Linear classifiers. Kernels

V. Kitov

Yandex School of Data Analysis

Imperial College London
Department of Physics

January 2015

Table of Contents

Linear discriminant functions

- Linear discriminant function: $g(x) = w^T x + w_0$,

$$\widehat{\omega} = \begin{cases} \omega_1, & g(x) \geq 0 \\ \omega_2, & g(x) < 0 \end{cases}$$

- If we denote classes ω_1 and ω_2 with $y = +1$ and $y = -1$ respectively, we get the decision rule $y = \operatorname{sign} g(x)$.
- Define new feature $x_0 \equiv 1$, then $g(x) = w^T x = \langle w, x \rangle$ for $w = [w_0, w_1, ..w_D]^T$.
- Define margin $M(x) = g(x)y$
    - $M(x) \geq 0 <=>$ object x is correctly classified
    - $|M(x)|$ measures confidence of decision

Weights selection problem

- Final task - minimize misclassifications count:

$$Q_{accurate}(w|X) = \sum_i \mathbb{I}[M(x_i|w) < 0] \to \min_w$$

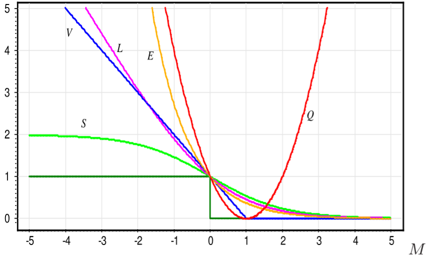
- Standard optimization techniques are impossible, because $Q(w, X)$ is discontinuous.
- Idea: approximate indicator of misclassification with smooth majorizing function $\mathcal{L}$:

$$\mathbb{I}[M(x_i|w) < 0] \leq \mathcal{L}(M(x_i|w))$$

Approximation of target criteria

We obtain approximation of target criteria:

$$Q_{accurate}(w|X) = \sum_i \mathbb{I}[M(x_i|w) < 0]$$
$$\leq \sum_i \mathcal{L}(M(x_i|w)) = Q_{approx}(w|X)$$



$$Q(M) = (1-M)^2$$
$$V(M) = (1-M)_+$$
$$S(M) = 2(1+e^M)^{-1}$$
$$L(M) = \log_2(1+e^{-M})$$
$$E(M) = e^{-M}$$

Optimization

- Optimization task to get weights:

$$Q_{approx}(w|X) = \sum_{i=1}^{n} \mathcal{L}(M(x_i|w)) = \sum_{i=1}^{n} \mathcal{L}(\langle w, x_i \rangle y_i) \to \min_{w}$$

- Gradient descent algorithm:
    - Iteratively until convergence

$$w \leftarrow w - \eta \frac{\partial Q_{approx}(w|X)}{\partial w} = w - \eta \sum_{i=1}^{n} \mathcal{L}'(\langle w, x_i \rangle y_i) x_i y_i$$

    - η - parameter, controlling the speed of convergence.
- Faster convergence when updates are more often - e.g. at each observation. Observations may be taken randomly.

Improved optimization

Stochastic gradient descent algorithm

Calculate $\widehat{Q}_{approx}(w, X) = \sum_{i=1}^{n} \mathcal{L}(M(x_i|w))$

Iteratively, until convergence of $\widehat{Q}_{approx}$ or convergence of w:

1. select random observation (x_i, y_i)
2. adapt weights: $w \leftarrow w - \eta\mathcal{L}'(\langle w, x_i\rangle y_i)x_i y_i$
3. Estimate error: $\varepsilon_i = \mathcal{L}(\langle w, x_i\rangle y_i)$
4. Recalculate $\widehat{Q}_{approx} = (1-\alpha)\widehat{Q}_{approx} + \alpha\varepsilon_i$

Initial weights selection:

- all zeros
- random at $[-\frac{1}{2D}, \frac{1}{2D}]$ (for logistic approximation) or arbitrary random
- $w_i = \frac{\langle x^i, y\rangle}{\langle x^i, x^i\rangle}$

Analysis

Advantages

- Easy to implement
- Works in online environments
- Small random subset of objects may be enough for accurate complete estimation

Analysis

Advantages

- Easy to implement
- Works in online environments
- Small random subset of objects may be enough for accurate complete estimation

Disadvantages

- May converge to local optima
- For improper choice of parameters
    - may diverge
    - may converge too slowly
- for large D and small n may overtrain
- when $\mathcal{L}(u)$ has horizontal asymptotes, algorithm may get stuck for large values of $\langle w, x_i \rangle$

Examples

Delta-rule $\mathcal{L}(u) = (u-1)^2$

$$w \leftarrow w - \eta(\langle w, x_i \rangle - y_i)x_i$$

This also fits for regression $y \in \mathbb{R}$, $a(x) = \langle w, x \rangle$ and cost function $(\langle w, x \rangle - y)^2$

Perceptron of Rosenblatt $\mathcal{L}(u) = [-u]_+$

$$w \leftarrow w + \begin{cases} 0, & \langle w, x_i \rangle y_i \geq 0 \\ \eta x_i y_i & \langle w, x_i \rangle y_i < 0 \end{cases}$$

Recommendations for usage

- Faster converges for scaled features
    - normalization equalizes long narrow valley structures
    - $\langle w, x_i \rangle y_i$ becomes limited at early iterations - SGD does not "get stuck" for $\mathcal{L}$ with horizontal asymptotes.
- Faster convergence when more errors are made:
    - random sampling with probabilities proportional to $\varepsilon_i = \mathcal{L}(\langle w, x_i \rangle y_i)$
    - random sampling with most diverse objects (e.g. sampling repeatedly from different classes)
- Faster calculation: make change to w only for mistakes large enough if $\varepsilon_i \geq \delta$, for some threshold $\delta > 0$.
- Find global minimum by starting the procedure from different starting points

Selection of η

- Larger η => algorithm more prone to diverge.
- Plot $Q_{approx}(w)$ (or $\widehat{Q}_{approx}(w)$) versus iteration number t to control convergence.
- Deterministic scheme:
  - Stochastic gradient descent converges to local optima if
    - $\eta_t \to 0$
    - $\sum_{t=1}^{\infty} \eta_t = \infty$
    - $\sum_{t=1}^{\infty} \eta_t^2 < \infty$
  - Example: $\eta_t = \frac{1}{t}$
- Data dependent scheme:
  - At each step find $\eta_t = \arg\min_{\eta} Q_{approx}(w - \eta \frac{\partial Q_{approx}}{\partial w})$
  - Often analytical solution for such η exists

Overtraining

- Early stopping:
  - control algorithm performance on separate validation set.
  - when performance start to increase - stop.
- Regularization
  - Add penalty for large weights:

$$Q_{approx}^{regularized}(w) = Q_{approx}(w) + \frac{\tau}{2}|w|^2$$

  - Gradient descent step becomes: $w \leftarrow w(1-\eta\tau) - \eta Q'_{approx}(w)$
  - Weights get exponential decay at each step
  - τ controls the trade-off between bias and variance
    - it prevents overfitting,
    - prevents non-stable estimates of w for correlated features
    - prevents SGD getting stuck for large weights and small $\mathcal{L}'(u)$
    - limits flexibility of the model

Table of Contents

Regularization

- Useful technique to control the trade-off between bias and variance, can be applied to any algorithm.

$$Q^{regularized}(w) = Q(w) + \tau ||w||_2$$

$$Q^{regularized}(w) = Q(w) + \tau ||w||_1$$

$$||w||_1 = \sum_{d=1}^{D} |w^d|, \quad ||w||_2 = \sqrt{\sum_{d=1}^{D} (w^d)^2}$$

- Examples:
  - LASSO: least-squares regression, using $||w||_1$
  - Ridge: least-squares regression, using $||w||_2$
  - Elastic Net: : least-squares regression, using both

