Kernel trick

Victor Kitov
v.v.kitov@yandex.ru

Yandex School of Data Analysis

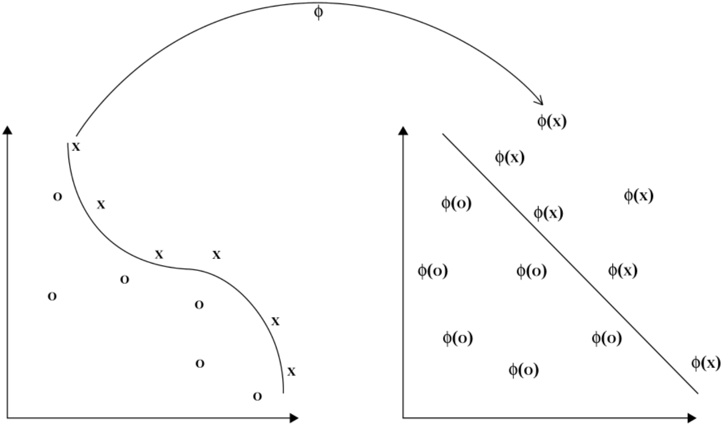
Kernel trick

Perform feature transformation: $x \to \phi(x)$. Scalar product becomes $\langle x, x' \rangle \to \langle \phi(x), \phi(x') \rangle = K(x, x')$

Kernel trick

Define not the feature representation x but only scalar product function $K(x, x')$

Illustration



Kernelizable algorithms[1]

- ridge regression:
- K-NN
- K-means
- PCA
- SVM
- etc...

[1] Prove tht $x^T Ax$ is a kernel for any $A \succcurlyeq 0$.

Kernel trick use cases

- high-dimensional data
  - polynomial of order up to M
  - Gaussian kernel $K(x, x') = e^{-\frac{1}{2\sigma^2}\left\|x-x'\right\|^2}$ corresponds to infinite-dimensional feature space.
- hard to vectorize data
  - strings, sets, images, texts, graphs, 3D-structures, sequences, etc.
- natural scalar product exist
  - strings: number of co-occuring substrings
  - sets: size of intersection of sets
    - example: for sets S_1 and S_2: $K(S_1, S_2) = 2^{|S_1 \cap S_2|}$ is a possible kernel.
  - etc.
- scalar product can be computed efficiently

General motivation for kernel trick

- perform generalization of linear methods to non-linear case
  - we use efficiency of linear methods
  - local minimum is global minimum
    - no local optima=>less overfitting
- non-vectorial objects
  - hard to obtain vector representation

Polynomial kernel[2]

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

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

Polynomial kernel[3]

- Example 2: let $D=2$.

$$\begin{aligned} K(x,z) &= (1+x^Tz)^2 = (1+x_1z_1+x_2z_2)^2 = \\ &= 1+x_1^2z_1^2+x_2^2z_2^2+2x_1z_1+2x_2z_2+2x_1z_1x_2z_2 \\ &= \phi^T(x)\phi(z) \end{aligned}$$

for $\phi(x)=(1,\, x_1^2,\, x_2^2,\, \sqrt{2}x_1,\, \sqrt{2}x_2,\, \sqrt{2}x_1x_2)$

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

Kernel properties

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

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

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

    - definition 2 (equivalent): for every finite set $x_1, x_2, ... x_M$ 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 definitness of kernel in general.
- Kernels can be constructed from other kernels, for example from:

1. scalar product $\langle x, x' \rangle$
2. constant $K(x, x') \equiv 1$
3. $x^T A x$ for any $A \succcurlyeq 0$.

