

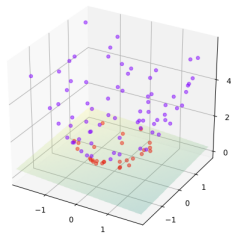
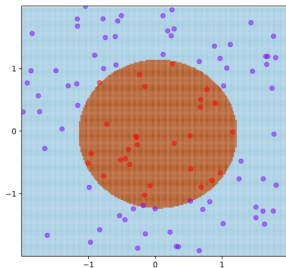
Illustration



Example of feature extension

Consider SVM with $\phi(x^1, x^2) = [x^1, x^2, (x^1)^2 + (x^2)^2]$.

- So $K(x, z) = \langle x, z \rangle + \|x\|^2 \|z\|^2$.



Example video.

Kernel trick use cases

Kernel trick use cases:

- x not enough, need transformation $\phi(x)$ for better prediction.
- $\langle \phi(x), \phi(z) \rangle$ - long to compute, but $K(x, z)$ can be computed easily.
 - applies when $\phi(x)$ is high dimensional (polynomial kernel) or infinite dimensional (rbf-kernel).
- cannot represent objects as fixed size vector, but natural scalar product (similarity function) $K(x, z)$ exists:
 - strings of different lengths ($K()$ depends on common substrings)
 - sets ($K()$ depends on sets intersection)
 - graphs ($K()$ depends on common subgraphs)
 - images of different sizes

Polynomial kernel¹

- Example 1: let $D = 2$.

$$\begin{aligned}K(x, z) &= (x^T z)^2 = (x_1 z_1 + x_2 z_2)^2 = \\&= x_1^2 z_1^2 + x_2^2 z_2^2 + 2x_1 z_1 x_2 z_2 \\&= \phi^T(x) \phi(z)\end{aligned}$$

for $\phi(x) = (x_1^2, x_2^2, \sqrt{2}x_1 x_2)$

¹What kind of feature transformation will correspond to $K(x, z) = (x^T z)^M$ for arbitrary M and D ?

Polynomial kernel²

- Example 2: let $D = 2$.

$$\begin{aligned}K(x, z) &= (1 + x^T z)^2 = (1 + x_1 z_1 + x_2 z_2)^2 = \\&= 1 + x_1^2 z_1^2 + x_2^2 z_2^2 + 2x_1 z_1 + 2x_2 z_2 + 2x_1 z_1 x_2 z_2 \\&= \phi^T(x) \phi(z)\end{aligned}$$

for $\phi(x) = (1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2)$

²What kind of feature transformation will correspond to $K(x, z) = (1 + x^T z)^M$ kernels for arbitrary M and D ?

Kernel properties

Theorem (Mercer): Function $K(x, x')$ is a Mercer kernel is and only if

- ① it is symmetric: $K(x, x') = K(x', x)$
- ② it is non-negative definite:
 - definition 1: for every function $g : X \rightarrow \mathbb{R}$

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

- definition 2 (equivalent): for every finite set x_1, x_2, \dots, x_M
Gramm matrix $\{K(x_i, x_j)\}_{i,j=1}^M \succeq 0$ (p.s.d.)

Kernel construction

- Kernel learning - separate field of study.
- Hard to prove non-negative definiteness of kernel in general.
- Kernels can be constructed from other kernels, for example from:
 - 1 scalar product $\langle x, z \rangle$
 - 2 constant $K(x, z) \equiv 1$
 - 3 $x^T A z$ for any $A \succ 0^3$

³Prove that it is a Mercer kernel. You may use Choletsky decomposition.