Maximum probability estimation

- $X = \{x_1, x_2, ... x_n\}$, $Y = \{y_1, y_2, ... y_n\}$ - training sample of i.i.d. observations, $(x_i, y_i) \sim p(y|x, w)$
- ML estimation $\widehat{w} = \arg\max_w p(Y|X, w)$
- Using independence assumption:

$$\prod_{i=1}^{n} p(y_i|x_i, w) = \sum_{i=1}^{n} \ln p(y_i|x_i, w) \to \max_w$$

- Approximated misclassification:

$$\sum_{i=1}^{n} \mathcal{L}(g(x_i)y_i|w) \to \min_w$$

- Interrelation:

$$\mathcal{L}(g(x_i)y_i|w) = -\ln p(y_i|x_i, w)$$

Maximum a prosteriori estimation

- $X = \{x_1, x_2, ...x_n\}$, $Y = \{y_1, y_2, ...y_n\}$ - training sample of i.i.d. observations, $(x_i, y_i) \sim p(x, y|w)$
- $x_i \sim p(x|w)$
- MAP estimation:
  - w is random with prior probability $p(w)$

$$p(w|X, Y) = \frac{p(X, Y, w)}{p(X, Y)} = \frac{p(X, Y|w)p(w)}{p(X, Y)} \propto p(X, Y|w)p(w)$$

$$w = \arg\max_w p(w|X, Y) = \arg\max_w p(X, Y|w)p(w)$$

$$\sum_{i=1}^{n} \ln p(x_i, y_i|\theta) + \ln p(w) \to \max_w$$

Gaussian prior

- Gaussian prior

$$\ln p(w, \sigma^2) = \ln \left(\frac{1}{(2\pi\sigma^2)^{n/2}} e^{-\frac{||w||_2^2}{2\sigma^2}} \right) = -\frac{1}{2\sigma^2}||w||_2^2 + \mathrm{const}(w)$$

- Laplace prior

$$\ln p(w, C) = \ln \left(\frac{1}{(2C)^n} e^{-\frac{||w||_1}{C}} \right) = -\frac{1}{C}||w||_1 + \mathrm{const}(w)$$

L_1 norm

- $||w||_1$ regularizer will do feature selection.
- Consider

$$Q(w) = \sum_{i=1}^{n} \mathcal{L}_i(w) + \frac{1}{C} \sum_{d=1}^{D} |w_d|$$

- if $\frac{1}{C} > \sup_w \left| \frac{\partial \mathcal{L}(w)}{\partial w_i} \right|$, then it becomes optimal to set $w_i = 0$
- For smaller C more inequalities will become active.

Adaptive feature importances

- Suppose
  - weights have prior Gaussian distribution
  - are uncorrelated
  - each weight w_d has individual prior variance $\sigma_d^2 = C_d$
- Prior distribution becomes:

$$p(w) = \frac{1}{(2\pi)^{n/2}\sqrt{C_1 \ldots C_D}} e^{-\sum_{d=1}^{D} \frac{w_d^2}{2C_d}}$$

- Target functional becomes:

$$Q_{approx}(w) = \sum_{i=1}^{n} \mathcal{L}_i(w) + \frac{1}{2}\sum_{d=1}^{D}\left(\ln C_d + \frac{w_d^2}{C_d}\right) \to \min_{w,C}$$

- If $\widehat{C}_d \to 0$, feature d is removed.

Table of Contents

Logistic regression

Assume (γ_1, γ_2 are the costs of misclassifying classes ω_1 and ω_2):

$$\ln\left(\frac{\gamma_1 p(\omega_1|x)}{\gamma_2 p(\omega_2|x)}\right) = \beta_0 + \boldsymbol{\beta}^T \mathbf{x}$$

It is equivalent to

$$\begin{aligned} p(\omega_2|x) &= \frac{1}{1 + exp(\beta_0' + \boldsymbol{\beta}^T \mathbf{x})} \\ p(\omega_1|x) &= \frac{exp(\beta_0' + \boldsymbol{\beta}^T \mathbf{x})}{1 + exp(\beta_0' + \boldsymbol{\beta}^T \mathbf{x})} \end{aligned}$$

where $\beta_0' = \beta_0 - \ln(\gamma_1/\gamma_2)$

Logistic regression

Decision rule (following Bayes minimum risk principle):

$$x = \begin{cases} \omega_1, & \beta_0' + \boldsymbol{\beta}^T \mathbf{x} > 0 \\ \omega_2, & \beta_0' + \boldsymbol{\beta}^T \mathbf{x} < 0 \end{cases}$$

Estimate with ML:

$$\prod_{i=1}^{n} p(c_i | x_i) \rightarrow \max_{\beta_0', \boldsymbol{\beta}}$$

where c_i is the class of x_i.

Multiclass logistic regression

- Assumption:

$$\ln\left(\frac{\gamma_s p(\omega_s|x)}{\gamma_C p(\omega_C|x)}\right) = \beta_{s0} + \boldsymbol{\beta}_s^T \boldsymbol{x}, \quad s = 1, 2, \dots C-1$$

- Posterior class probabilities:

$$\begin{aligned} p(\omega_s|x) &= \frac{exp(\beta'_{s0} + \beta_s^T x)}{1 + \sum_{s=1}^{C-1} exp(\beta'_{s0} + \beta_s^T x)}, \quad s = 1, 2, \dots C-1 \\ p(\omega_C|x) &= \frac{1}{1 + \sum_{s=1}^{C-1} exp(\beta'_{s0} + \beta_s^T x)} \end{aligned}$$

$$\beta'_{s0} = \beta_{s0} - \ln\left(\gamma_s/\gamma_C\right)$$

Multiclass logistic regression

- Decision rule (following Bayes minimum risk principle): assign x to class $c = \arg\max_c \beta_{c0} + \beta_c^T x$ if $\beta_{c0} + \beta_c^T x > 0$ otherwise assign x to class C.
- Estimate with ML:

$$\prod_{i=1}^{n} p(c_i|x_i) \to \max_{\beta_0', \boldsymbol{\beta}}$$

where c_i is the class of x_i.

- Please pay attention to the difference between β_0 and β_0'.

Logistic regression - loss function

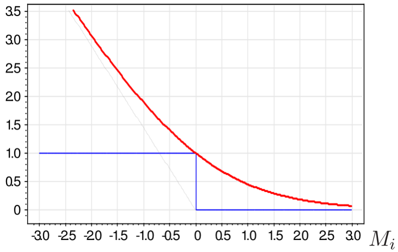
For two class situation $p(y|x) = \sigma(\langle w, x\rangle y)$ for $\sigma = \frac{1}{1+e^{-z}}$, $w = [\beta_0', \beta]$, $x = [1, x_1, x_2, ...x_D]$.

Estimation with ML:

$$\prod_{i=1}^{n} \sigma(\langle w, x_i\rangle y_i) \to \max_w$$

which is equivalent to

$$\sum_i^n \ln(1 + e^{-\langle w, x_i\rangle y_i}) \to \min_w$$



It follows that logistic regression is linear discriminant estimated with loss function $\mathcal{L}(M) = \ln(1 + e^{-M})$.

SGD realization of logistic regression

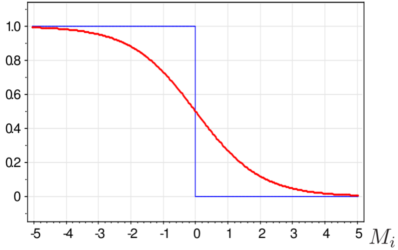
Substituting $\mathcal{L}(M) = \ln(1 + e^{-M})$ into update rule, we obtain that for each sample (x_i, y_i) weights should be adapted according to

$$w \leftarrow w + \eta\sigma(-M_i)x_i y_i$$

Perceptron of Rosenblatt update rule:

$$w \leftarrow w + \eta\mathbb{I}[M_i < 0]x_i y_i$$

- Logistic rule update is the smoothed variant of perceptron's update.
- The more severe the error (according to margin) - the more weights are adapted.