Constructing kernels from other kernels

If $K_1(x,x')$, $K_2(x,x')$ 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 $\mathcal{X}\to\mathbb{R}$ and $\mathcal{X}\to\mathbb{R}^M$ respectively, then these are valid kernels[4]:

1. $K(x,x') = cK_1(x,x')$
2. $K(x,x') = K_1(x,x')K_2(x,x')$
3. $K(x,x') = K_1(x,x') + K_2(x,x')$
4. $K(x,x') = K_1(\varphi(x),\varphi(x'))$
5. $K(x,x') = h(x)K_1(x,x')h(x')$
6. $K(x,x') = e^{K_1(x,x')}$

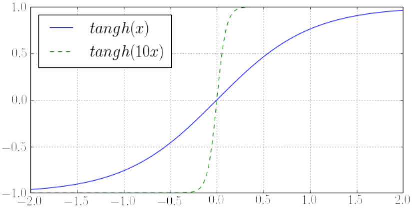
[4] prove some of these statements

Commonly used kernels

Let x and x' be two objects and take any $\gamma > 0, r > 0, d > 0$.

Kernel	Mathematical form
linear	$\langle x, x' \rangle$
polynomial	$(\gamma \langle x, x' \rangle + r)^d$
RBF	$\exp(-\gamma \left\| x - x' \right\|^2)$

- Standard transformation is also sigmoid=tangh$(\gamma \langle x, y \rangle + r)$ but its not a Mercer kernel.



Addition[5]

- Other kernelized algorithms: K-NN, K-means, K-medoids, nearest medoid, PCA, SVM, etc.
- Kernelization of distance:

[5] How can we calculate scalar product between normalized (unit norm) vectors $\phi(x)$ and $\phi(x')$?

Addition[5]

- Other kernelized algorithms: K-NN, K-means, K-medoids, nearest medoid, PCA, SVM, etc.
- Kernelization of distance:

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

[5] How can we calculate scalar product between normalized (unit norm) vectors $\phi(x)$ and $\phi(x')$?

Table of Contents

Making predictions

1. Solve dual task to find α_i^*, $i = 1, 2, ...N$

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

2. Find optimal w_0:

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

3. Make prediction for new x:

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

Making predictions

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

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

2. Find optimal w_0:

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

3. Make prediction for new x:

$$\widehat{y} = \text{sign}[w^T x + w_0] = \text{sign}[\sum_{i \in \mathcal{SV}} \alpha_i^* y_i \langle x_i, x \rangle + w_0]$$

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

Kernel trick generalization

1. Solve dual task to find α_i^*, $i = 1, 2, ... N$

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

2. Find optimal w_0:

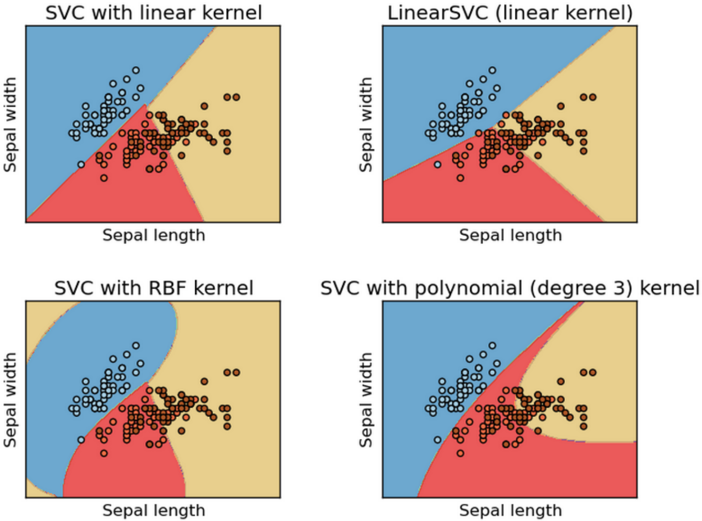
$$w_0 = \frac{1}{n_{\tilde{SV}}}\left(\sum_{j \in \tilde{SV}} y_j - \sum_{j \in \tilde{SV}} \sum_{i \in \mathcal{SV}} \alpha_i^* y_i K(x_i, x_j) \right)$$