Constructing kernels from other kernels

If $K_1(x, z)$, $K_2(x, z)$ are arbitrary kernels, $c > 0$ is a constant, $q(\cdot)$ is a polynomial with non-negative coefficients, $h(x)$ and $\varphi(x)$ are arbitrary functions $\mathcal{X} \rightarrow \mathbb{R}$ and $\mathcal{X} \rightarrow \mathbb{R}^M$ respectively, then these are valid kernels⁴:

- ❶ $K(x, z) = cK_1(x, z)$
- ❷ $K(x, z) = K_1(x, z)K_2(x, z)$
- ❸ $K(x, z) = K_1(x, z) + K_2(x, z)$
- ❹ $K(x, z) = K_1(\varphi(x), \varphi(z))$
- ❺ $K(x, z) = h(x)K_1(x, z)h(z)$
- ❻ $K(x, z) = e^{K_1(x, z)}$

⁴prove some of these statements

Commonly used kernels

Kernel	Mathematical form
linear	$\langle x, z \rangle$
polynomial	$(\alpha \langle x, z \rangle + \beta)^M$
RBF	$\exp(-\gamma \ x - z\ ^2)$

- Parameter constraints: $\alpha > 0, \beta > 0, \gamma > 0, M = 1, 2, 3, \dots$
- Linear kernel reduces to method in its original form ($\phi(x) \equiv x$).
- Polynomial kernel corresponds to extension of x with polynomial features of order $\leq d$.
 - direct scalar product takes $O(C_{M+D}^D)$
 - $K(x, z)$ takes $O(D)$
- Gaussian kernel corresponds to $\phi(x)$ mapping to infinite dimensional space!

Kernelized distance⁵

- Kernelization of distance:

⁵How can we calculate distance between vectors $\phi(x)/\|\phi(x)\|$ and $\phi(z)/\|\phi(z)\|$?

Kernelized distance⁵

- Kernelization of distance:

$$\begin{aligned}\rho(x, z)^2 &= \langle \phi(x) - \phi(z), \phi(x) - \phi(z) \rangle \\ &= \langle \phi(x), \phi(x) \rangle + \langle \phi(z), \phi(z) \rangle - 2\langle \phi(x), \phi(z) \rangle \\ &= K(x, x) + K(z, z) - 2K(x, z)\end{aligned}$$

- So all distance based algorithms are kernelizable: K-NN, K-means? nearest centroid, PCA, etc.

⁵How can we calculate distance between vectors $\phi(x)/\|\phi(x)\|$ and $\phi(z)/\|\phi(z)\|$?

Table of Contents

1 Recap of SVM solution

2 Kernel SVM parameters

Kernel SVM prediction

Kernel SVM prediction for x :

$$\hat{y}(x) = \text{sign}[w^T x + w_0] = \text{sign}\left[\sum_{i \in \mathcal{SV}} \alpha_i^* y_i K(x_i, x) + w_0\right]$$

- $\alpha_i^* = 0$ for non-informative vectors
- $\alpha_i^* = C$ for violating support vectors
- $\alpha_i^* \in [0, C]$ for boundary support vectors

Solution for kernels:

- $(\alpha \langle x, z \rangle + \beta)^M$
- $\exp(-\gamma \|x - z\|^2)$

Kernel SVM prediction

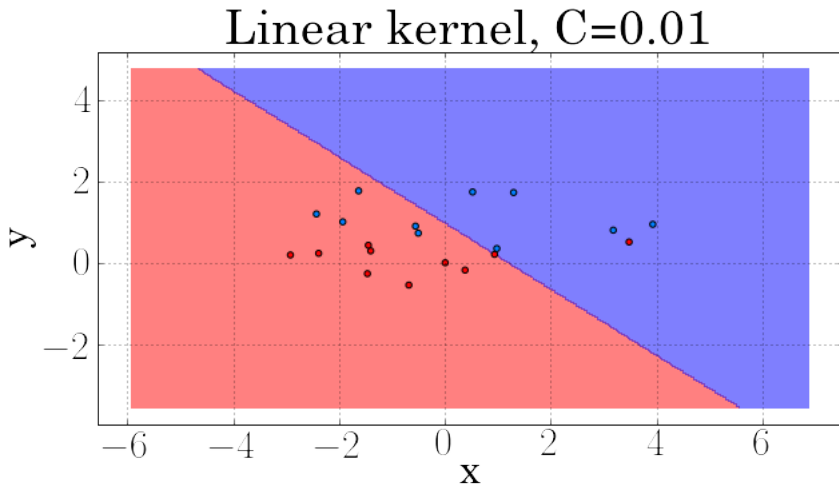
Kernel SVM prediction for x :

$$\hat{y}(x) = \text{sign}[w^T x + w_0] = \text{sign}\left[\sum_{i \in \mathcal{SV}} \alpha_i^* y_i K(x_i, x) + w_0\right]$$

