Support Vector Machines

Sandro Cumani

sandro.cumani@polito.it

Politecnico di Torino

Support Vector Machines

Let's consider again a binary classification problem

We have shown that logistic regression provides a linear classification rule that maximizes the log–probability of class assignments $\sum_{i=1}^{n} \log P(C_i = c_i | \boldsymbol{X}_i = \boldsymbol{x}_i, \boldsymbol{w}, b)$ for the training set

The LR solution can be found by minimizing the logistic loss

Support Vector Machines

As we have mentioned, it is often useful to add a regularization term of the form $\frac{\lambda}{2}\|\boldsymbol{w}\|^2$:

$$\boldsymbol{w}^*, b = \arg\min_{\boldsymbol{w},b} \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n} \log\left(1 + e^{-z_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)}\right)$$

where z_i is the class label for sample $\boldsymbol{x}_i$, encoded as

$$z_i = \begin{cases} +1 & \text{if } c_i = \mathcal{H}_T \\ -1 & \text{if } c_i = \mathcal{H}_F \end{cases}$$

We can alternatively minimize a scaled version of the objective:

$$\boldsymbol{w}^*, b = \arg\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n} \log\left(1 + e^{-z_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)}\right)$$

where $C = \frac{1}{n\lambda}$

Support Vector Machines

We now consider a different classifier that allows us to give a geometrical interpretation of the regularization term: the Support Vector Machine

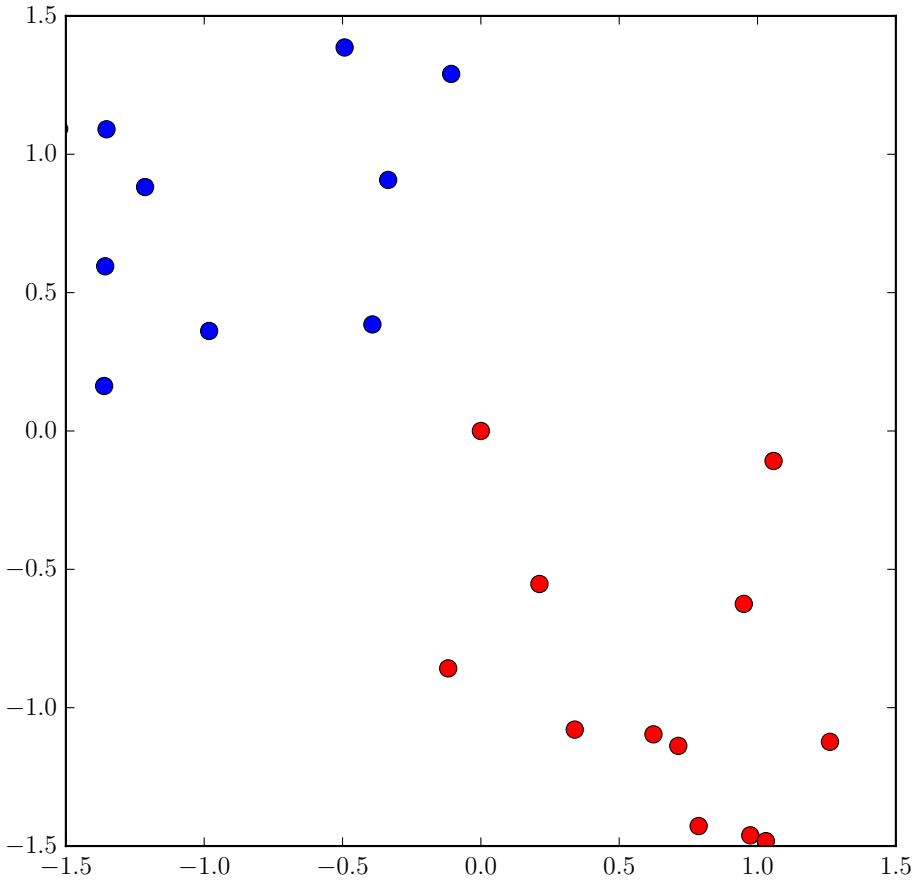
We will show that SVM can be cast as a generalized risk minimization problem

Furthermore, SVMs provide a natural way to achieve non–linear separation without the need for an explicit expansion of our features.

In contrast with LR, however, the output of SVMs cannot be directly interpreted as class posteriors.

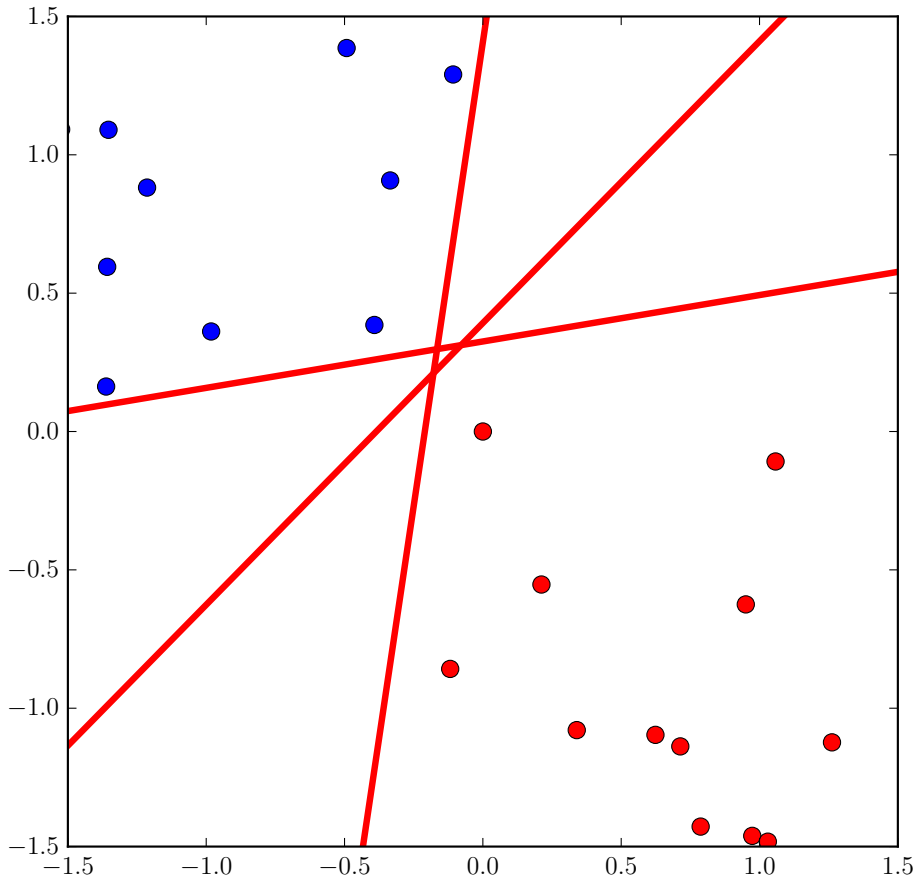
Support Vector Machines

Assume that we have two classes that are linearly separable (i.e., we can find a linear separation hyperplane that correctly classifies all points of our training set)