3. Make prediction for new x:

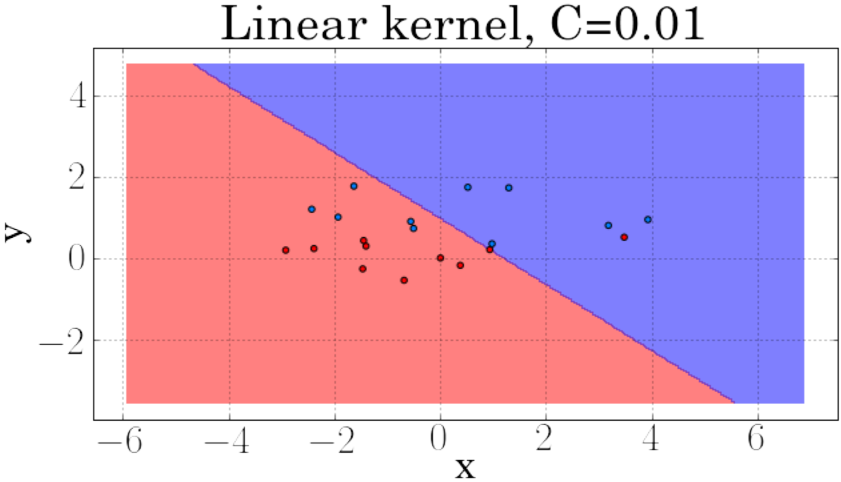
$$\widehat{y} = \text{sign}[w^T x + w_0] = \text{sign}[\sum_{i \in \mathcal{SV}} \alpha_i^* y_i K(x_i, x_j) + w_0]$$