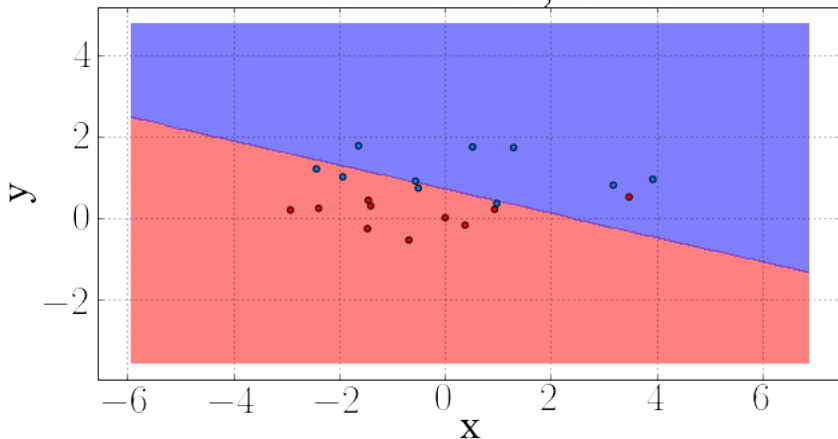
- $\alpha_i^* = 0$ for non-informative vectors
- $\alpha_i^* = C$ for violating support vectors
- $\alpha_i^* \in [0, C]$ for boundary support vectors

Solution for kernels:

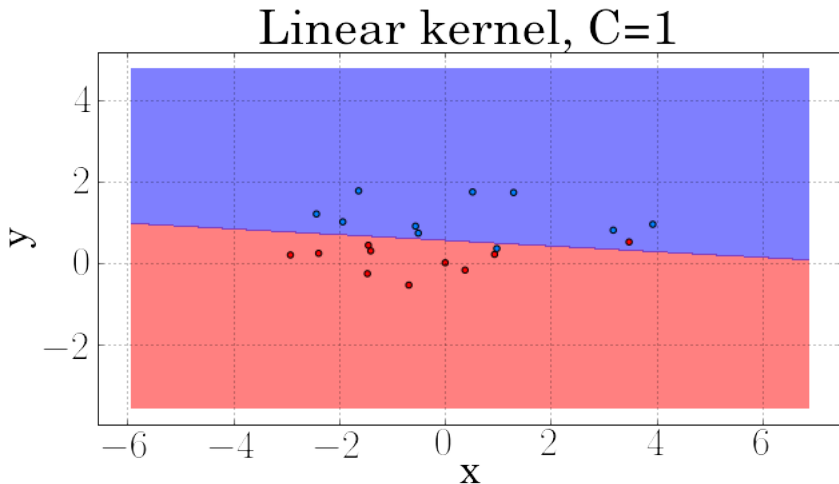
- $(\alpha \langle x, z \rangle + \beta)^M$ - polynomial degree M boundary.
- $\exp(-\gamma \|x - z\|^2)$ - weighted Parzen window method among support vectors.