Support Vector Machines

Problem: there is an infinite number of separating hyperplanes



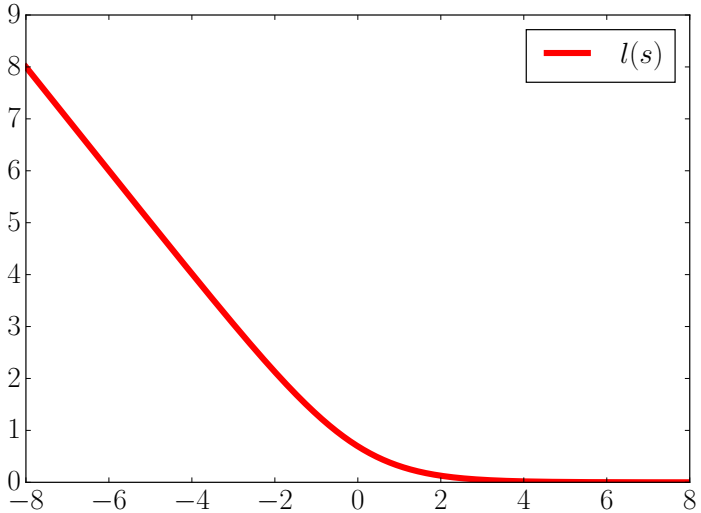
Support Vector Machines

Logistic Regression will find the hyperplane that maximizes class probabilities

Since $P(C_i = c_i|\boldsymbol{x}, \boldsymbol{w}, b) = \sigma(z_i(\boldsymbol{w}^T\boldsymbol{x} + b))$, and classes are separable, posterior probabilities can grow arbitrarily close to 1 and 0 (depending on the class) as long as the norm of $\boldsymbol{w}$ is sufficiently large

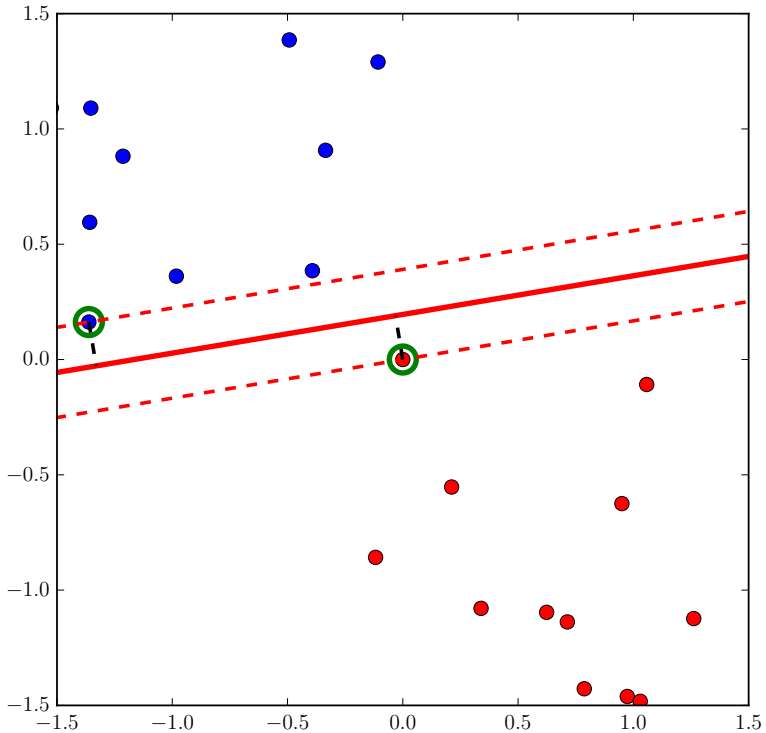
The "optimal" LR solution would have $||\boldsymbol{w}||^2 \rightarrow +\infty$

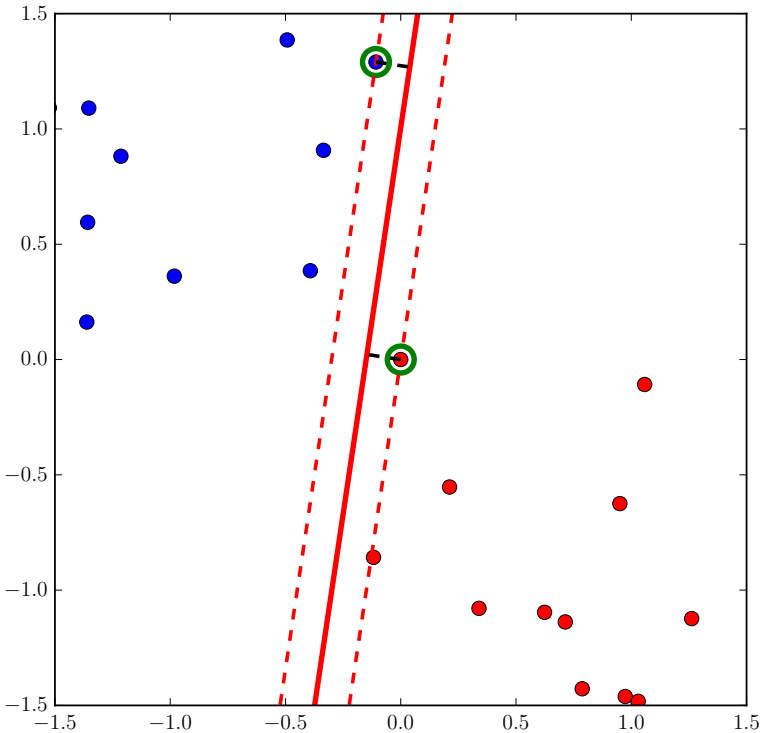
LR optimizes the Logistic Loss. As we move further along the x axis the loss tends to 0