Logistic regression - assumptions

In logistic regression it was assumed that (for equal misclassification costs):

$$\ln\left(\frac{p(\omega_1|x)}{1-p(\omega_1|x)}\right) = \beta_0 + \boldsymbol{\beta}^T\boldsymbol{x}$$

which is equivalent to

$$p(\omega_1|x) \quad = \quad \frac{exp(\beta_0 + \boldsymbol{\beta}^T\boldsymbol{x})}{1 + exp(\beta_0 + \boldsymbol{\beta}^T\boldsymbol{x})}$$

Decision rule (following Bayes minimum risk principle):

$$x = \begin{cases} \omega_1, & \beta_0{+}\boldsymbol{\beta}^T\boldsymbol{x}{>}0 \\ \omega_2, & \beta_0 + \boldsymbol{\beta}^T\boldsymbol{x} < 0 \end{cases}$$

What assumption allowed to obtain probabilities?

Logistic regression - assumptions

In logistic regression it was assumed that (for equal misclassification costs):

$$F\left(p(\omega_1|x)\right) = \beta_0 + \boldsymbol{\beta}^T \boldsymbol{x}$$

Any $F(z)$ satisfying:

- $F(z)$ is increasing
- $\mathrm{Dom}[F] = (0, 1)$
- $\mathrm{Im}[F] = \mathbb{R}$
- $F(1/2) = 0$

leads to the same decision rule:

$$x = \begin{cases} \omega_1, & \beta_0 + \boldsymbol{\beta}^T \boldsymbol{x} > 0 \\ \omega_2, & \beta_0 + \boldsymbol{\beta}^T \boldsymbol{x} < 0 \end{cases}$$

Logistic regression - assumptions

This is equivalent to

$$p(\omega_1|x) = G(\beta_0 + \boldsymbol{\beta}^T \mathbf{x})$$

for any $G = F^{-1}$, satisfying:

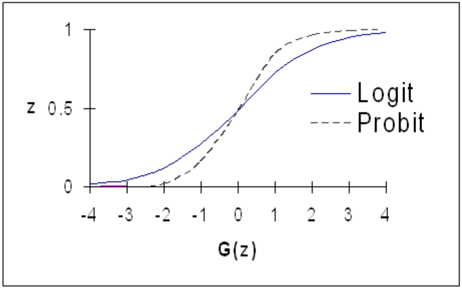
- $G(z)$ is increasing
- $\mathrm{Dom}[G] = \mathbb{R}$
- $\mathrm{Im}[G] = (0, 1)$
- $G(0) = 1/2$

leads to the same decision rule:

$$x = \begin{cases} \omega_1, & \beta_0 + \boldsymbol{\beta}^T \mathbf{x} > 0 \\ \omega_2, & \beta_0 + \boldsymbol{\beta}^T \mathbf{x} < 0 \end{cases}$$

Probit/Logit

- $G(z)$ may be distribution function of any continuous symmetrical random variable, taking values on $\mathbb{R}$.
- Examples:
  - $G(z) = \frac{e^z}{1+e^z}$ - logit (leads to logistic regression)
  - $G(z) = \Phi(z)$ - normal c.d.f., probit.



Analysis of logistic regression

Advantages

- Implements margin offset strategy (using smoothed weights adaptation in SGD)
- Gives estimates of class probabilities

Analysis of logistic regression

Advantages

- Implements margin offset strategy (using smoothed weights adaptation in SGD)
- Gives estimates of class probabilities

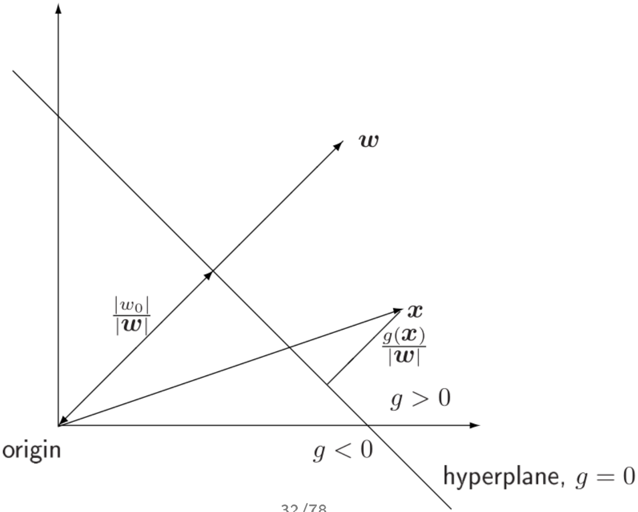
Disadvantages

- Quality deteriorates if model assumptions are violated
- Disadvantages inherited from SGD:
  - need to normalize features
  - filter outliers
  - regularization for multiple or correlated features

Table of Contents

Reminder

For linear discriminant function $g(x) = w^T x + w_0$:



Support vector machines

(a) (b)

Support vector machines

(a) (b)

Main idea

Select hyperplane maximizing the margin - the sum of distances from nearest ω_1 object to hyperplane and from nearest ω_2 object to hyperplane.

Support vector machines

Objects x_i for $i = 1, 2, ...n$ lie at distance $b/|w|$ from discriminant hyperplane if

$$\begin{cases} x_i^T w + w_0 \geq b, & y_i = +1 \\ x_i^T w + w_0 \leq b & y_i = -1 \end{cases} \quad i = 1, 2, ...n.$$

This can be rewritten as

$$y_i(x_i^T w + w_0) \geq b, \quad i = 1, 2, ...n.$$

The margin is equal to $2b/|w|$. Since w, w_0 and b are defined up to multiplication constant, we can set $b = 1$.

Problem statement

Problem statement:

$$\begin{cases} w^T w \to \min\limits_{w,w_0} \\ y_i(x_i^T w + w_0) \geq 1, \quad i = 1, 2, ...n. \end{cases}$$

According to Karush-Kuhn-Takker theorem, solution satisfies the following problem:

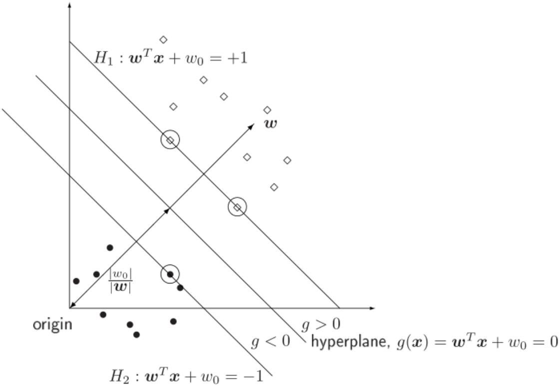
$$L_P = \frac{1}{2} w^T w - \sum_{i=1}^{n} \alpha_i (y_i (w^T x + w_0) - 1) \to \min_{w,w_0} \max_{\alpha}, \quad \alpha_i \geq 0, \ i = 1, 2, ...$$

with the constraints:

$$\begin{cases} \alpha_i \geq 0, \\ y_i(x_i^T w + w_0) - 1 \geq 0, \\ \alpha_i(y_i(x_i^T w + w_0) - 1) = 0. \end{cases}$$

Support vectors

Condition $\alpha_i(y_i(x_i^T w + w_0) - 1) = 0$ is satisfied when either $\alpha_i = 0$ or $y_i(x_i^T w + w_0) - 1 = 0$. Second case describes support vectors, which lie at distance $1/|w|$ to separating hyperplane and which affect the weights. Other vectors don't affect the solution.



Dual problem

$$\frac{\partial L}{\partial w_0} = 0 : \sum_{i=1}^{n} \alpha_i y_i = 0$$

$$\frac{\partial L}{\partial w} = 0 : \; w = \sum_{i=1}^{n} \alpha_i y_i x_i$$

Substituting into Lagrangian L_P, we get:

$$L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j \to \max_{\alpha}$$

α_i can be found from the dual optimization problem:

$$\begin{cases} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j \to \max_{\alpha} \\ \alpha_i \geq 0, \; i = 1, 2, \ldots n; \; \sum_{i=1}^{n} \alpha_i y_i = 0 \end{cases}$$

Solution

Denote $\mathcal{SV}$ - the set of indexes of support vectors.
Optimal α_i determine weights directly:

$$w = \sum_{i \in \mathcal{SV}} \alpha_i y_i x_i$$

w_0 can be found from any edge equality for support vectors:

$$y_i(x_i^T w + w_0) = 1, \; i \in \mathcal{SV}$$

Solution from summation over n_{SV} equation provides a more robust estimate of w_0:

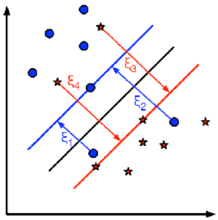
$$n_{\mathcal{SV}} w_0 + \sum_{i \in \mathcal{SV}} x_i^T w = \sum_{i \in \mathcal{SV}} y_i$$

Linearly non-separable case

No separating hyperplane exists. Errors are permitted by including slack variables ξ_i:

$$\begin{cases} \frac{1}{2}w^T w + C\sum_{i=1}^n \xi_i \to \min_{w,\xi} \\ y_i(w^T x_i + w_0) \geq 1 - \xi_i,\ i = 1, 2, \ldots n \\ \xi_i \geq 0,\ i = 1, 2, \ldots n \end{cases}$$

- Parameter C is the cost for misclassification and controls the bias-variance trade-off.
- It is chosen on validation set.
- Other penalties are possible, e.g. $C\sum_i \xi_i^2$.