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

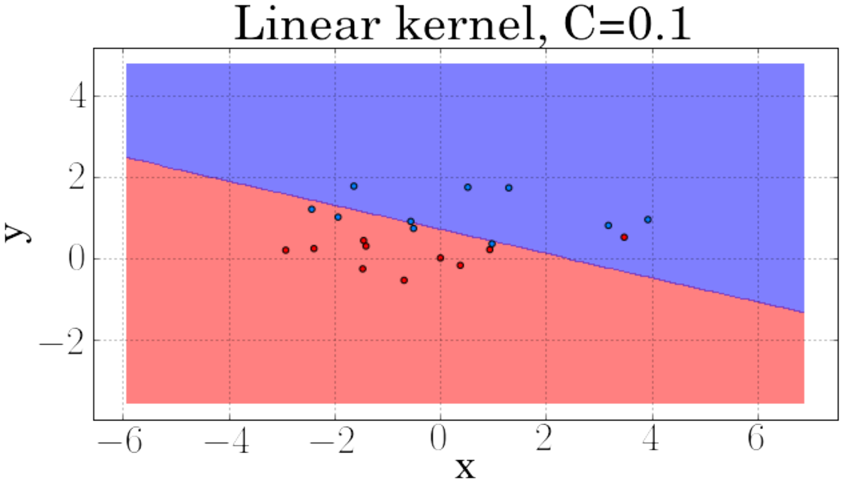
Kernel results



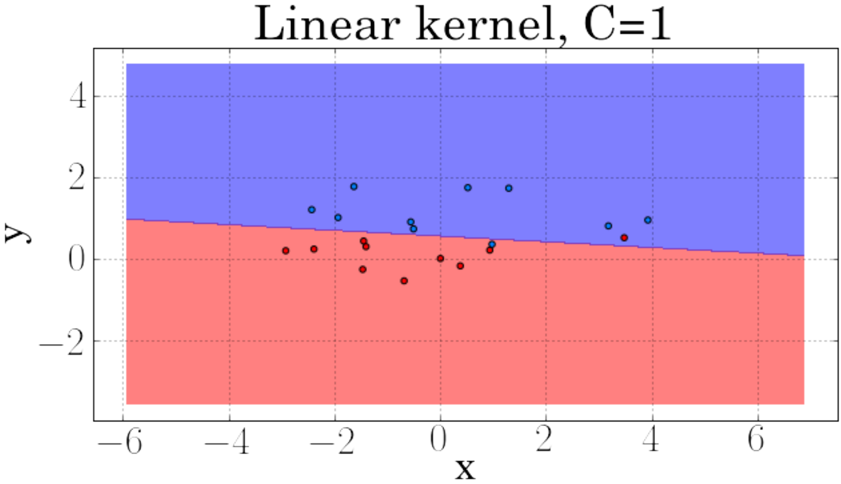
Linear kernel - variable C



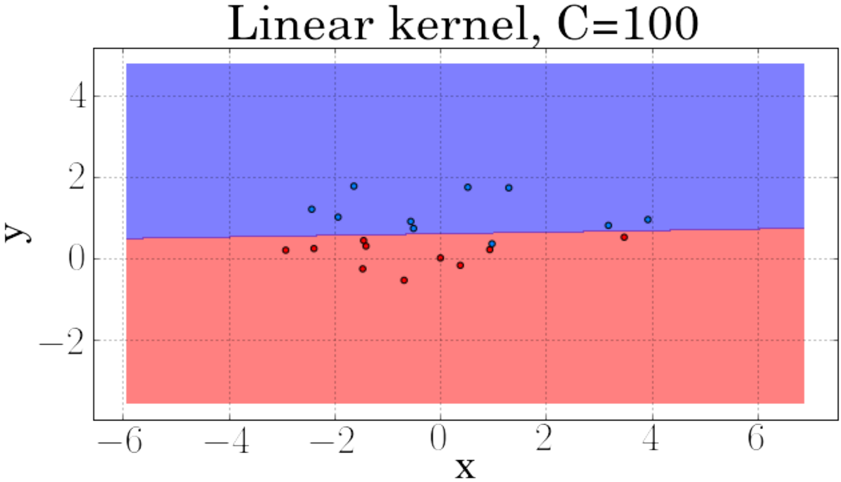
Linear kernel - variable C



