Kernel trick

Victor Kitov

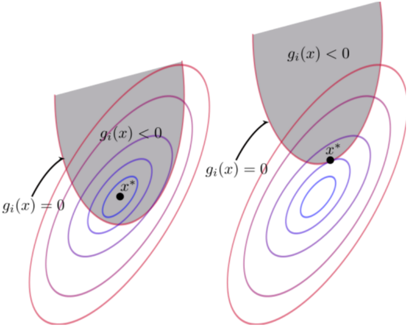
v.v.kitov@yandex.ru

Table of Contents

Kuhn-Takker conditions

Consider the optimization task:

$$\begin{cases} f(x) \to \min_x \\ g_m(x) \le 0 \qquad m = \overline{1, M} \end{cases} \tag{1}$$



Necessary conditions for optimality

Define Largangian

$$L(x,\lambda)=f(x)+\sum_{m=1}^{M}\lambda_m g_m(x)$$

Theorem (necessary conditions for optimality):

- Let x^* be the solution to (1),
- $f(x^*)$ and $g_m(x^*)$, $m=1,2,...M$ - continuously differentiable at x^*.
- Slater regularity satisfied: $\exists x:\ g_m(x)<0\,\forall m$.

Then coefficients $\lambda_1^*,\lambda_2^*,...\lambda_M^*$ exist, such that x^* satisfies the conditions for $m=\overline{1,M}$:

$$\begin{cases} \nabla_x f(x^*)+\sum_{i=1}^{M}\lambda_i^*\nabla_x g_i(x^*)=0 & \text{stationarity} \\ g_m(x^*)\le 0 & \text{feasibility} \\ \lambda_m^*\ge 0 & \text{non-negativity} \\ \lambda_m^* g_m(x^*)=0 & \text{comp.slackness} \end{cases} \quad (2)$$

Kuhn-Takker conditions

- Suppose $f(x)$ and $g_m(x)$, $m = \overline{1, M}$ are convex. Then
  1. Kuhn-Takker conditions (2) become **sufficient** for x^* to be the solution of (1).
  2. (x^*, λ^*) form the **saddle point for Lagrangian**:
  $$L(x^*, \lambda) \le L(x^*, \lambda^*) \le L(x, \lambda^*) \quad \forall x \, \forall \lambda \in \mathbb{R}_+^M$$
  3. May find $x^* = x(\lambda^*)$ from $\nabla_x L(x^*, \lambda^*) = 0$. Since $L(x^*, \lambda^*)$ is saddle point, find λ^* from dual task:
  $$\begin{cases} L(x(\lambda), \lambda) \to \max_\lambda & \\ g_m(x(\lambda)) \le 0 & m = \overline{1, M} \\ \lambda_m \ge 0 & m = \overline{1, M} \\ \lambda_m g_m(x(\lambda)) = 0 & m = \overline{1, M} \end{cases}$$

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:

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

Making predictions

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

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

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:

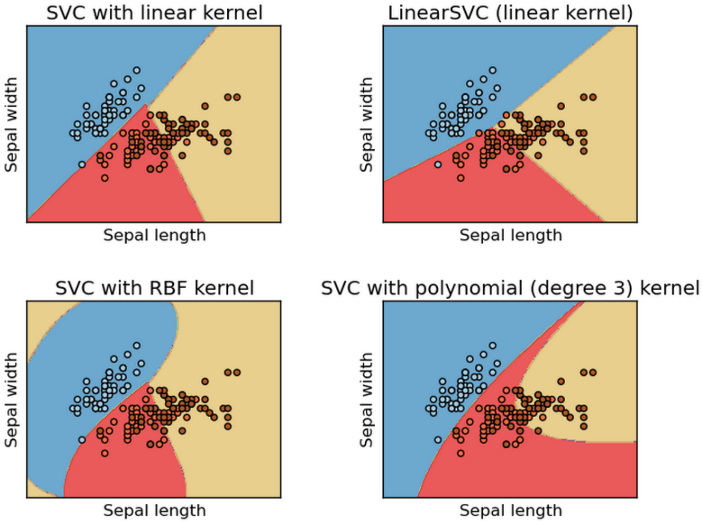
$$w_0 = \frac{1}{n_{\tilde{SV}}}\left(\sum_{j \in \tilde{SV}} y_j - \sum_{j \in \tilde{SV}} \sum_{i \in \mathcal{SV}} \alpha_i^* y_i K(x_i, x_j)\right)$$