Linearly non-separable case

According to Karush-Kuhn-Takker theorem, the solution satisfies:

$$L_P = \frac{1}{2}w^T w + C\sum_i \xi_i - \sum_{i=1}^{n} \alpha_i(y_i(w^T x_i + w_0) - 1 + \xi_i) - \sum_{i=1}^{n} r_i \xi_i$$

$$L_P \to \min_{w, w_0, \xi} \max_{\alpha, r}$$

under constraints:

$$\begin{cases} \xi_i \geq 0,\ \alpha_i \geq 0,\ r_i \geq 0 \\ y_i(x_i^T w + w_0) \geq 1 - \xi_i, \\ \alpha_i(y_i(w^T x_i + w_0) - 1 + \xi_i) = 0 \\ r_i \xi_i = 0 \end{cases}$$

$$\frac{\partial L_P}{\partial \xi_i} = 0 :\ C - \alpha_i - r_i = 0 \quad \Rightarrow \quad \alpha_i \in [0, C].$$

Classification of training objects

- **Non-informative objects**:
  - have $\alpha_i = 0$ ($\Leftrightarrow r_i = C \Leftrightarrow \xi_i = 0 \Leftrightarrow y_i(w^T x_i + w_0) \geq 1$)
- **Support vectors:**
  - have $\alpha_i > 0$ ($\Leftrightarrow y_i(w^T x_i + w_0) = 1 - \xi_i$)
  - **boundary support vectors**:
    - have $\xi_i = 0$ ($\Leftrightarrow r_i > 0 \Leftrightarrow \alpha_i \in (0, C) \Leftrightarrow y(w^T x_i + w_0) = 1$) then support vector lies at $1/|w|$ distance to separating hyperplane and is called boundary support vector.
  - **violating support vectors**:
    - have $\xi_i > 0$ ($\Leftrightarrow r_i = 0 \Leftrightarrow \alpha_i = C$), so lies closer than $1/|w|$ to separating hyperplane.
    - If $\xi_i \in (0, 1)$ then violating support vector is correctly classified.
    - If $\xi_i > 1$ then violating support vector is misclassified.

Linearly non-separable case - dual problem

$$\frac{\partial L_P}{\partial w_0} = 0 : \sum_{i=1}^{n} \alpha_i y_i = 0$$

$$\frac{\partial L_P}{\partial w} = 0 : w = \sum_{i=1}^{n} \alpha_i y_i x_i$$

$$\frac{\partial L_P}{\partial \xi_i} = 0 : C - \alpha_i - r_i = 0$$

Substituting these constraints into L_P, we obtain the dual problem:

$$\begin{cases} L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j \to \max_\alpha \\ \sum_{i=1}^{n} \alpha_i y_i = 0 \\ 0 \leq \alpha_i \leq C \end{cases}$$

Solution

Denote $\mathcal{SV}$ - the set of indexes of support vectors with $\alpha_i > 0$ ($\Leftrightarrow y(w^T x_i + w_0) = 1 - \xi_i$) and $\widetilde{\mathcal{SV}}$ - the set of indexes of support vectors with $\alpha_i \in (0, C)$ ($\Leftrightarrow \xi_i = 0$, $y(w^T x_i + w_0) = 1$)
Optimal α_i determine weights directly:

$$w = \sum_{i \in \mathcal{SV}} \alpha_i y_i x_i$$

w_0 can be found from any edge equality for support vectors, having $\xi_i = 0$:

$$y_i(x_i^T w + w_0) = 1,\ i \in \widetilde{\mathcal{SV}}$$

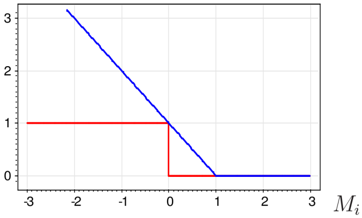
Solution from summation of equations for each $i \in \widetilde{SV}$ provides a more robust estimate of w_0:

$$n_{\widetilde{\mathcal{SV}}} w_0 + \sum_{i \in \widetilde{\mathcal{SV}}} x_i^T w = \sum_{i \in \widetilde{\mathcal{SV}}} y_i$$

Another view on SVM

Optimization problem:

$$\begin{cases} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi_i \to \min_{w,\xi} \\ y_i(w^T x_i + w_0) = M_i(w, w_0) \geq 1 - \xi_i, \\ \xi_i \geq 0, \; i = 1, 2, \dots n \end{cases}$$



can be rewritten as

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

Thus SVM is linear discriminant function with cost approximated with $\mathcal{L}(M) = [1 - M]_+$ and L_2 regularization.

Probabilistic interpretation

SVM optimization task may be obtained if

$$p(x_i, y_i|w, w_0) \sim z_1 e^{-[1-M_i(w,w_0)]_+}$$

with Gaussian prior probability

$$p(w|C) = z_2 e^{-|w|^2/(2C)}$$

Properties

Solution:

$$y = \text{sign}\left\{\sum_{i \in \mathcal{SV}} \alpha_i y_i < x_i, x > + w_0\right\}$$

Sparsity of SVM: solution depends only on support vectors:

- more affected by outliers
- possible filtering scheme (like editing):
  - solve
  - remove lowest margin objects
  - solve on refined sample
- Relevant vectors machine: filter support vectors by regularization of $\alpha \sim \frac{1}{(2\pi)^{n/2}\sqrt{C_1 C_2 \ldots C_n}} e^{\left(-\sum_{i=1}^{n} \frac{\alpha_i^2}{2C_i}\right)}$
- if only a small fraction of objects are incorrectly classified, they may be removed from the training sample and it becomes separable => no need to select C.

Multiclass classification

C classes $\omega_1, \omega_2, ...\omega_C$.

- One-against-all:
  - build C binary classifiers, classifying class ω_i against other classes
  - select the class with highest margin
- One-against-one:
  - build C(C-1)/2 classifiers, classifying class ω_i against ω_j.
  - select the class having maximum votes
- Multiclass variant of initial algorithm

Multiclass SVM

C discriminant functions are built simultaneously:

$$g_k(x) = (w^k)^T x + w_0^k$$

Linearly separable case:

$$\begin{cases} \sum_{k=1}^{C} (w^k)^T w^k \to \min_w \\ (w^{c(i)})^T x + w_0^{c(i)} - (w^k)^T x - w_0^k \geq 1 \, \forall k \neq c(i), \; i = 1, 2, \ldots n \end{cases}$$

Linearly non-separable case:

$$\begin{cases} \sum_{k=1}^{C} (w^k)^T w^k + C \sum_{i=1}^{n} \xi_i \to \min_w \\ (w^{c(i)})^T x + w_0^{c(i)} - (w^k)^T x - w_0^k \geq 1 - \xi_i \, \forall k \neq c(i), \; i = 1, 2, \ldots n \\ \xi_i \geq 0 \end{cases}$$

Table of Contents

Linear SVM reminder

- Solution for weights:

$$w = \sum_{i \in \mathcal{SV}} \alpha_i y_i x_i$$

Discriminant function

$$g(x) = \sum_{i \in \mathcal{SV}} \alpha_i y_i < x_i, x > + w_0$$

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

Kernel SVM

- x is replaced with $\phi(x)$
- $[x] \to [x,\ x^2,\ x^3]$

Kernel

Function $K(x, y) : X \times X \to \mathbb{R}$ is a kernel function if it may be represented as $K(x, y) = \langle \psi(x), \psi(y) \rangle$ for some mapping $\psi : X \to H$, with scalar product defined on H.