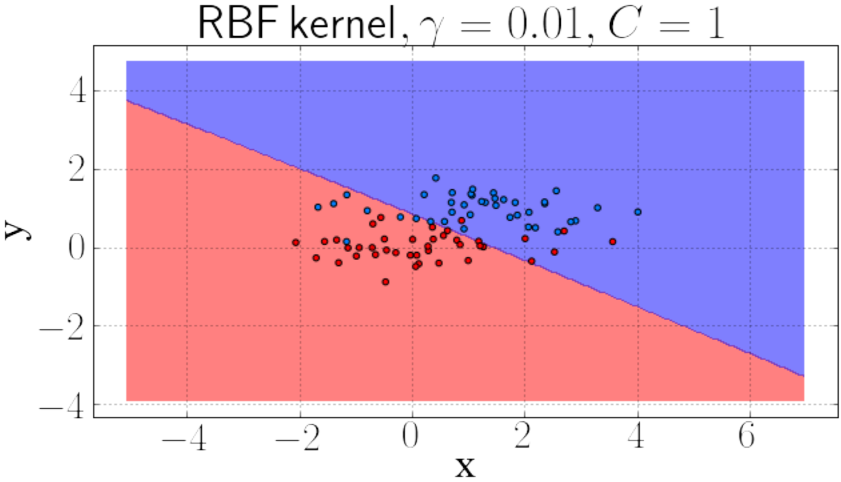
Linear kernel - variable C



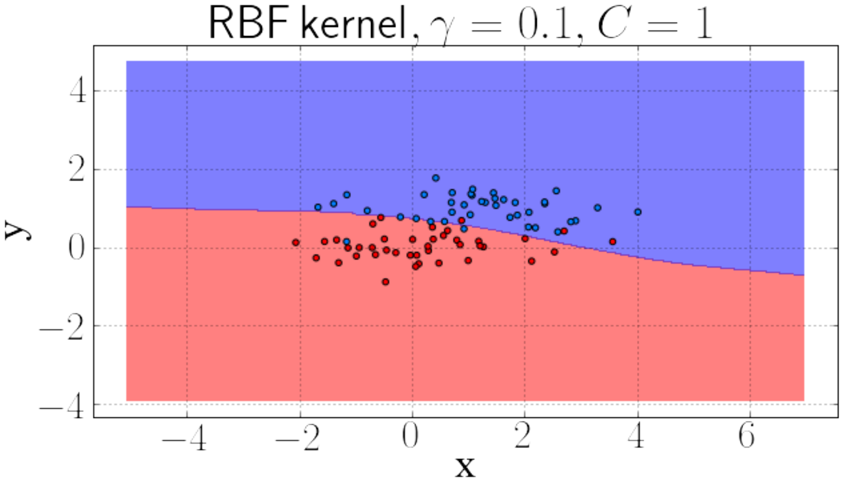
Linear kernel - variable C



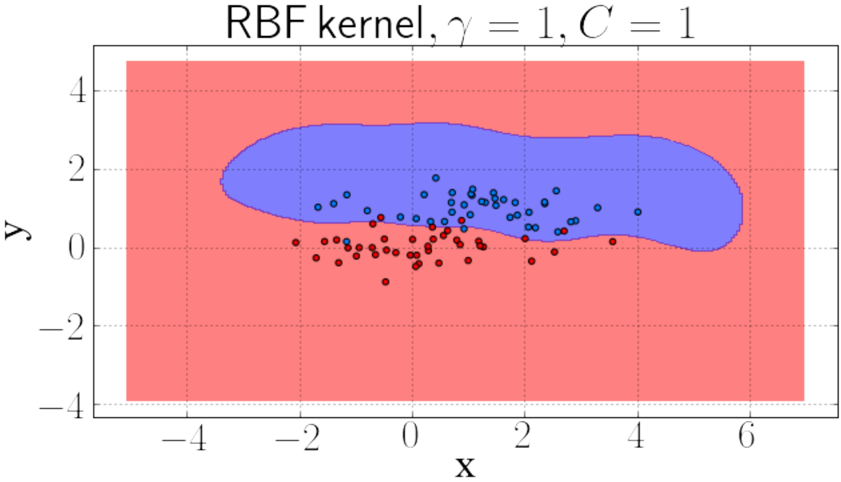
RBF kernel - variable γ



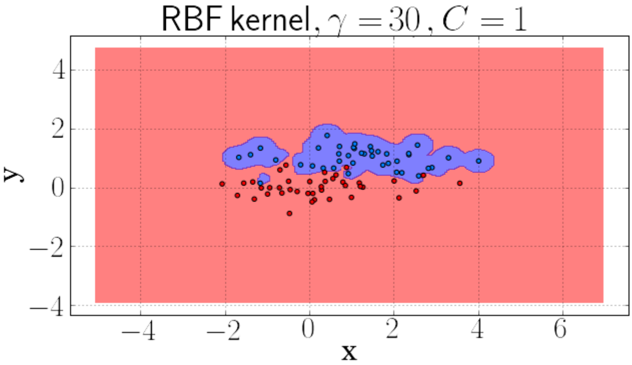