3. Make prediction for new x:

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

- 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



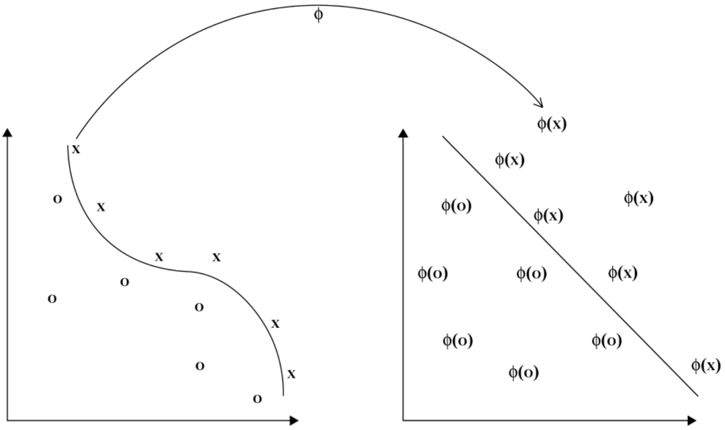
Kernel trick

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 $x \to \phi(x)$.
- 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

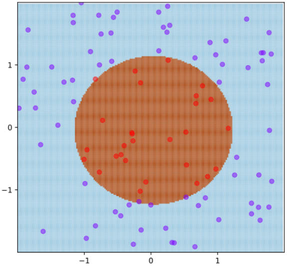
Illustration

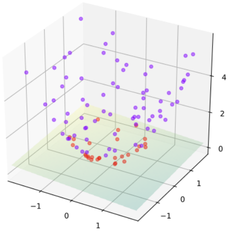


Example of feature extension

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 kenrel) or infinite dimensional (rbf-kernel).
- cannot represent objects as fixed size vector, but natural scalar prodict (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 kernel[1]

- Example 1: let $D=2$.

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

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

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

Polynomial kernel[2]

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

[2] What 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 $g:\ X\to\mathbb{R}$
   $$\int_X\int_X K(x,x')g(x)g(x')dxdx'\geq 0$$
   - 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 $\langle x, z \rangle$
  2. constant $K(x, z) \equiv 1$
  3. $x^T Az$ for any $A \succ 0$[3]

[3] Prove 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 $\mathcal{X}\to\mathbb{R}$ and $\mathcal{X}\to\mathbb{R}^M$ respectively, then these are valid kernels[4]:

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

[4] prove some of these statements

Commonly used kernels

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

- 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 $O\left(C_{M+D}^{D}\right)$
  - $K(x, z)$ takes $O(D)$
- Gaussian kernel corresponds to $\phi(x)$ mapping to infinite dimensional space!

Kernelized distance[5]

- Kernelization of distance:

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

Kernelized distance[5]

- 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.

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

Table of Contents

Kernel SVM prediction

Kernel SVM prediction for x:

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

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

Solution for kernels:

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

Kernel SVM prediction

Kernel SVM prediction for x:

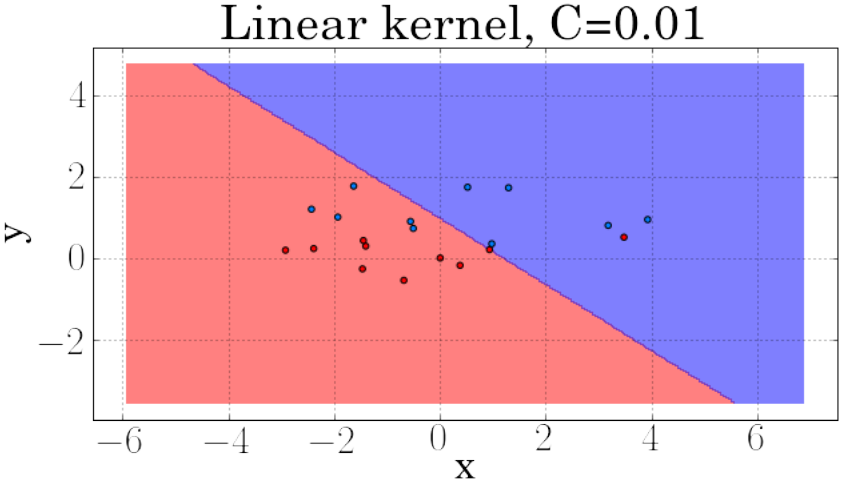
$$\widehat{y}(x) = \text{sign}[w^T x + w_0] = \text{sign}[\sum_{i \in \mathcal{SV}} \alpha_i^* y_i K(x_i, x) + w_0]$$

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

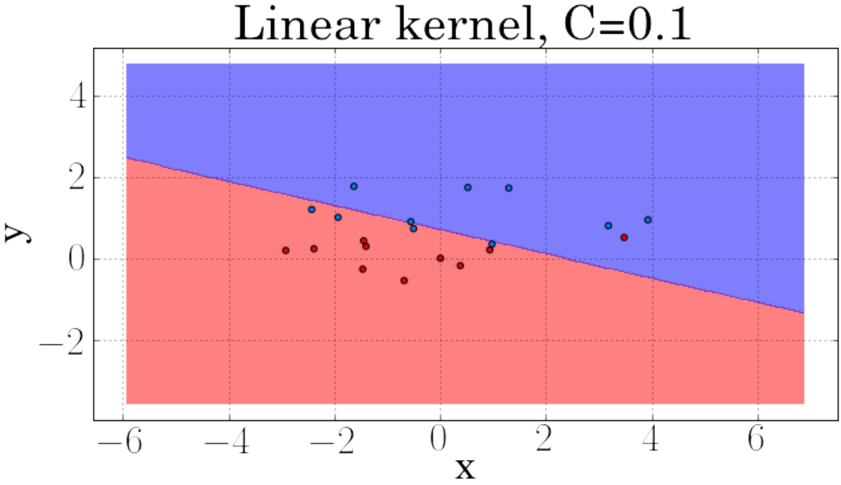
Solution for kernels:

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

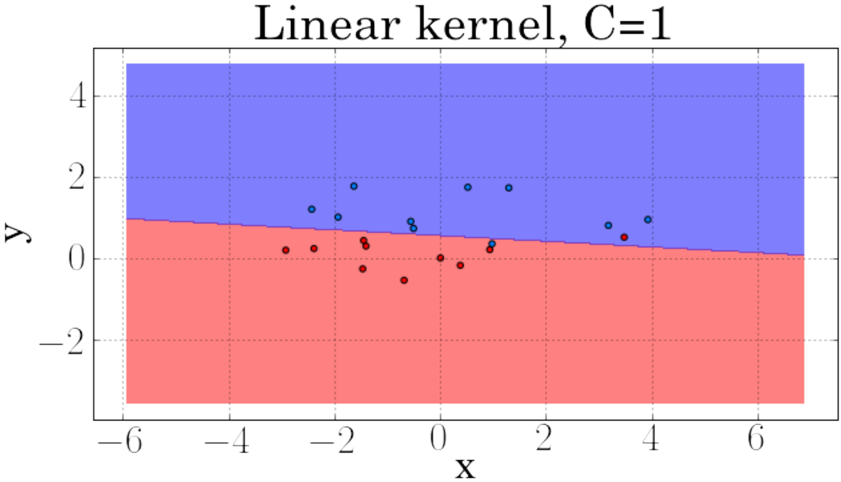
Linear kernel - variable C



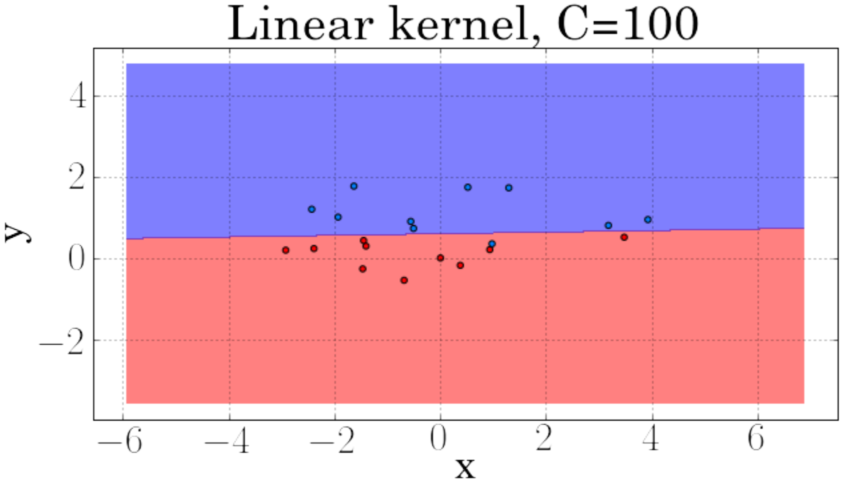
Linear kernel - variable C



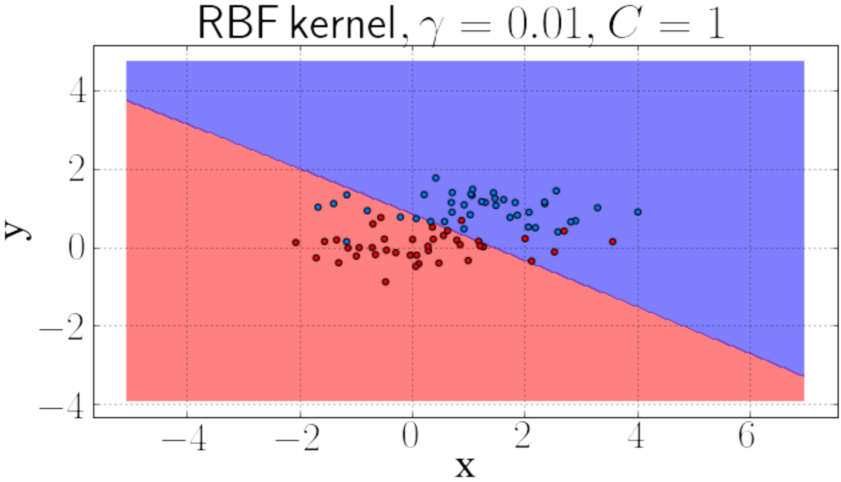
Linear kernel - variable C