Support Vector Machines

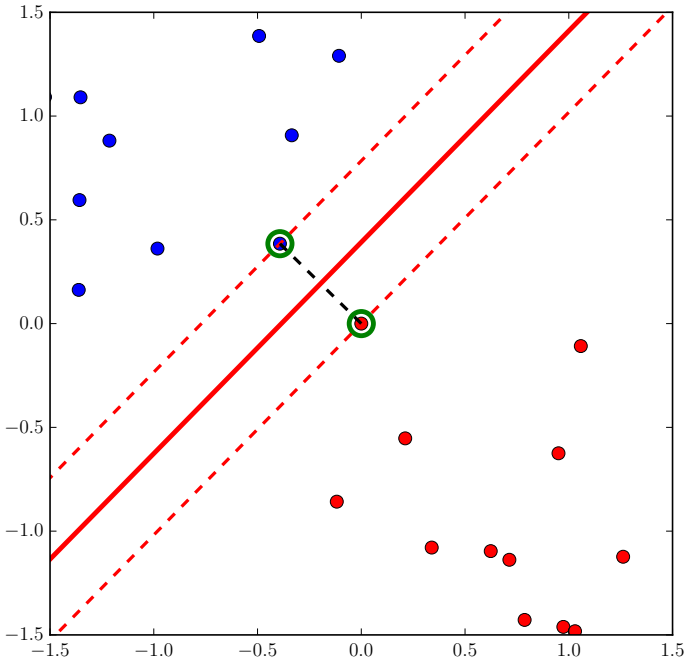
Intuitively, we can select the hyperplane that separates the classes with the largest margin





Support Vector Machines

Intuitively, we can select the hyperplane that separates the classes with the largest margin



The margin is defined as the distance of the closest point w.r.t. the separation hyperplane[1]

[1] Some authors separately introduce the margin for each class. This distinction has no practical effects on the derivation of the SVM objective.

Support Vector Machines

Let $f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$ be the function representing the separation surface

The distance of $\boldsymbol{x}_i$ from the hyperplane is

$$d(\boldsymbol{x}_i) = \frac{|f(\boldsymbol{x}_i)|}{\|\boldsymbol{w}\|}$$

Let z_i denote the class for $\boldsymbol{x}_i$, with the usual encoding

$$z_i = \begin{cases} +1 & \text{if } c_i = \mathcal{H}_T \\ -1 & \text{if } c_i = \mathcal{H}_F \end{cases}$$

Since classes are separable, we consider solutions which correctly classify all points:

$$f(\boldsymbol{x}_i) > 0 \text{ if } c_i = \mathcal{H}_T$$
$$f(\boldsymbol{x}_i) < 0 \text{ if } c_i = \mathcal{H}_F$$

The distance of $\boldsymbol{x}$ from the hyperplane can be rewritten as

$$d(\boldsymbol{x}) = \frac{|f(\boldsymbol{x})|}{\|\boldsymbol{w}\|} = \frac{|z_i(\boldsymbol{w}^T\boldsymbol{x} + b)|}{\|\boldsymbol{w}\|}$$

The maximum margin hyperplane is the hyperplane which maximizes the minimum distance of all points from the hyperplane:

$$\boldsymbol{w}^*, b^* = \arg\max_{\boldsymbol{w},b} \min_{i\in\{1\ldots n\}} d(\boldsymbol{x}_i) = \arg\max_{\boldsymbol{w},b} \min_{i\in\{1\ldots n\}} \frac{|z_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)|}{\|\boldsymbol{w}\|}$$

subject to $z_i(\boldsymbol{w}^T\boldsymbol{x} + b) > 0$.

Support Vector Machines

For values $\boldsymbol{w}, b$ which can correctly separate the classes we have $z_i(\boldsymbol{w}^T\boldsymbol{x} + b) > 0$ for all samples, and $\min_i z_i(\boldsymbol{w}^T\boldsymbol{x} + b) > 0$.

We can drop the constraint by considering an equivalent problem

$$\begin{aligned} \boldsymbol{w}^*, b^* &= \arg\max_{\boldsymbol{w},b} \min_i \frac{z_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)}{\|\boldsymbol{w}\|} \\ &= \arg\max_{\boldsymbol{w},b} \frac{1}{\|\boldsymbol{w}\|} \min_i \left[z_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)\right] \end{aligned} \tag{1}$$

We can show the two problems are equivalent:

- Hyperplanes that do not satisfy this constraint will have $\min_i z_i(\boldsymbol{w}^T\boldsymbol{x} + b) < 0$, and thus cannot be an optimal solution of the second formulation (classes are separable, so an optimal solution exists with $z_i(\boldsymbol{w}^T\boldsymbol{x} + b) > 0\ , \forall i = 1 \ldots n$)
- For hyperplanes that correctly classify all patterns, the objective functions are the same

Direct optimization of (1) is non trivial, thus we further transform the problem as to arrive to an easier-to-solve, equivalent, problem.

We can observe that the objective function in (1) is invariant under re-scaling of the parameters:

$$\frac{1}{\|\boldsymbol{w}\|} \min_i \left[z_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) \right] = \frac{1}{\|\alpha \boldsymbol{w}\|} \min_i \left[z_i(\alpha \boldsymbol{w}^T \boldsymbol{x}_i + \alpha b) \right]$$

for $\alpha > 0$ Thus, if $(\boldsymbol{w}^*, b^*)$ is an optimal solution, then also $(\boldsymbol{w}', b') = (\alpha \boldsymbol{w}^*, \alpha b^*)$ is optimal, and viceversa.