Linear kernel - variable C 

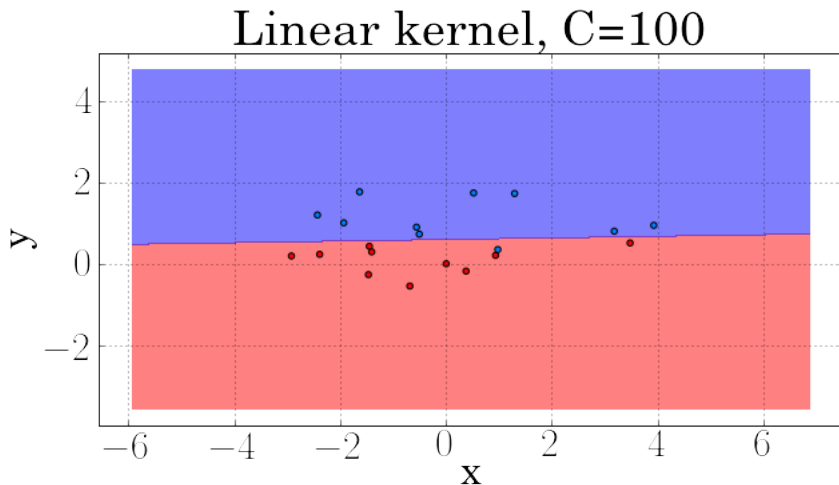
Linear kernel - variable C

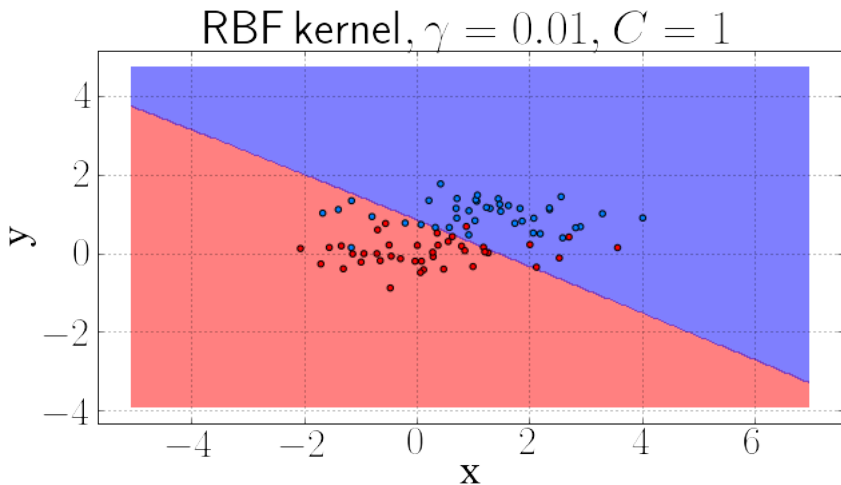
Linear kernel, $C=0.1$ 

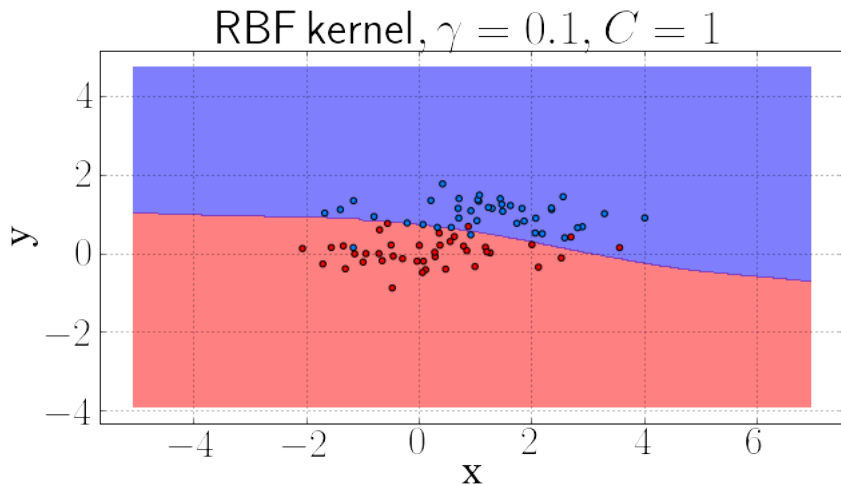
Linear kernel - variable C

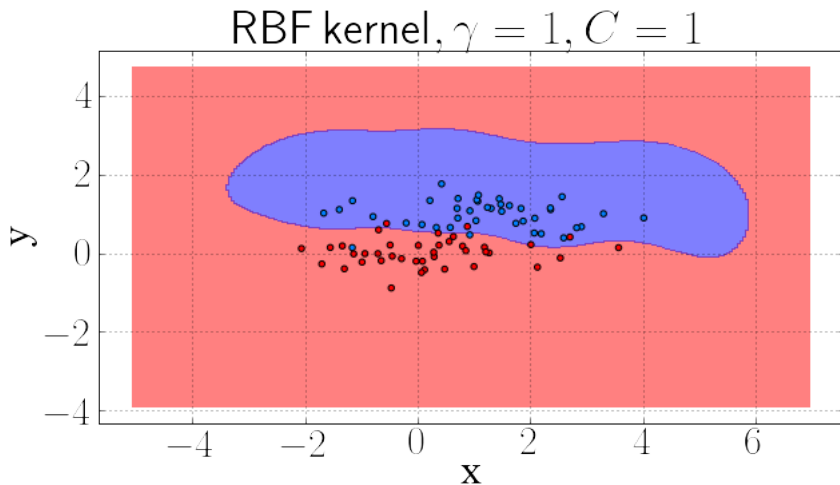


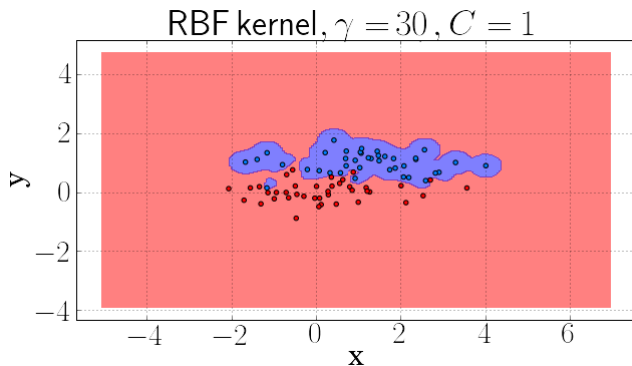
Linear kernel - variable C