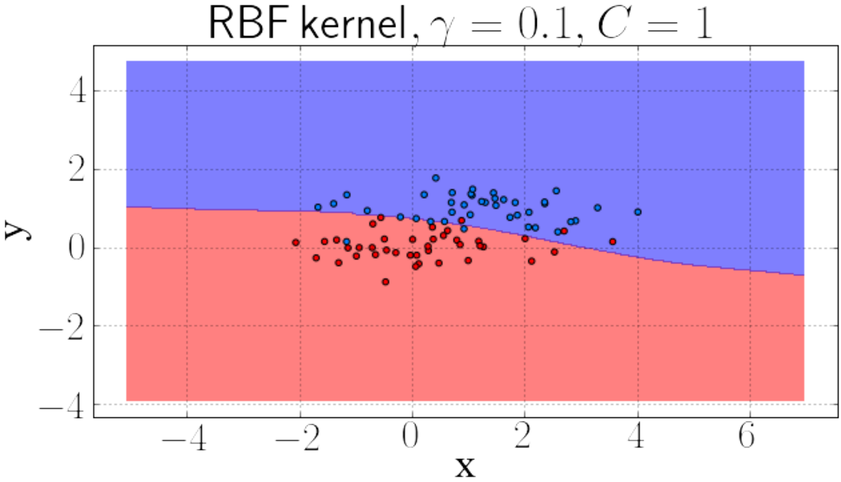
Linear kernel - variable C



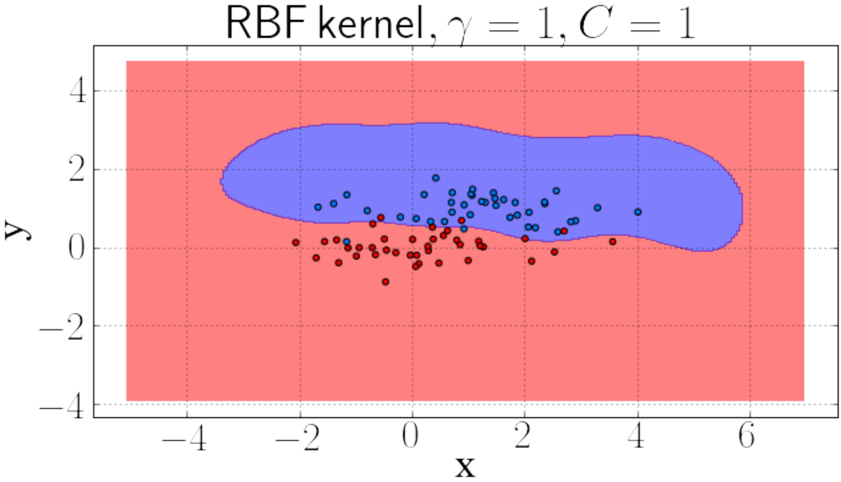
RBF kernel - variable γ



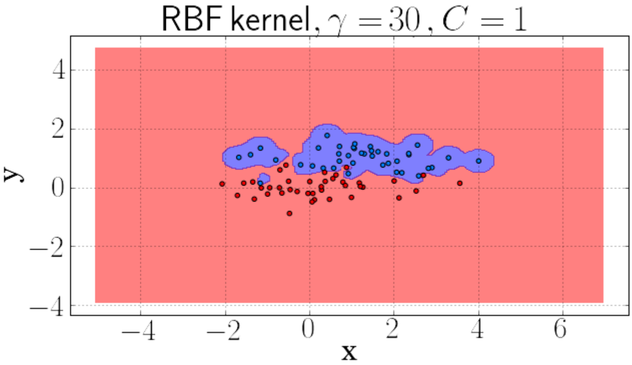
RBF kernel - variable γ



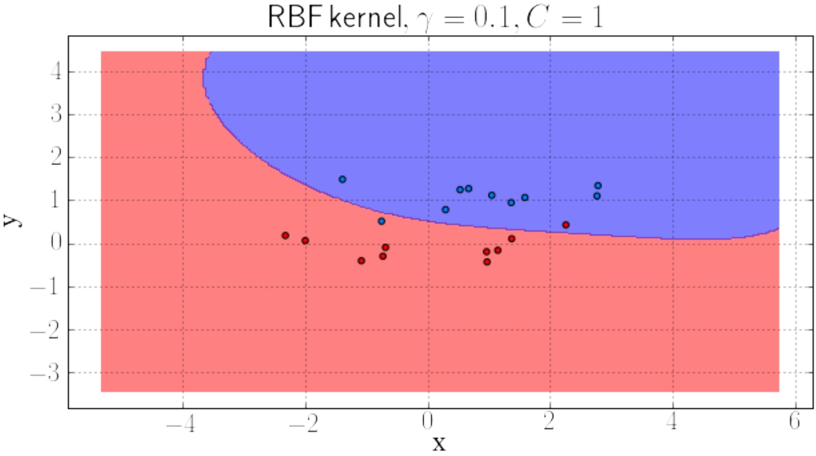
RBF kernel - variable γ