The collections of values $\{(\alpha \boldsymbol{w}, \alpha b)\}_{\alpha \in \mathbb{R}_+}$ forms an equivalence class of equivalent solutions

For each of these equivalence classes, we are free to select any one of the equivalent solutions. In particular, we restrict our problem to solutions for which

$$\min_i z_i(\boldsymbol{w}^T \boldsymbol{x}_k + b) = 1$$

For all training points we will thus have

$$z_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) \geq 1, \quad i = 1 \dots n$$

The problem in (1) then becomes equivalent to optimizing

$$\arg\max_{\boldsymbol{w},b} \frac{1}{\|\boldsymbol{w}\|}$$
$$\text{s.t.} \begin{cases} z_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1, \quad i = 1 \dots n \\ \min_i z_i(\boldsymbol{w}^T\boldsymbol{x}_k + b) = 1 \end{cases}$$

or, equivalently, minimizing the squared norm of $\boldsymbol{w}$:

$$\arg\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2$$
$$\text{s.t.} \begin{cases} z_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1, \quad i = 1 \dots n \\ \min_i z_i(\boldsymbol{w}^T\boldsymbol{x}_k + b) = 1 \end{cases}$$

Finally, we observe that we can drop the last constraint, and solve the equivalent problem

$$\begin{aligned} &\arg\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 \\ &\text{s.t.} \quad z_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1, \quad i = 1 \dots n \end{aligned} \tag{2}$$

Indeed, an optimal solution of (2) will automatically satisfy $\min_i z_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) = 1$

In fact, if $\boldsymbol{w}^*, b^*$ was a solution for which $\min_i z_i(\boldsymbol{w}^T\boldsymbol{x}_k + b) = \psi > 1$, then we could build the solution $(\boldsymbol{w}', b') = \left(\frac{\boldsymbol{w}}{\psi}, \frac{b}{\psi}\right)$ which would still satisfy all constraints, and have a lower norm, thus $\boldsymbol{w}^*, b^*$ would not be optimal

Support Vector Machines

The SVM objective (primal formulation) thus consist in finding the minimizer of

$$\arg\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2$$

$$\text{s.t.} \quad z_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1, \quad i = 1 \ldots n$$

This is a convex quadratic programming problem

- The objective function is convex
- The constraints form a convex set

Support Vector Machines

To solve the SVM problem we consider a Lagrangian formulation of the problem

For each constraint we introduce the Lagrange multiplier $\alpha_i \geq 0$:

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \|\boldsymbol{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left[z_i \left(\boldsymbol{w}^T \boldsymbol{x}_i + b \right) - 1 \right]$$

The optimal solution is obtained by minimizing L w.r.t. $\boldsymbol{w}, b$ while requiring that either the derivatives w.r.t. α_i vanish, subject to the constraints that $\alpha_i \geq 0$, or the corresponding α_i is 0

Since the problem is convex, we can equivalently maximize L w.r.t. α_i while requiring that the derivatives w.r.t. $\boldsymbol{w}$ and b vanish, subject to $\alpha_i \geq 0$

Setting the derivative of L w.r.t. $\boldsymbol{w}$ and b equal to zero gives

$$\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i z_i \boldsymbol{x}_i$$

$$0 = \sum_{i=1}^{n} \alpha_i z_i$$

Replacing these constraints in L gives the *dual* SVM problem

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j z_i z_j \boldsymbol{x}_i^T \boldsymbol{x}_j$$

s.t.

$$\alpha_i \geq 0, \quad i = 1, \ldots, n$$

$$\sum_{i=1}^{n} \alpha_i z_i = 0$$

The dual objective function can be expressed in matrix form as:

$$L_D(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j z_i z_j \boldsymbol{x}_i^T \boldsymbol{x}_j = \boldsymbol{\alpha}^T \mathbf{1} - \frac{1}{2} \boldsymbol{\alpha}^T \boldsymbol{H} \boldsymbol{\alpha}$$

where $\boldsymbol{H}$ is the matrix

$$\boldsymbol{H}_{ij} = z_i z_j \boldsymbol{x}_i^T \boldsymbol{x}_j$$

Matrix $\boldsymbol{H}$ depends on the data only through dot–products $\boldsymbol{x}_i^T \boldsymbol{x}_j$

As we will see shortly, this allows to extend SVMs to non–linear classification

For any feasible solution of the primal problem $\boldsymbol{w}, b$ and any feasible solution of the dual problem $\boldsymbol{\alpha}$ we have that

$$L_D(\boldsymbol{\alpha}) \leq L_P(\boldsymbol{w}, b)$$

where

$$L_P = \frac{1}{2} \|\boldsymbol{w}\|^2$$

is the primal objective

For a pair of optimal primal-dual solutions, the two values become equal (strong duality):

$$L_D(\boldsymbol{\alpha}^*) = L_P(\boldsymbol{w}^*, b^*)$$

The difference

$$L_P(\boldsymbol{w}, b) - L_D(\boldsymbol{\alpha}) \geq 0$$

is called duality gap, and is 0 for optimal solutions

A solution $\boldsymbol{w}, b, \boldsymbol{\alpha}$ will be optimal if and only if it satisfies the Karush–Kuhn–Tucker (KKT) conditions[2]

$$\nabla_{\boldsymbol{w}} L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \boldsymbol{w} - \sum_{i=1}^{n} \alpha_i z_i \boldsymbol{x}_i = \mathbf{0}$$

$$\frac{\partial L(\boldsymbol{w}, b, \boldsymbol{\alpha})}{\partial b} = -\sum_{i=1}^{n} \alpha_i z_i = 0$$

$$z_i \left(\boldsymbol{w}^T \boldsymbol{x}_i + b\right) - 1 \geq 0 \qquad \forall i$$

$$\alpha_i \geq 0 \qquad \forall i$$

$$\alpha_i \left[z_i \left(\boldsymbol{w}^T \boldsymbol{x}_i + b\right) - 1\right] = 0 \qquad \forall i$$

[2]These are both necessary and sufficient conditions for optimality of the SVM solution

The first and second equations encode that, at the optimal primal solution $(\boldsymbol{w}, b)$, the gradient of the Lagrangian with respect to $\boldsymbol{w}$ and b becomes zero.

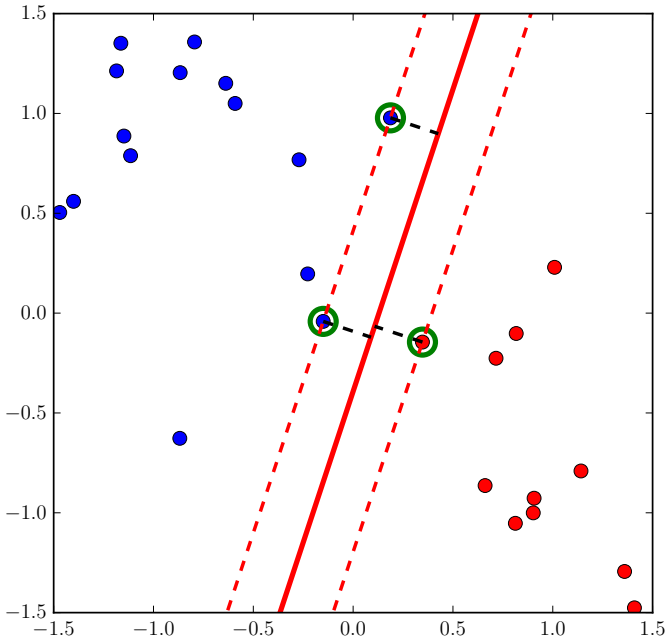
The next two equations encode that both the primal solution $(\boldsymbol{w}, b)$ and the corresponding dual solution $\boldsymbol{\alpha}$ are feasible.