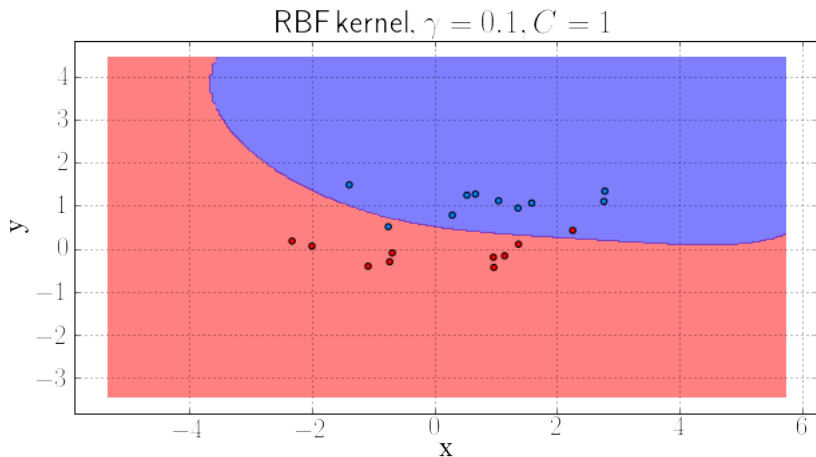
RBF kernel - variable γ 

RBF kernel - variable γ 

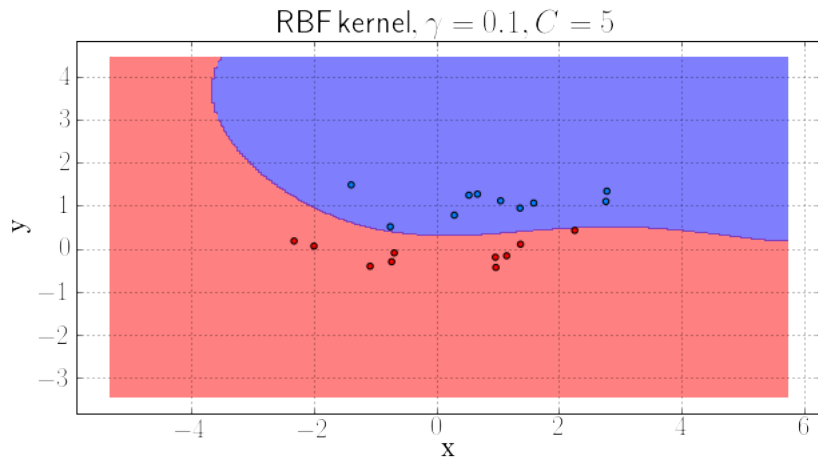
RBF kernel - variable γ 

RBF kernel - variable γ 

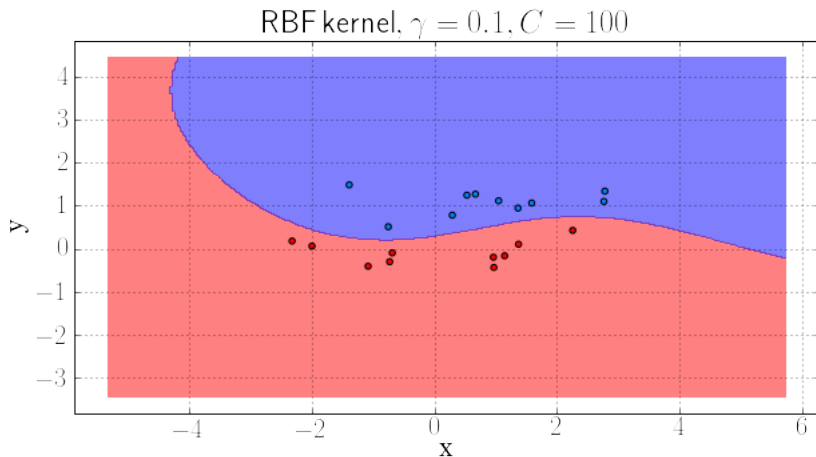
RBF kernel - variable C



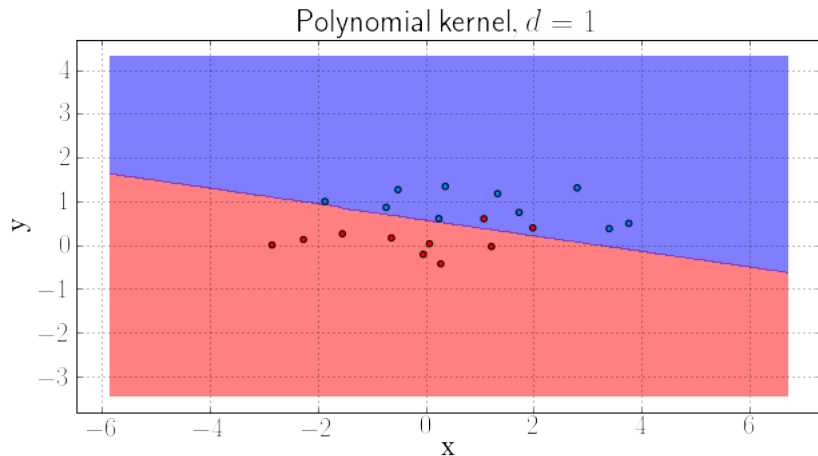
RBF kernel - variable C