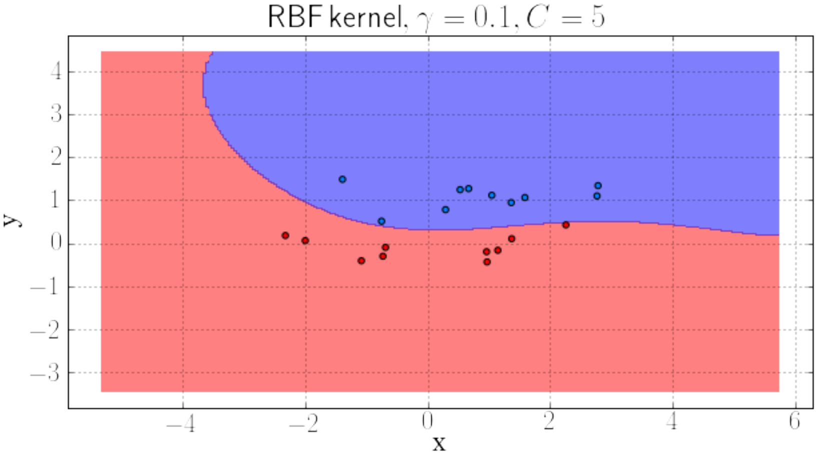
RBF kernel - variable γ



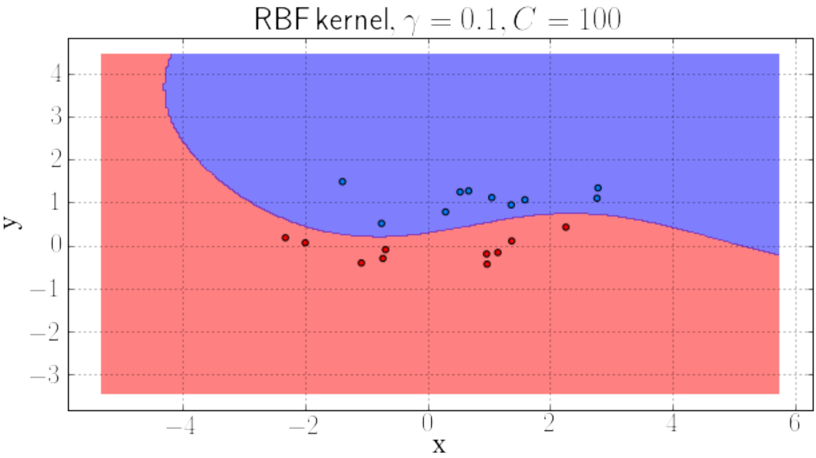
RBF kernel - variable C



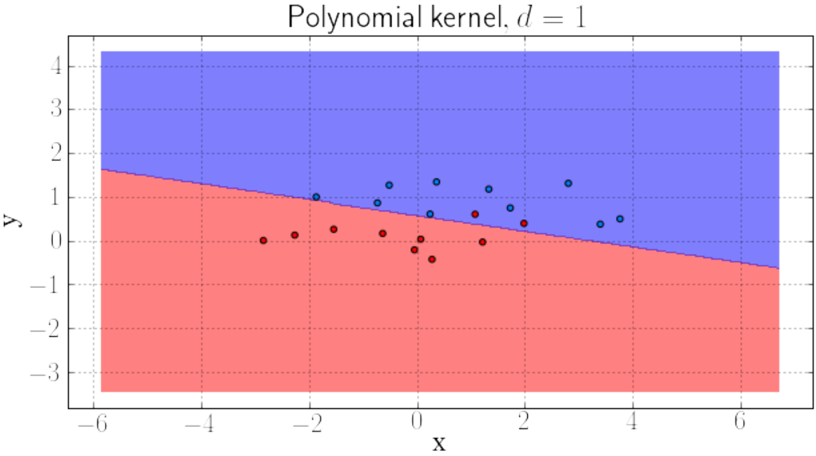
RBF kernel - variable C



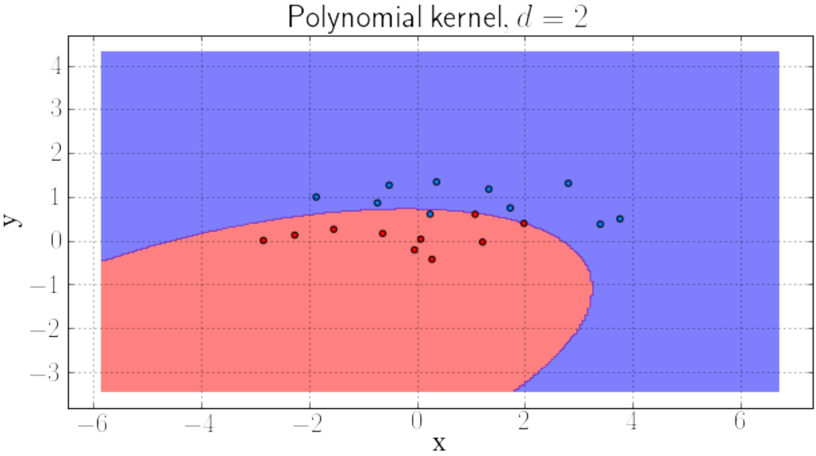
RBF kernel - variable C