The last equation encodes that the optimal dual solution maximizes the Lagrangian. This requires that either the maximum is on the constraint, i.e. $\alpha_i = 0$, or the derivative of the Lagrangian is $\frac{\partial L}{\partial \alpha_i} = 0$. In compact form, $\alpha_i \frac{\partial L}{\partial \alpha_i} = 0$

Support Vector Machines

The optimal solution satisfies the Karush–Kuhn–Tucker (KKT) conditions

- For all points that do not lie on the margin (i.e. $z_i \left(\boldsymbol{w}^T \boldsymbol{x}_i + b\right) > 1$) the corresponding Lagrange multiplier is $\alpha_i = 0$
- $\alpha_i \neq 0$ implies that the corresponding point is on the margin, i.e., it is a Support Vector
- After we have obtained $\boldsymbol{\alpha}$, the KKT conditions allow us to estimate b



Support Vector Machines

Since $\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i z_i \boldsymbol{x}_i$, the score for a test point x_t can be computed as

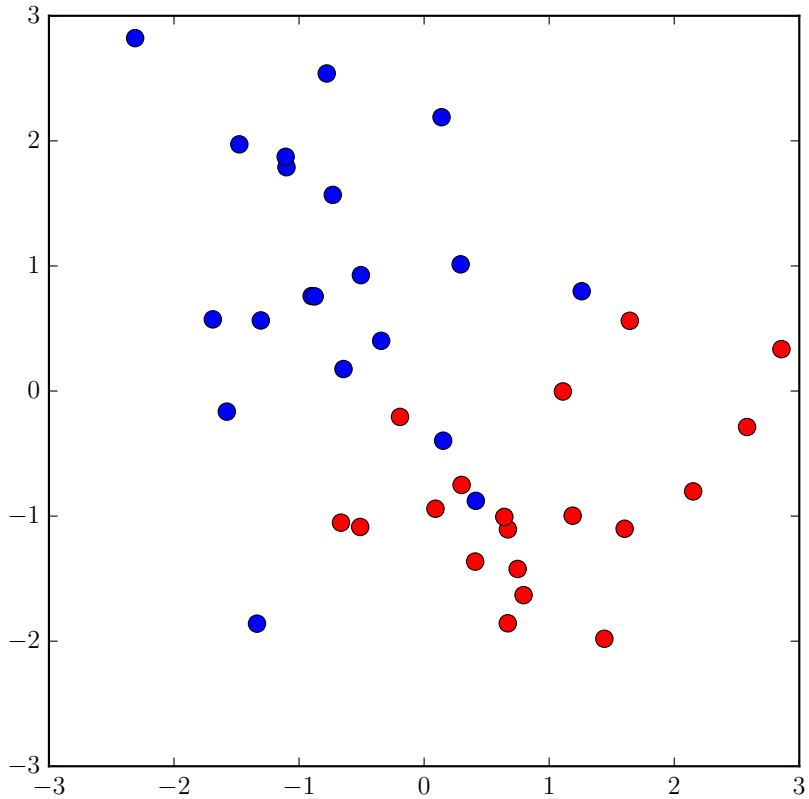
$$s(\boldsymbol{x}_t) = \boldsymbol{w}^T \boldsymbol{x}_t + b = \sum_{i=1}^{n} \alpha_i z_i \boldsymbol{x}_i^T \boldsymbol{x}_t + b$$

Notice that, again, this depends only on dot products $\boldsymbol{x}_i^T \boldsymbol{x}_t$

The training points that are not Support Vectors do not affect the separation surface

Support Vector Machines

Let's consider a problem where classes are not linearly separable



No matter the value of $\boldsymbol{w}$, some points will violate the constraint

$$z_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1$$

We can try to minimize the number of points that violate the constraint

To do this, we introduce the slack variables $\xi_i \geq 0$, which represent how much a point is violating the constraint

We can replace the constraints $z_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1$ with

$$z_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 - \xi_i, \qquad \xi_i \geq 0$$

i.e. we allow training points to be inside the margin by a factor ξ_i

We want to minimize the number of data points that lie inside the margin, i.e. the number of points for which $\xi_i > 0$

We can consider the functional

$$\Phi(\boldsymbol{\xi}) = \sum_{i=1}^{n} \xi_i^{\sigma}$$

with $\sigma > 0$

For sufficiently small values of σ, $\Phi(\boldsymbol{\xi})$ represents the number of points inside the margin

If we remove these data points the classes become linearly separable and we can thus train a maximum margin hyperplane over the remaining points

Formally, this corresponds to the minimization of the functional

$$\frac{1}{2}\|\boldsymbol{w}\|^2 + CF\left(\sum_{i=1}^{n}\xi_i^{\sigma}\right)$$

s.t.

$$\begin{aligned} z_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) &\geq 1 - \xi_i \qquad \forall i \\ \xi_i &\geq 0 \qquad \forall i \end{aligned}$$

where F is a monotone convex function and C is a constant

For sufficiently large C and small σ the optimal solution minimizes the number of points that violate the margin constraints and maximizes the margin between the remaining points

Unfortunately, for small values of σ the problem is difficult

We simplify the problem by considering $\sigma = 1$, which guarantees a unique solution for the separation surface, and[3] $F(u) = u$