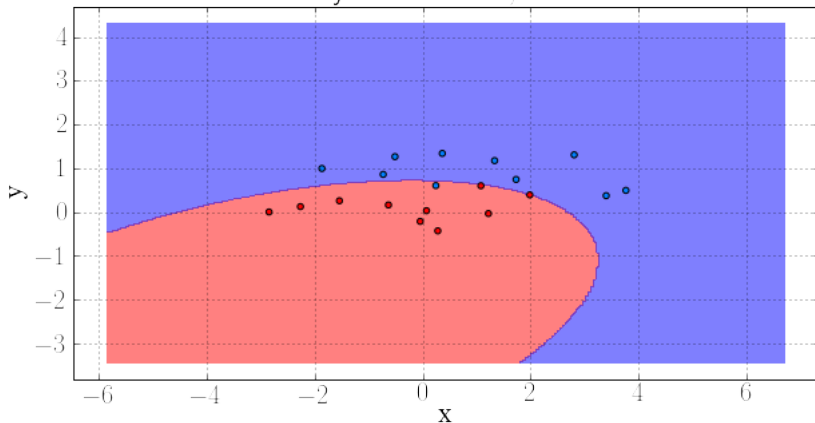


RBF kernel - variable C

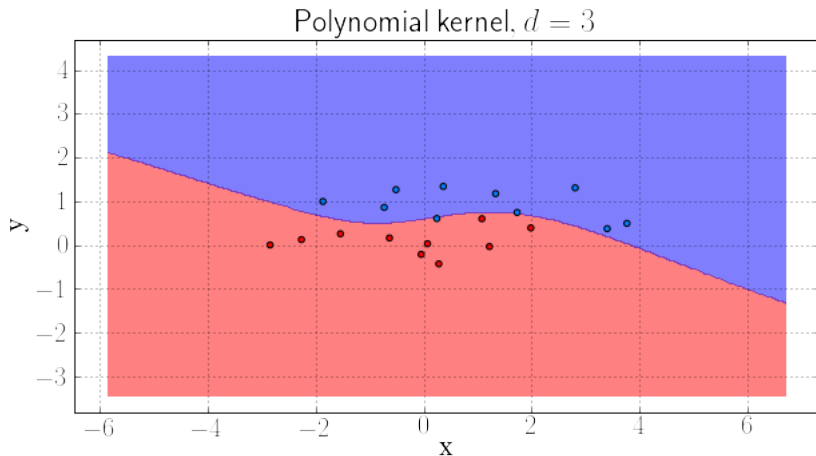


Polynomial kernel - variable d

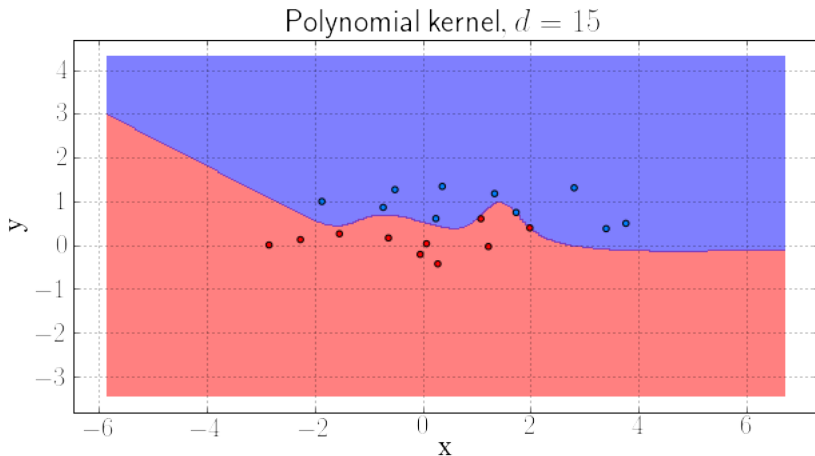


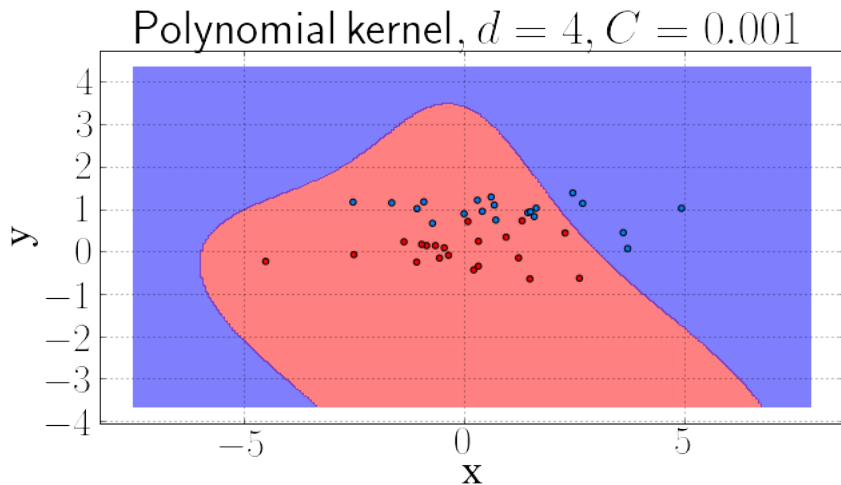
Polynomial kernel, $d = 2$ 

Polynomial kernel - variable d

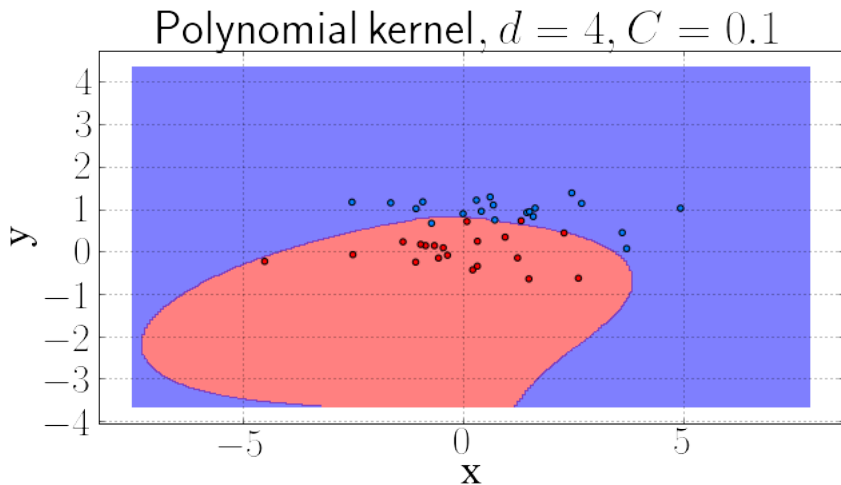


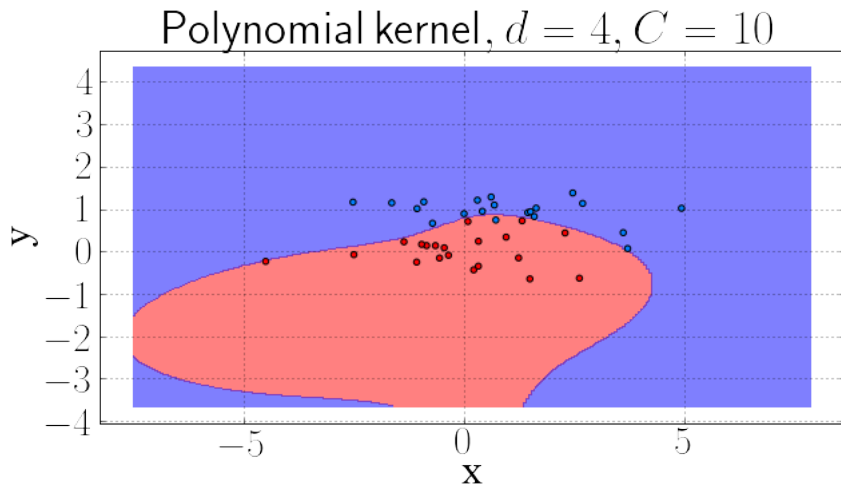
Polynomial kernel - variable d



Polynomial kernel - variable C 

Polynomial kernel - variable C



Polynomial kernel - variable C 

Summary

- Kernel trick applies when:
 - x not enough, need transformation $\phi(x)$ for better prediction.
 - $\langle \phi(x), \phi(z) \rangle$ is long to compute, $K(x, z)$ is fast to compute.
 - cannot represent objects as fixed size vector, but natural scalar product (similarity function) $K(x, z)$ exists.
- Kernelizable algorithms: SVM, ridge regression, K-NN, K-means, PCA and more.
- Mostly used kernels: polynomial, RBF.
- Mercer theorem: condition for $K(x, z)$ to be a kernel.
- Kernels can be constructed from other kernels.