RBF kernel - variable γ



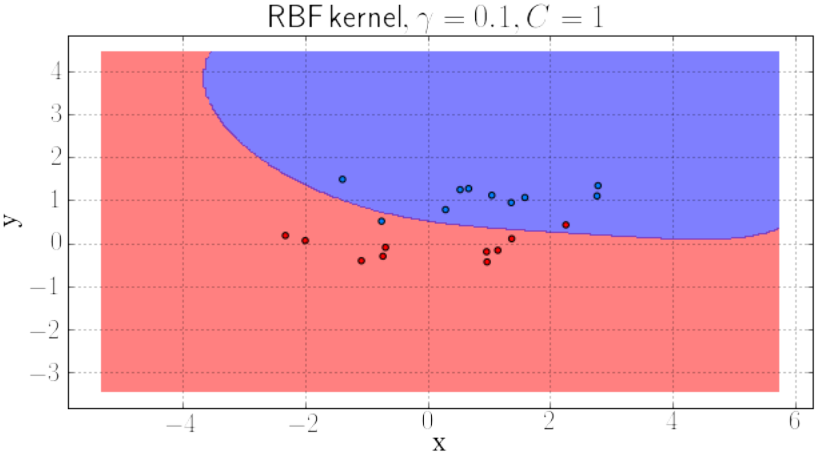
RBF kernel - variable γ



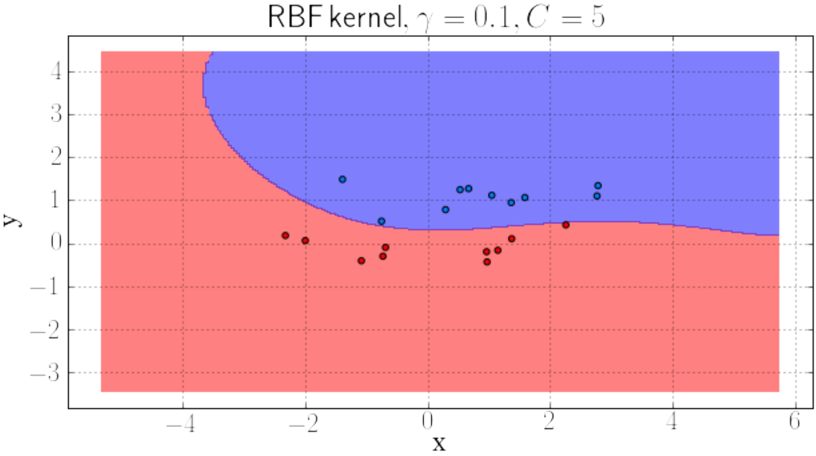
RBF kernel - variable γ



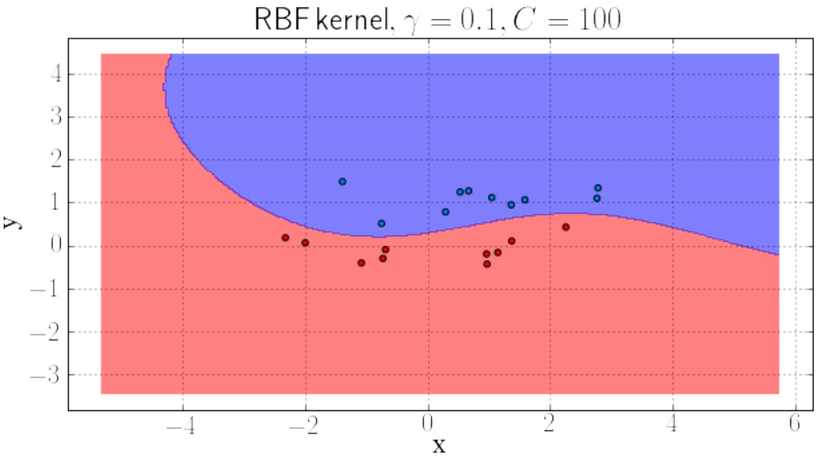
RBF kernel - variable C



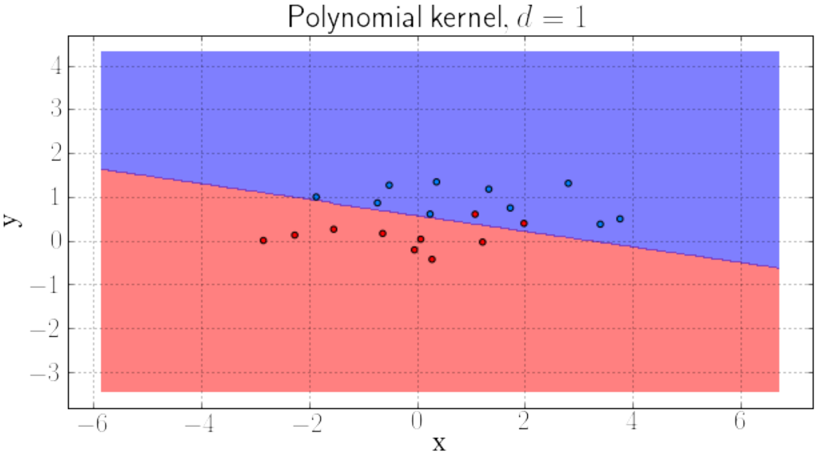