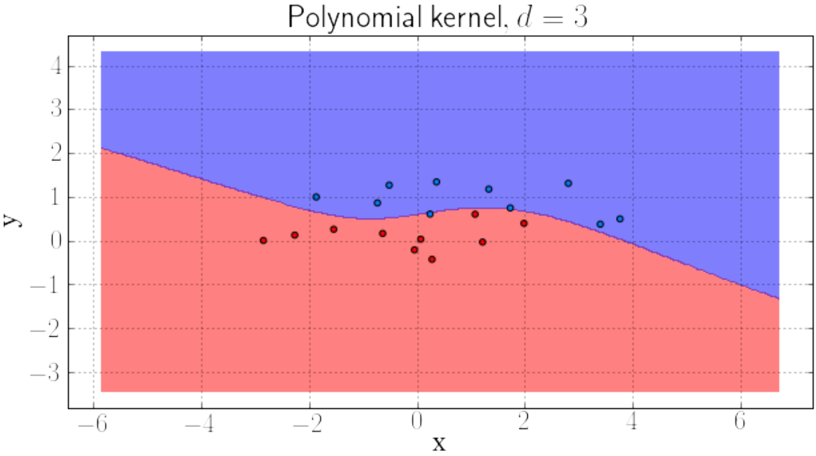
Polynomial kernel - variable d



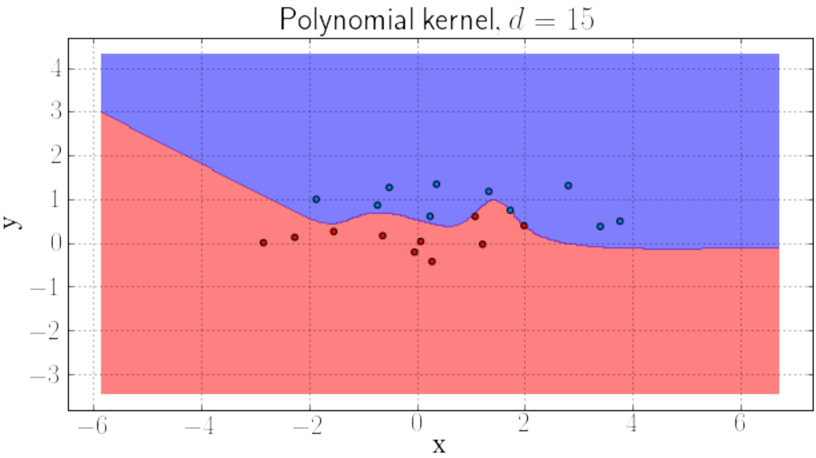
Polynomial kernel - variable d



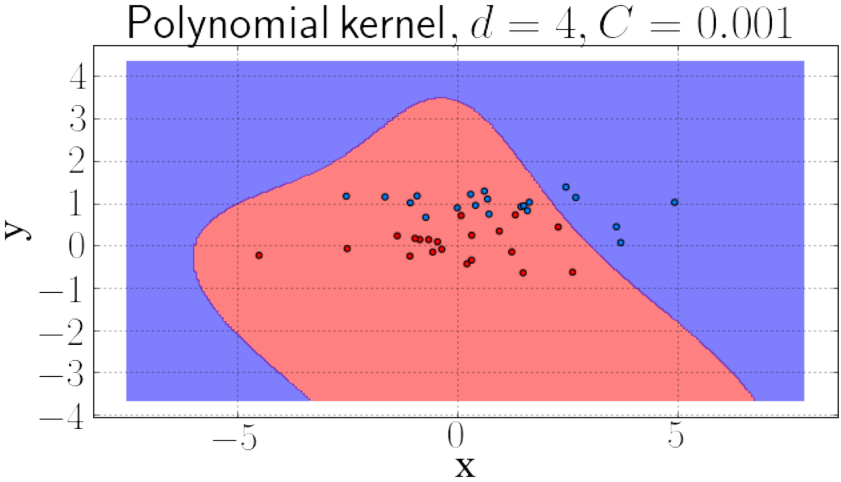
Polynomial kernel - variable d



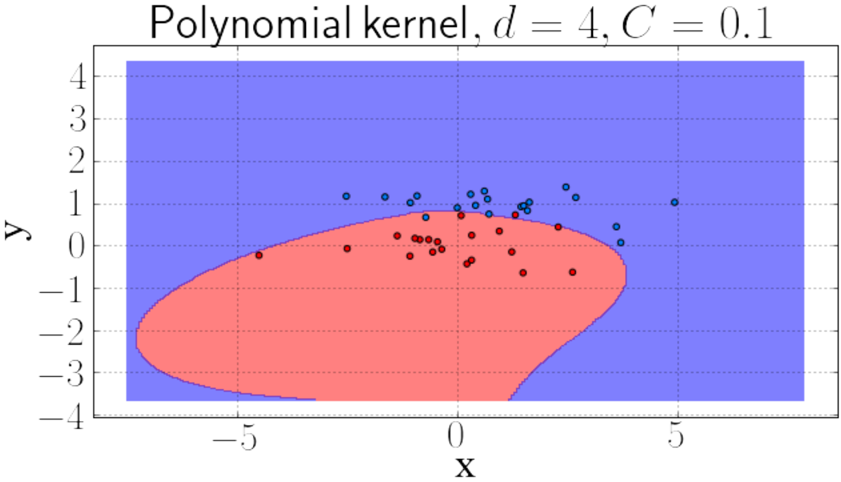
Polynomial kernel - variable d