The objective function becomes

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{s.t. } z_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 - \xi_i \qquad \forall i = 1\ldots n$$

$$\xi_i \geq 0 \qquad \forall i = 1\ldots n$$

Note that this is again a convex quadratic programming problem

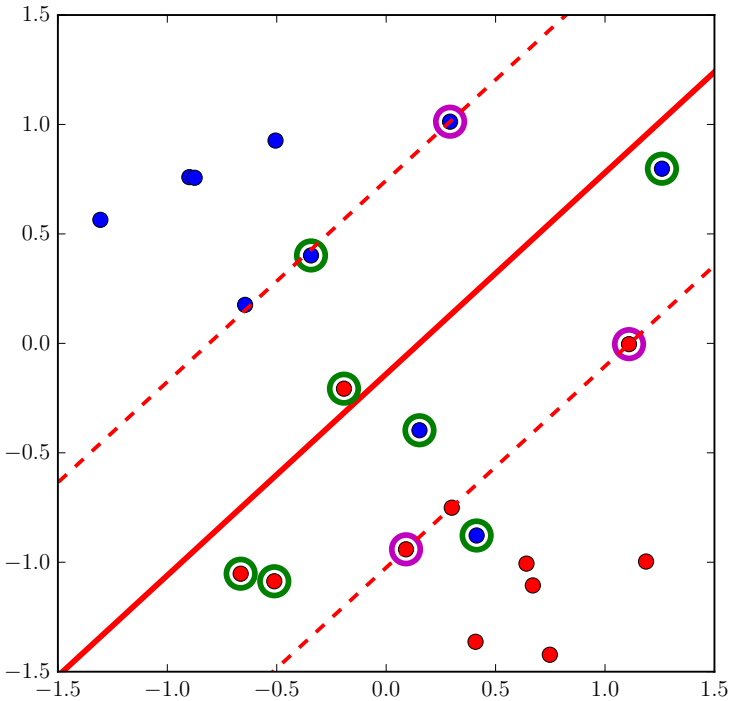
[3] SVMs with $\sigma = 2$ are sometimes used, and are usually referred to as L2-SVMs

Support Vector Machines

The terms ξ_i do not describe the number of errors, but act as a penalty

- Points inside the margin have $\xi_i > 0$, and miss–classified points have $\xi_i > 1$
- The further the point is from the hyperplane, the larger the value of ξ_i
- $\sum_{i=1}^{n} \xi_i$ is an upper bound on the number of miss–classified points (errors)
- C allows selecting a trade–off between margin and errors on the training set
- The solution is often called a *soft margin* hyperplane

Soft margin classification:



As we did for the hard–margin SVM, we can introduce the Lagrangian problem

$$L(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\mu}) =$$

$$= \frac{1}{2} \|\boldsymbol{w}\|^2 + C \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i \left[z_i \left(\boldsymbol{w}^T \boldsymbol{x} + b \right) - 1 + \xi_i \right] - \sum_{i=1}^{n} \mu_i \xi_i$$

where $\alpha_i \geq 0$, and $\mu_i \geq 0$ are an additional set of Lagrange multipliers relative to the constraints $\xi_i \geq 0$

The necessary and sufficient KKT conditions are

$$\nabla_{\boldsymbol{w}} L(\boldsymbol{w}, b, \boldsymbol{\alpha}, \boldsymbol{\mu}) = \boldsymbol{w} - \sum_{i=1}^{n} \alpha_i z_i \boldsymbol{x}_i = \mathbf{0}$$

$$\frac{\partial L(\boldsymbol{w}, b, \boldsymbol{\alpha}, \boldsymbol{\mu})}{\partial b} = -\sum_{i=1}^{n} \alpha_i z_i = 0$$

$$\frac{\partial L(\boldsymbol{w}, b, \boldsymbol{\alpha}, \boldsymbol{\mu})}{\partial \xi_i} = C - \alpha_i - \mu_i = 0 \qquad \forall i$$

$$z_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) - 1 + \xi_i \geq 0 \qquad \forall i$$

$$\xi_i \geq 0 \qquad \forall i$$

$$\alpha_i \geq 0 \qquad \forall i$$

$$\mu_i \geq 0 \qquad \forall i$$

$$\alpha_i \left[z_i \left(\boldsymbol{w}^T \boldsymbol{x}_i + b \right) - 1 + \xi_i \right] = 0 \qquad \forall i$$

$$\mu_i \xi_i = 0 \qquad \forall i$$

From the KKT conditions (1)–(3)

$$\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i z_i \boldsymbol{x}_i \quad , \qquad \sum_{i=1}^{n} \alpha_i z_i = 0 \; , \qquad \alpha_i = C - \mu_i$$

Note that $\mu_i \geq 0$ implies that $\alpha_i \leq C$

We can then optimize the Lagrangian w.r.t. $\boldsymbol{w}$, b and $\boldsymbol{\xi}$ to obtain the dual problem

$$\max_{\boldsymbol{\alpha}} L_D(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j z_i z_j \boldsymbol{x}_i^T \boldsymbol{x}_j$$

s.t.

$$0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n$$

$$\sum_{i=1}^{n} \alpha_i z_i = 0$$

Note that this problem is very similar to the hard–margin SVM

The constraint

$$\alpha_i \geq 0$$

is replaced by

$$0 \leq \alpha_i \leq C$$

Predictions are again computed as in the hard–margin case

$$s(\boldsymbol{x}_t) = \boldsymbol{w}^T \boldsymbol{x}_t + b = \sum_{i=1}^{n} \alpha_i z_i \boldsymbol{x}_i^T \boldsymbol{x}_t + b$$

Again, this depends only on dot products $\boldsymbol{x}_i^T \boldsymbol{x}_t$

Several methods to solve the *dual* problem

Packages are available for several solvers

If we are interested in linear separation we can directly solve the *primal* SVM problem

Let's recall the primal formulation

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

s.t.

$$z_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 - \xi_i \qquad \forall i$$
$$\xi_i \geq 0 \qquad \forall i$$

For data points that are on the correct side of the margin

$$\xi_i = 0$$

while for the others we have

$$\xi_i = 1 - z_i(\boldsymbol{w}^T \boldsymbol{x}_i + b)$$

Thus the SVM primal problem can then be rewritten as

$$\min_{\boldsymbol{w},b} \frac{1}{2} \|\boldsymbol{w}\|^2 + C \sum_{i=1}^{n} \max\left[0, 1 - z_i(\boldsymbol{w}^T \boldsymbol{x}_i + b)\right]$$

where the constraints have been absorbed by the *loss* terms

$$\max\left[0, 1 - z_i(\boldsymbol{w}^T \boldsymbol{x}_i + b)\right]$$

Function

$$f(s) = \max(0, 1 - s)$$

is known as *hinge loss*, and is sometimes denoted as

$$f(s) = [1 - s]_+$$

We can rewrite the objective as

$$\min_{\boldsymbol{w},b} \frac{\lambda}{2} \|\boldsymbol{w}\|^2 + \frac{1}{n} \sum_{i=1}^{n} \max\left[0, 1 - z_i(\boldsymbol{w}^T \boldsymbol{x}_i + b)\right]$$