RBF kernel - variable C



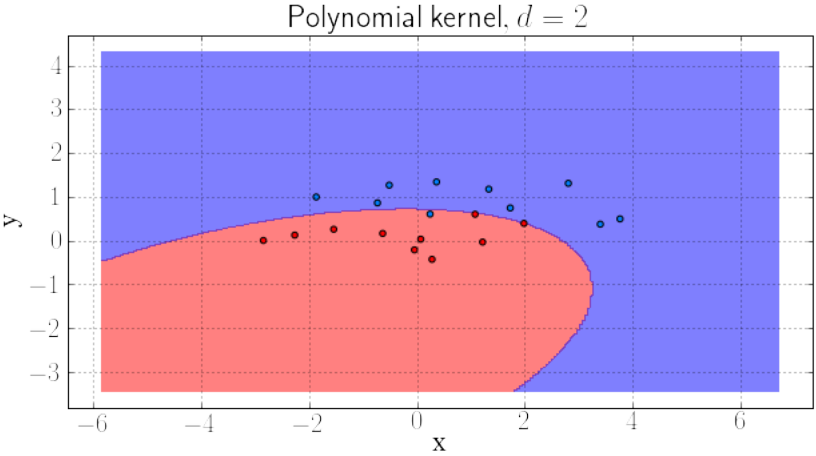
RBF kernel - variable C



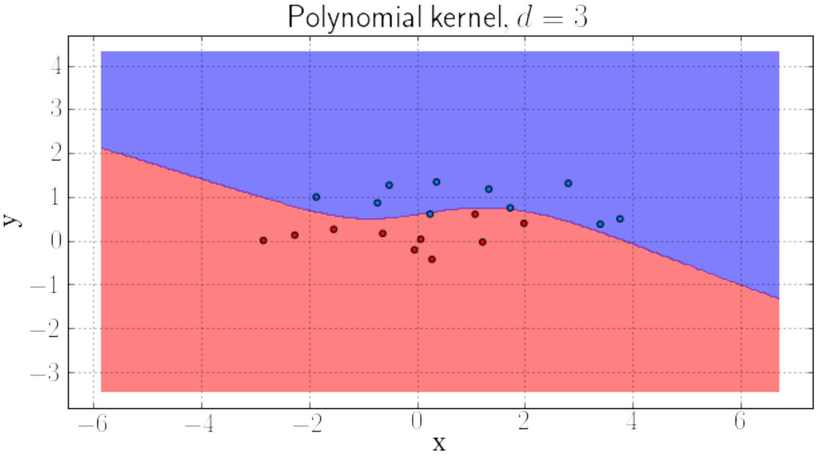
Polynomial kernel - variable d



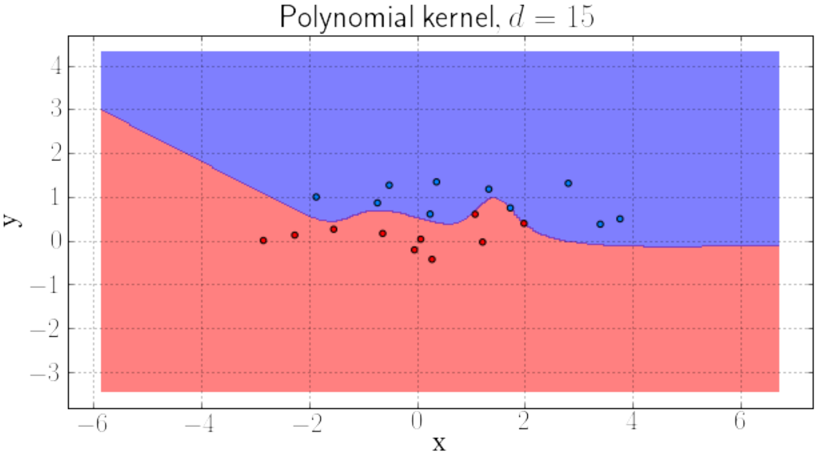
Polynomial kernel - variable d