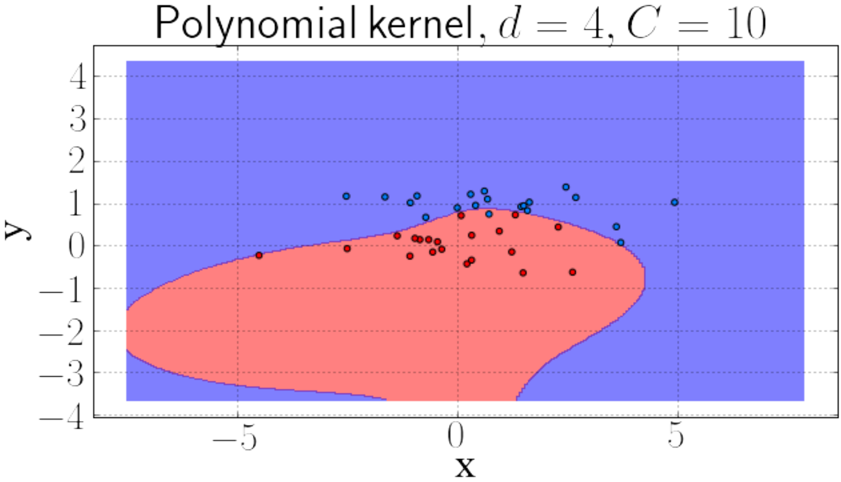
Polynomial kernel - variable C



Polynomial kernel - variable C



Polynomial kernel - variable C



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 prodict (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.