- $< x, y >$ is replaced by $< \phi(x), \phi(y) >= K(x, y)$

Kernel SVM

Discriminant function

$$g(x) = \sum_{i \in \mathcal{SV}} \alpha_i y_i K(x_i, x) + w_0$$

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

Kernel properties

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

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

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

- Example: $K(x, y) = (1 + x^T y)^2 = (1 + x_1 y_1 + x_2 y_2)^2 = 1 + 2x_1 y_1 + 2x_1 y_2 + 2x_1 x_2 y_1 y_2 + x_1^2 y_1^2 + x_2^2 y_2^2 = \phi^T(x)\phi(x)$
- $\phi(x) = (1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2, x_1^2, x_2^2)$

Kernel properties

Kernels can be constructed manually:

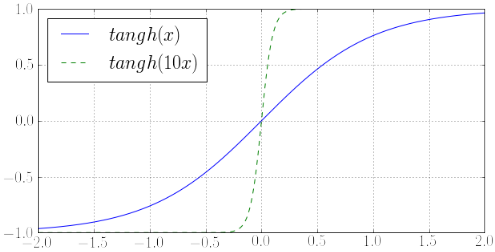
- Scalar product $\langle x, x' \rangle$ is a kernel
- Constant $K(x, x') \equiv 1$ is a kernel
- Product of kernels $K(x, x') = K_1(x, x')K_2(x, x')$ is a kernel.
- For every function $\psi : X \to \mathbb{R}$ the product $K(x, x') = \psi(x)\psi(x')$ is a kernel
- Linear combination of kernels $K(x, x') = \alpha_1 K_1(x, x') + \alpha_2 K(x, x')$ with positive coefficients is a kernel
- Composition of function $\varphi : X \to X$ and kernel K_0 is a kernel: $K(x, x') = K_0(\varphi(x), \varphi(x'))$
- etc.

Useful collection of datasets matching datasets and research papers: https://archive.ics.uci.edu/ml/datasets.html

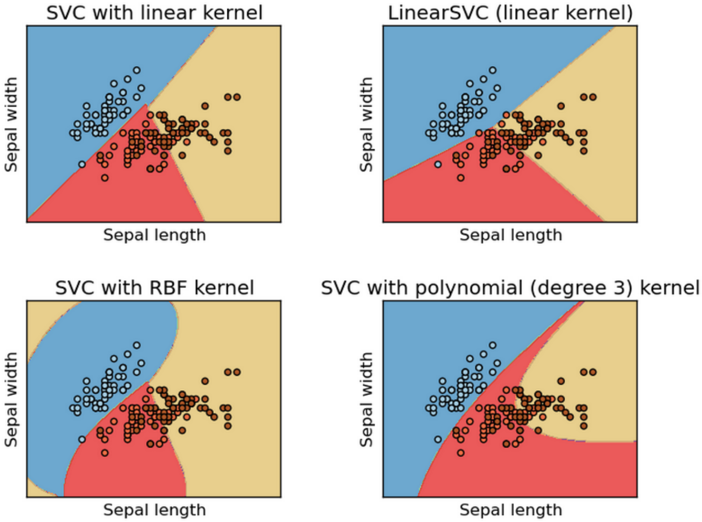
Commonly used kernels

Let x and y be two objects.

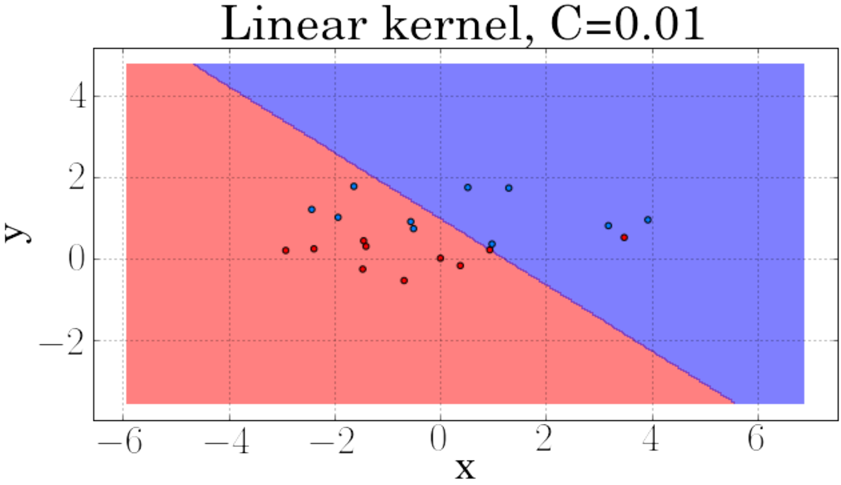
Kernel	Mathematical form
linear	$\langle x, y\rangle$
polynomial	$(\gamma\langle x, y\rangle + r)^d$
RBF	$\exp(-\gamma\lvert x - y\rvert^2)$
sigmoid	$\mathrm{tangh}(\gamma\langle x, y\rangle + r)$