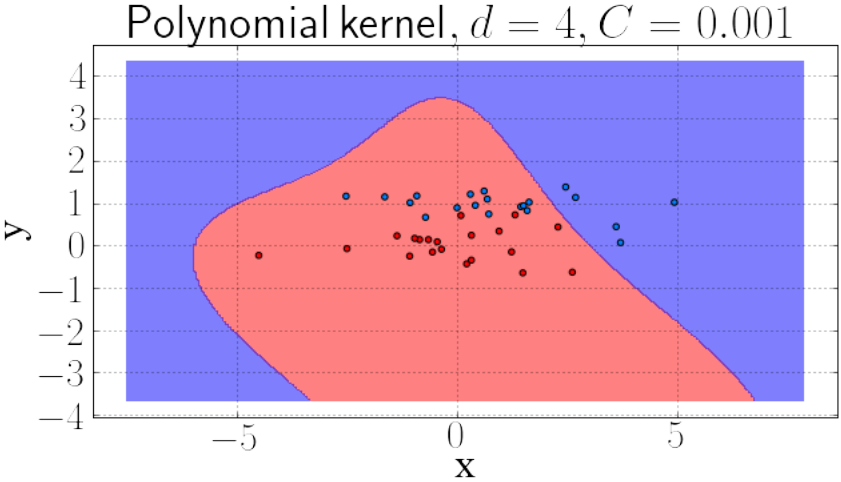
Polynomial kernel - variable d



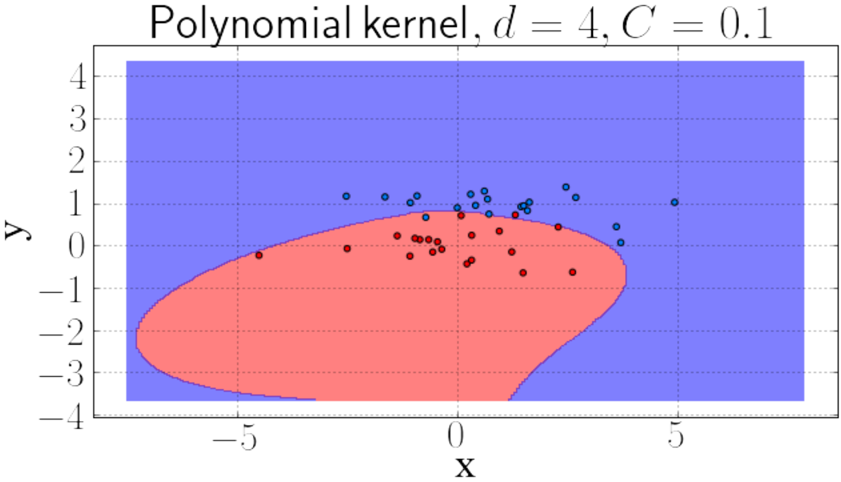
Polynomial kernel - variable d



Polynomial kernel - variable C



Polynomial kernel - variable C



Polynomial kernel - variable C