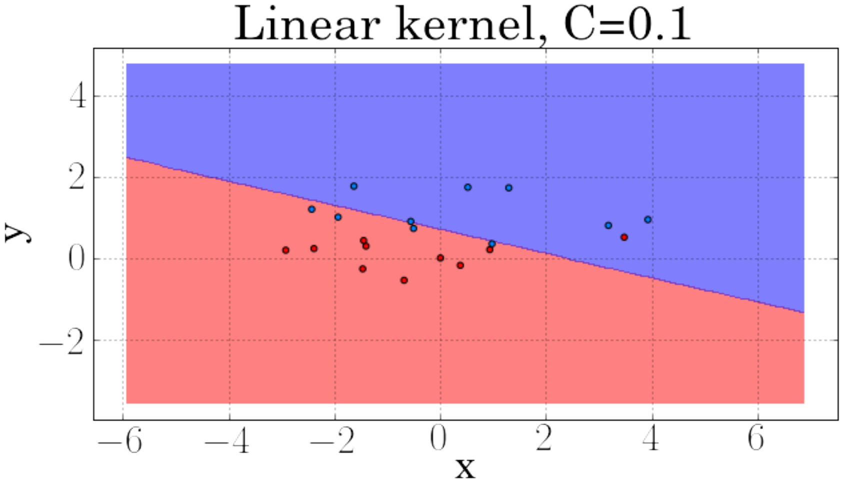
Kernel results



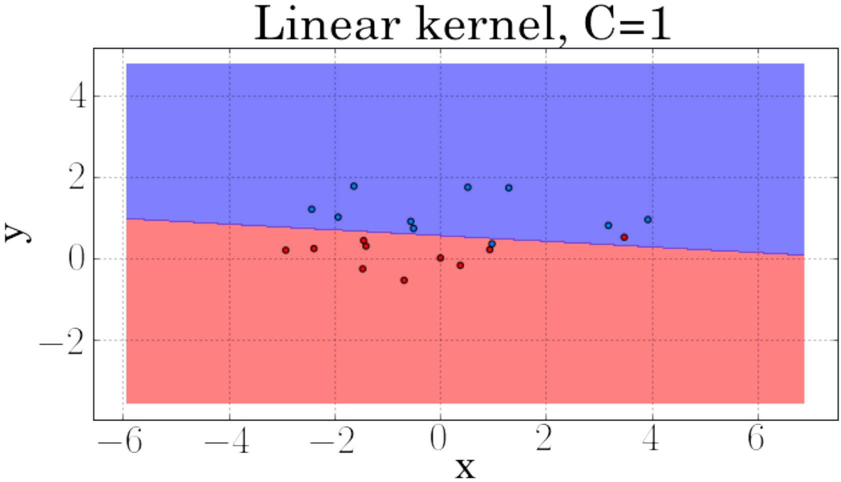
Linear kernel - variable C



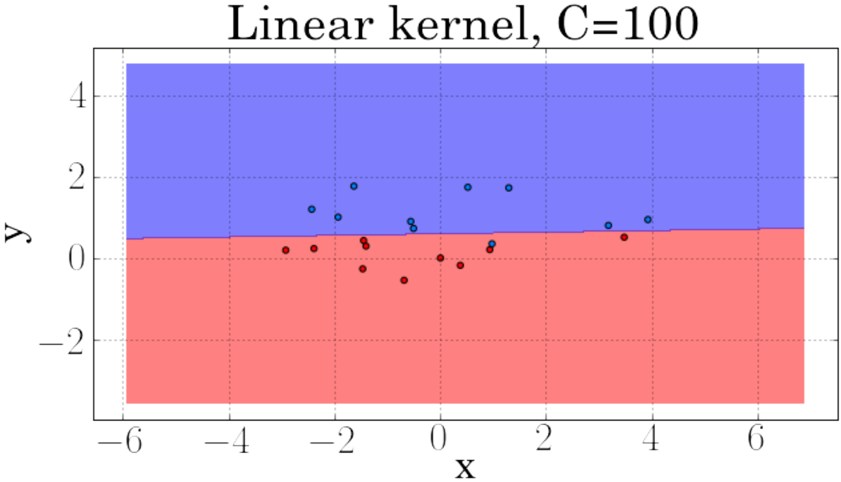
Linear kernel - variable C



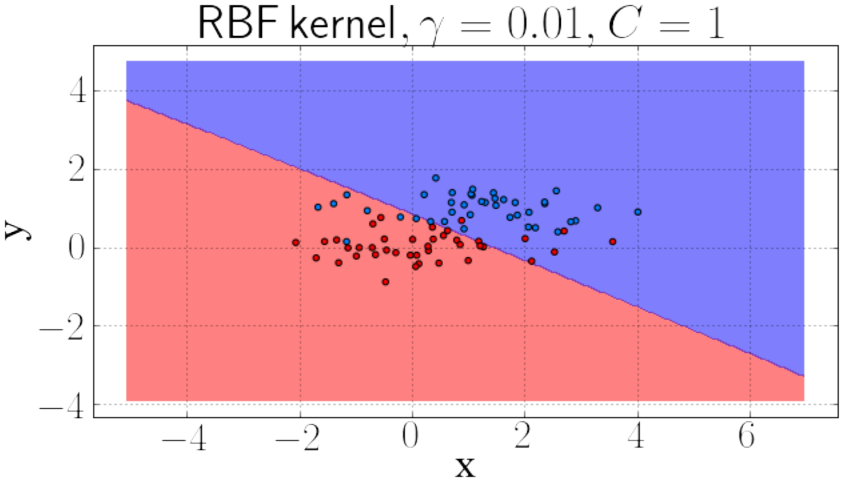
Linear kernel - variable C



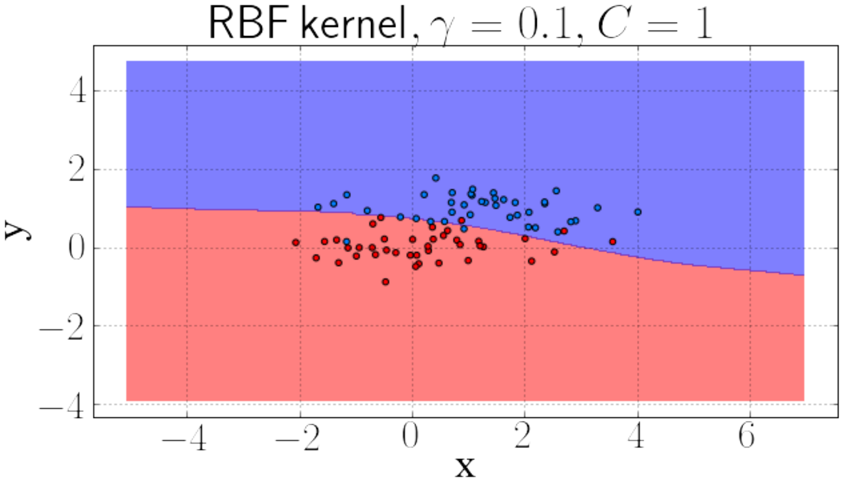
Linear kernel - variable C



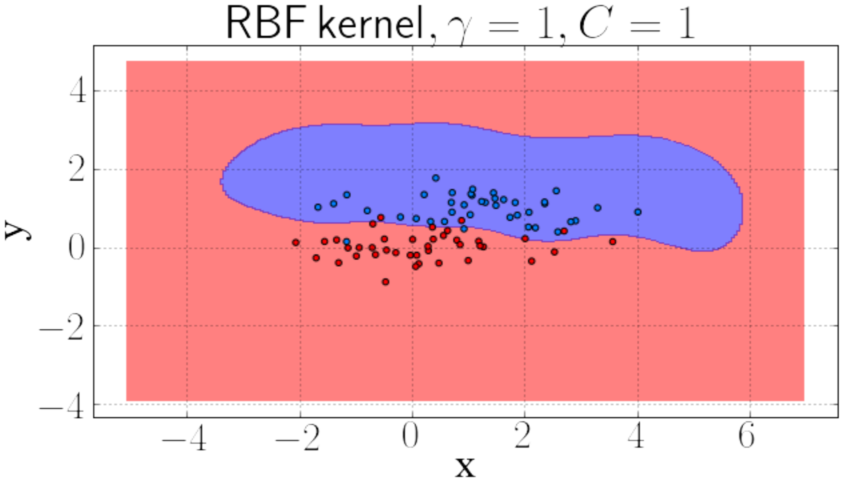
RBF kernel - variable γ



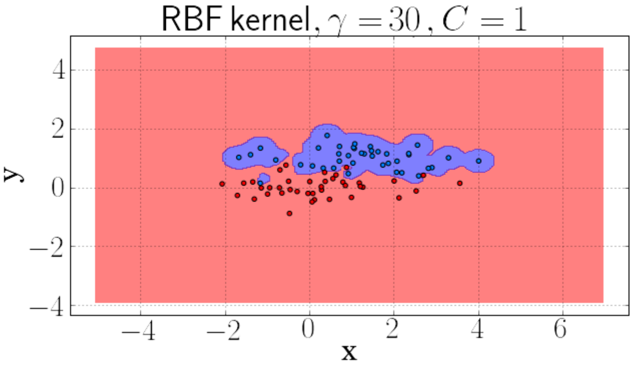
RBF kernel - variable γ



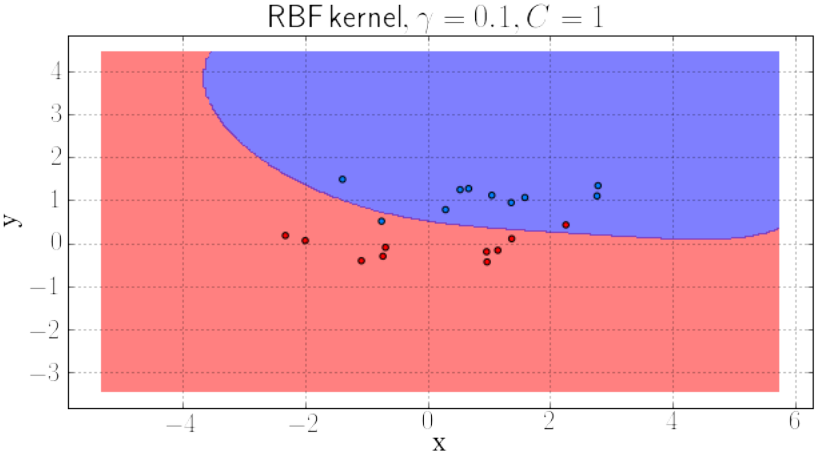
RBF kernel - variable γ



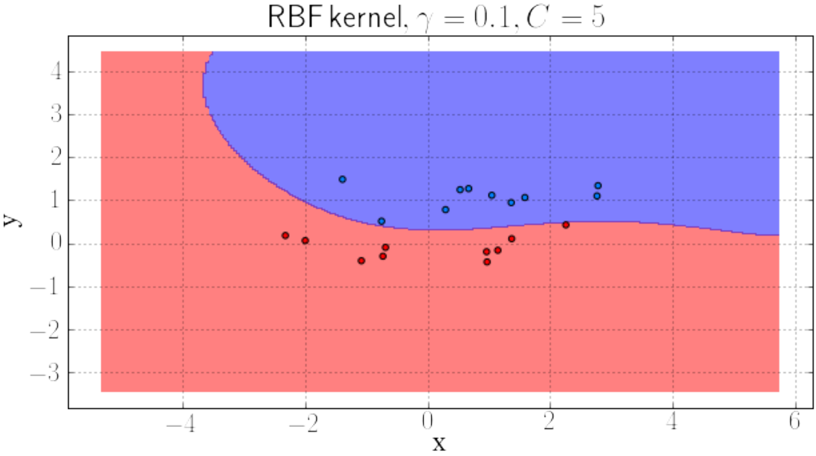
RBF kernel - variable γ



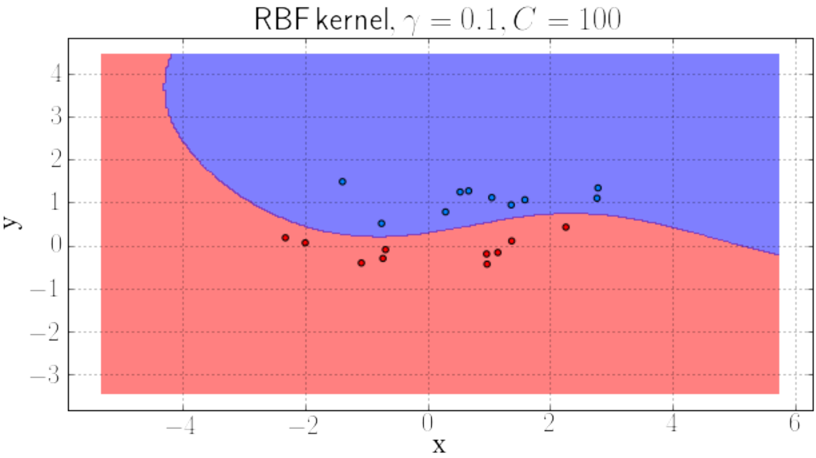
RBF kernel - variable C



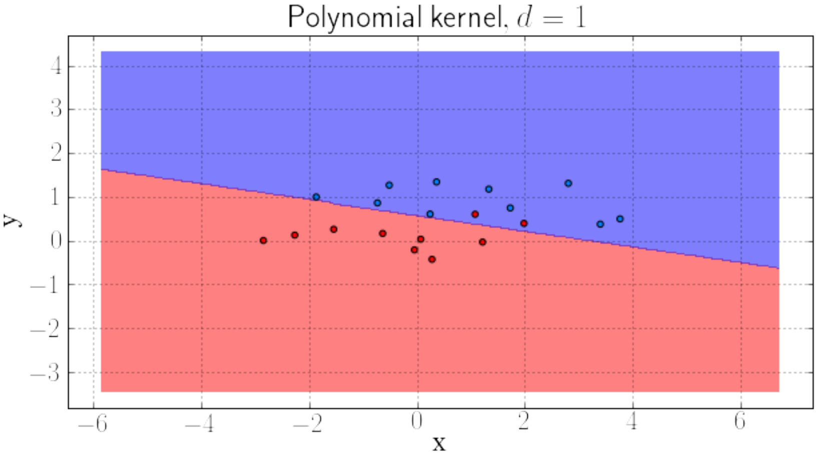
RBF kernel - variable C



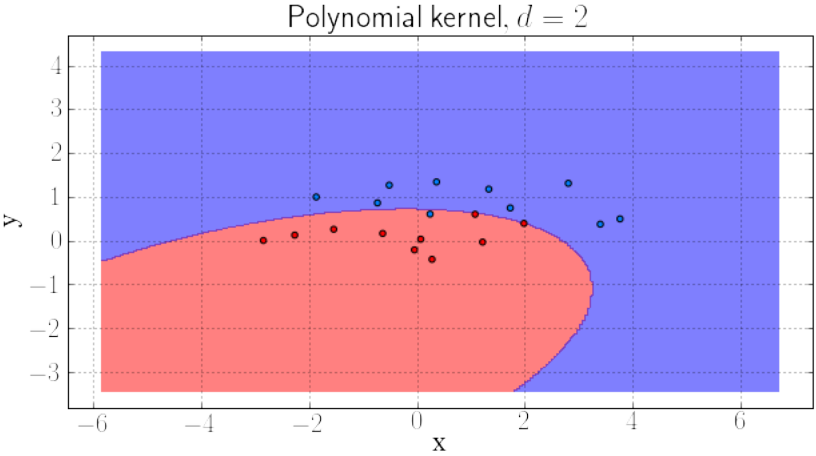
RBF kernel - variable C



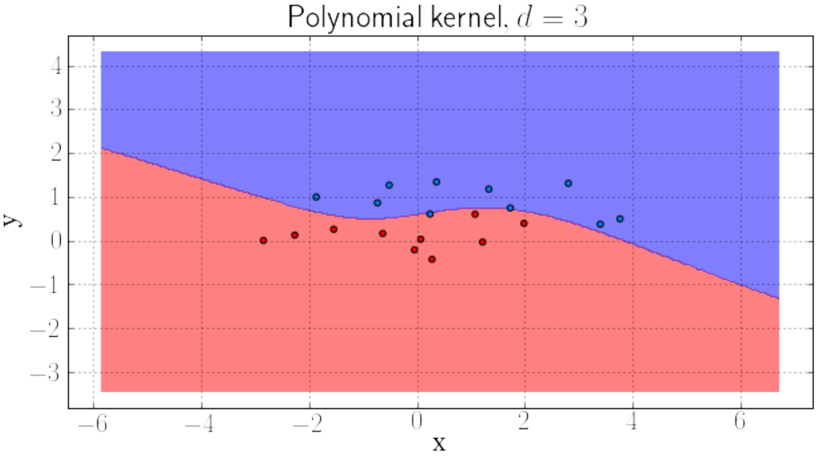
Polynomial kernel - variable d



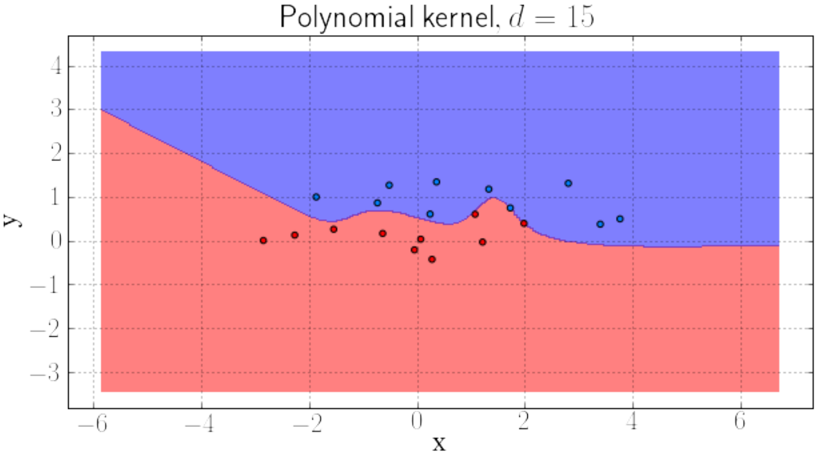
Polynomial kernel - variable d



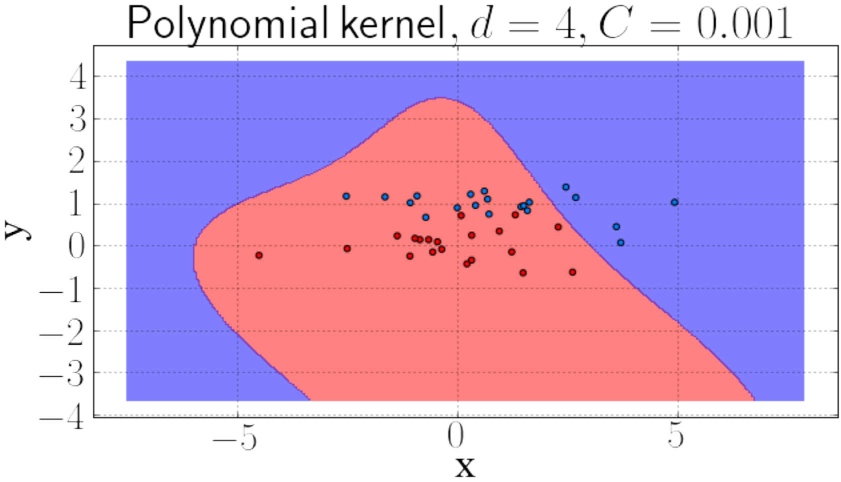
Polynomial kernel - variable d



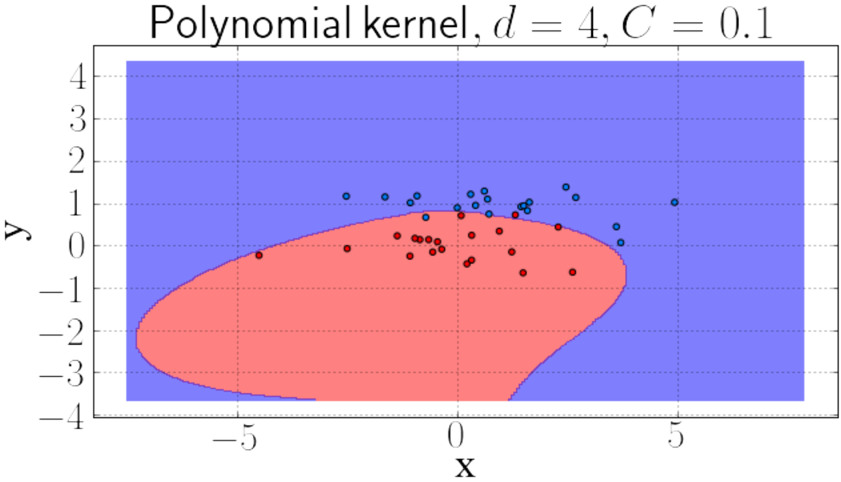
Polynomial kernel - variable d



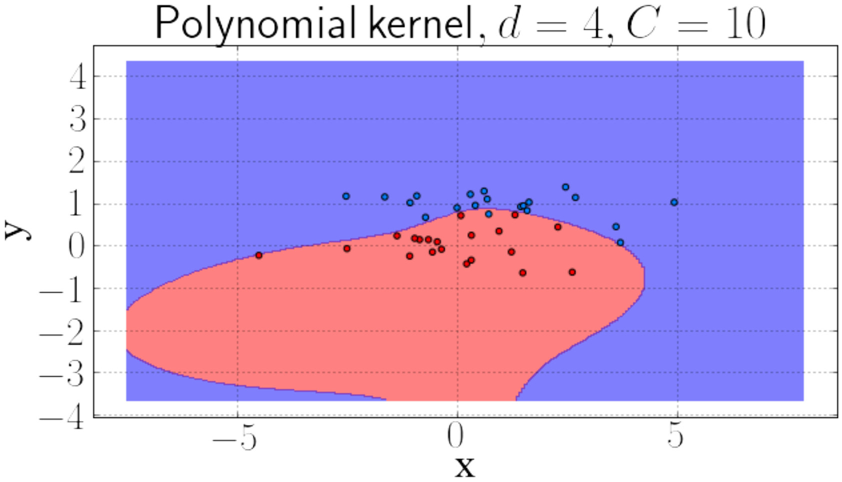
Polynomial kernel - variable C



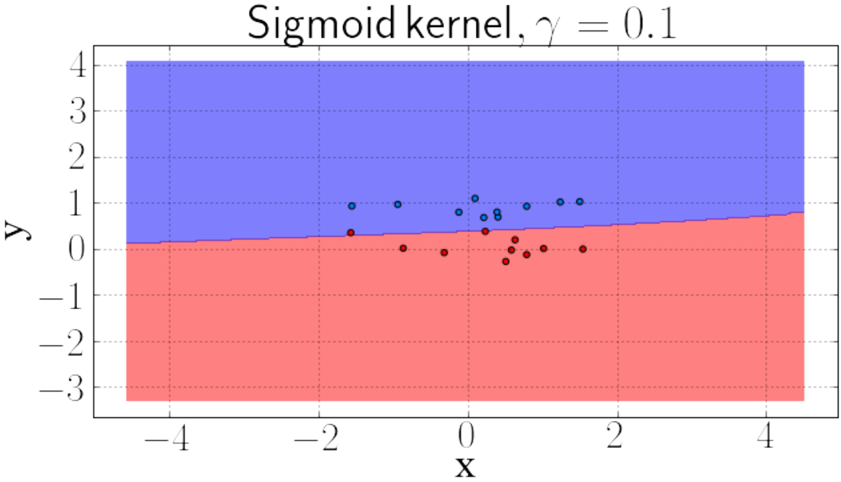
Polynomial kernel - variable C



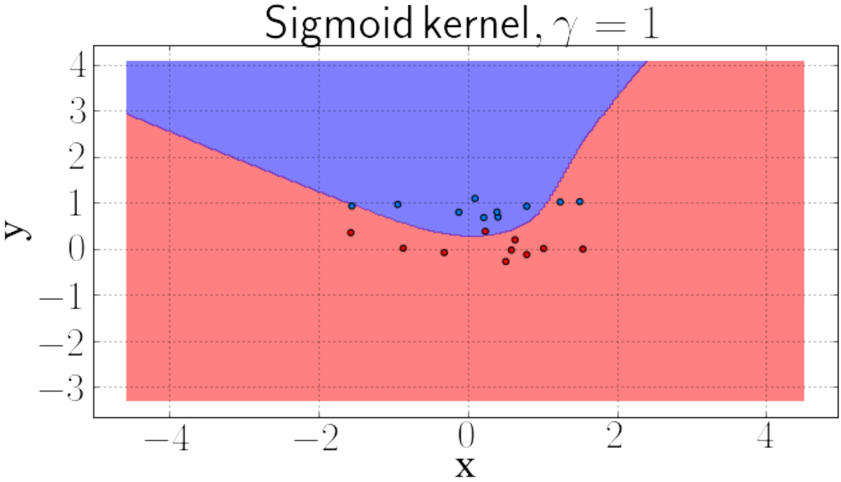
Polynomial kernel - variable C



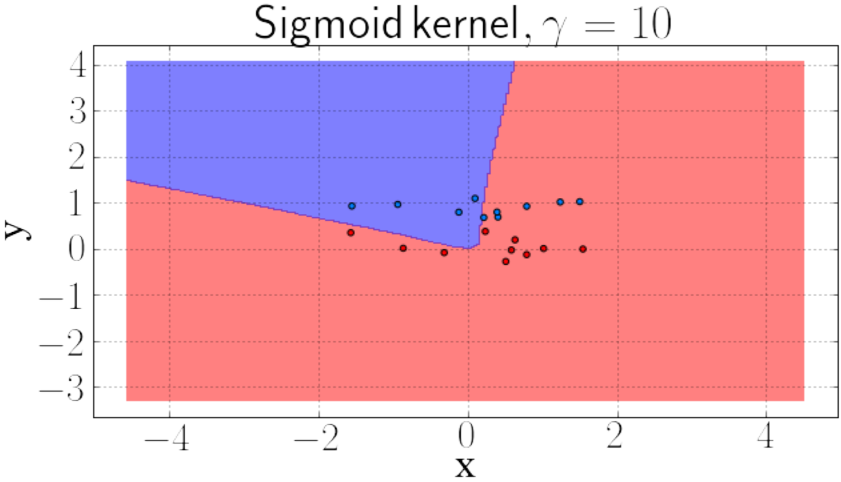
Sigmoid kernel - variable γ



Sigmoid kernel - variable γ



Sigmoid kernel - variable γ



Sigmoid kernel - variable C