Compare with Logistic Regression:

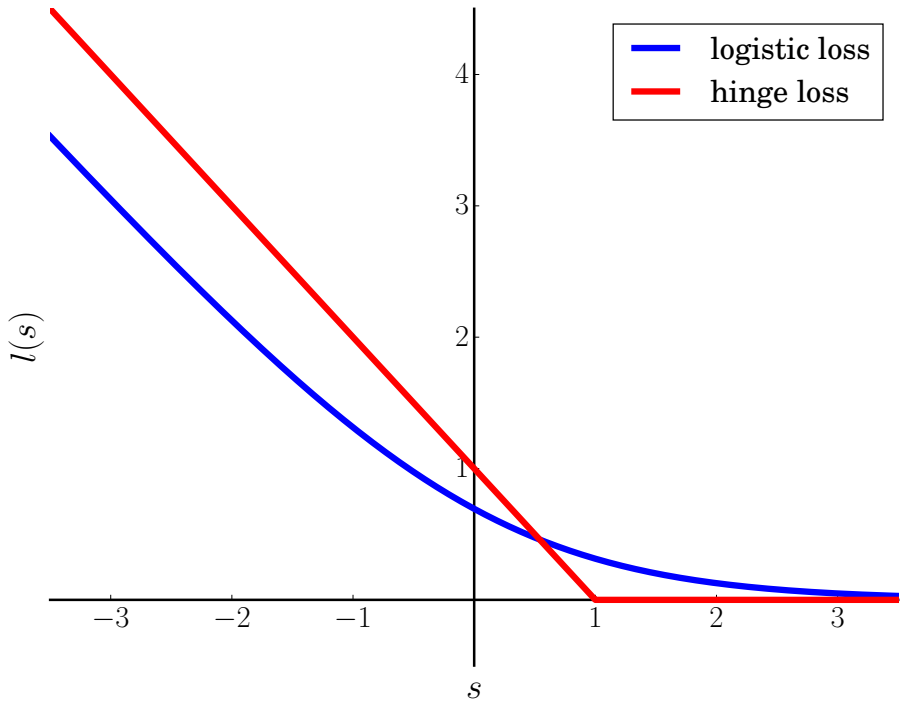
$$\min_{\boldsymbol{w},b} \frac{\lambda}{2} \|\boldsymbol{w}\|^2 + \frac{1}{n} \sum_{i} \log\left[1 + e^{-z_i\left(\boldsymbol{w}^T \boldsymbol{x}_i + b\right)}\right]$$

LR: Logistic Loss

$$l(s) = \log\left(1 + e^{-s}\right)$$

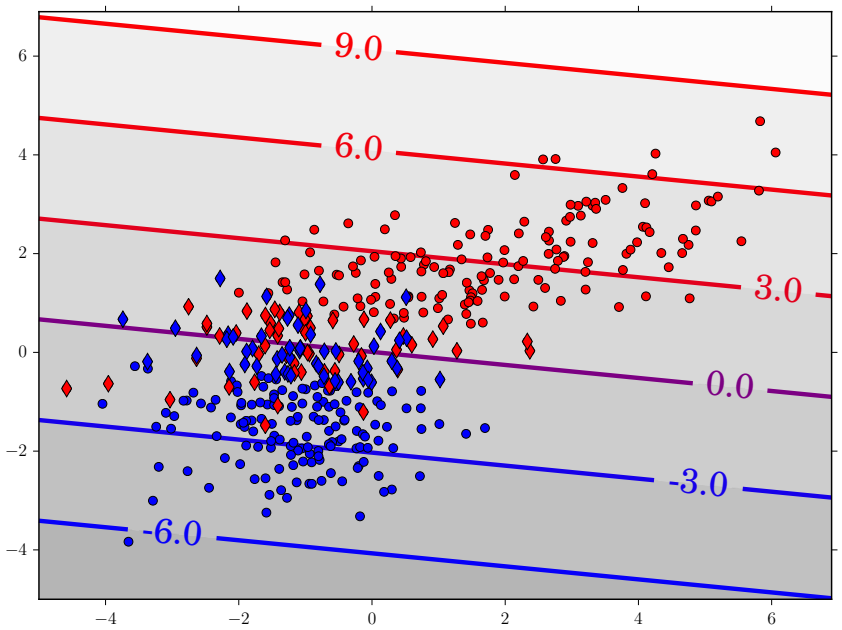
SVM: Hinge Loss

$$l(s) = \max(0, 1 - s)$$

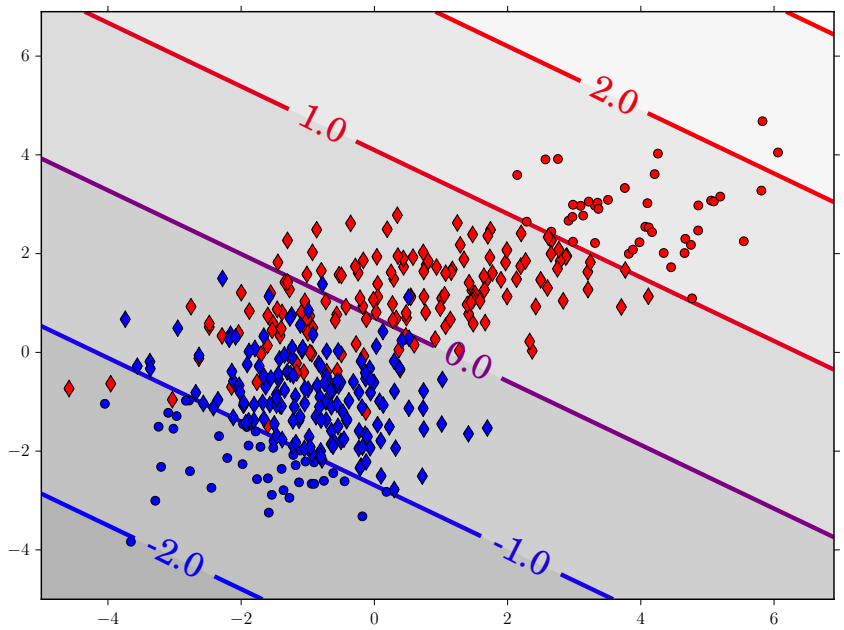


Binary classification:

$C = 1.0\ (\lambda = 0.0025)$



$C = 0.001\ (\lambda = 2.5)$



Support Vector Machines

We have seen that we can obtain the separation hyperplane by solving either the primal or the dual problem

Primal

Minimize w.r.t. $\boldsymbol{w}$ and b

$$L_P(\boldsymbol{w}, b) = \\ = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\left[1 - z_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)\right]_+$$

Dual

Maximize w.r.t. $\boldsymbol{\alpha}$

$$L_D(\boldsymbol{\alpha}) = \alpha^T\mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{H}\boldsymbol{\alpha}$$

subject to

$$0 \leq \alpha_i \leq C,\ i = 1, \ldots, n$$

$$\sum_{i=1}^{n}\alpha_i z_i = 0$$

For a pair of feasible solutions $L_P(\boldsymbol{w}, b) - L_D(\boldsymbol{\alpha})$ is again the duality gap
At the optimum, the gap becomes 0, i.e. $L_P(\boldsymbol{w}^*, b^*) = L_D(\boldsymbol{\alpha}^*)$

Primal

Parameters $\boldsymbol{w} \in \mathbb{R}^D$, $b \in \mathbb{R}$

Scoring complexity is $O(D)$:

$$s(\boldsymbol{x}_t) = \boldsymbol{w}^T \boldsymbol{x}_t + b$$

Embedding a non–linear transformation requires explicitly defining the mapping from $\mathbb{R}^D$ to the Hilbert space $\mathcal{H}$

$$\boldsymbol{\Phi} : \mathbb{R}^D \longrightarrow \mathcal{H}$$

Dual

Parameters $\boldsymbol{\alpha} \in \mathbb{R}^N$

Scoring complexity is $O(SV)$:

$$\begin{aligned} s(\boldsymbol{x}_t) &= \sum_{i=1}^{N} \alpha_i z_i \boldsymbol{x}_i^T \boldsymbol{x}_t + b \\ &= \sum_{i|\alpha_i>0} \alpha_i z_i \boldsymbol{x}_i^T \boldsymbol{x}_t + b \end{aligned}$$

Embedding a non–linear transformation only requires dot–products in the expanded space

$$\boldsymbol{\Phi}(\boldsymbol{x}_i)^T \boldsymbol{\Phi}(\boldsymbol{x}_j)$$

If we have a function that efficiently computes the dot-products in the expanded space

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \boldsymbol{\Phi}(\boldsymbol{x}_1)^T \boldsymbol{\Phi}(\boldsymbol{x}_2)$$

then both training and scoring can be performed by using only k

Remember that, for mapping $\boldsymbol{\Phi}$, matrix $\boldsymbol{H}$ is given

$$\boldsymbol{H}_{ij} = z_i z_j \boldsymbol{\Phi}(x_i)^T \boldsymbol{\Phi}(\boldsymbol{x}_j)$$

From the definition of k:

$$\boldsymbol{H}_{ij} = z_i z_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

Scoring becomes

$$s(\boldsymbol{x}_t) = \sum_{i=1|\alpha_i>0} \alpha_i z_i \boldsymbol{\Phi}(\boldsymbol{x}_i)^T \boldsymbol{\Phi}(\boldsymbol{x}_t) + b = \sum_{i=1|\alpha_i>0} \alpha_i z_i k(\boldsymbol{x}_i, \boldsymbol{x}_t) + b$$

Function k is called kernel function

A kernel function allows training a SVM in a large (even infinite) dimensional Hilbert space $\mathcal{H}$, without requiring to explicitly compute the mapping

Even if the expansion was finite, the complexity of the primal problem may be too large

On the other hand, the complexity of the dual problem only depends on the number of training points.

In practice, we are computing a linear separation surface in the expanded space, which corresponds to a non–linear separation surface in the original feature space

We have already seen an example of feature expansion that provides quadratic separation surfaces, given by the mapping

$$\boldsymbol{\Phi}(\boldsymbol{x}) = \begin{bmatrix} vec(\boldsymbol{x}\boldsymbol{x}^T) \\ \sqrt{2}\boldsymbol{x} \\ 1 \end{bmatrix} \tag{3}$$

We can compute

$$\boldsymbol{\Phi}(\boldsymbol{x}_1)^T \boldsymbol{\Phi}(\boldsymbol{x}_2) = \left(\boldsymbol{x}_1^T \boldsymbol{x}_2\right)^2 + 2\boldsymbol{x}_1^T \boldsymbol{x}_2 + 1$$

It is easy verifying that the corresponding kernel is

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \left(\boldsymbol{x}_1^T \boldsymbol{x}_2 + 1\right)^2$$

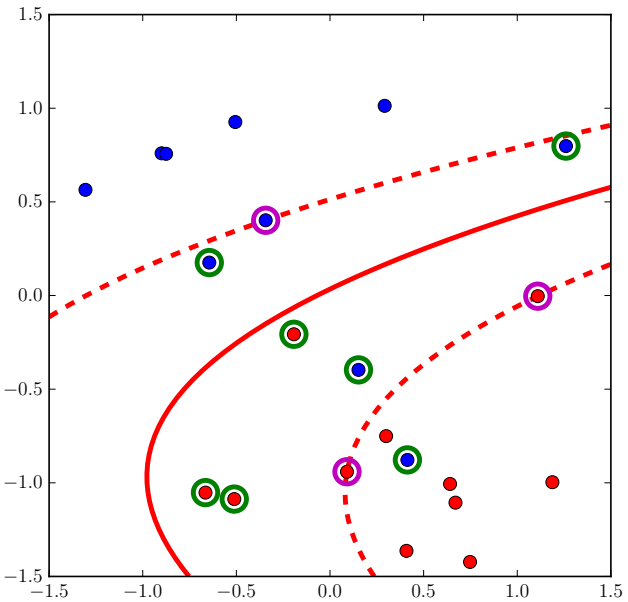
Support Vector Machines

In general, we can define the polynomial kernels of degree d as

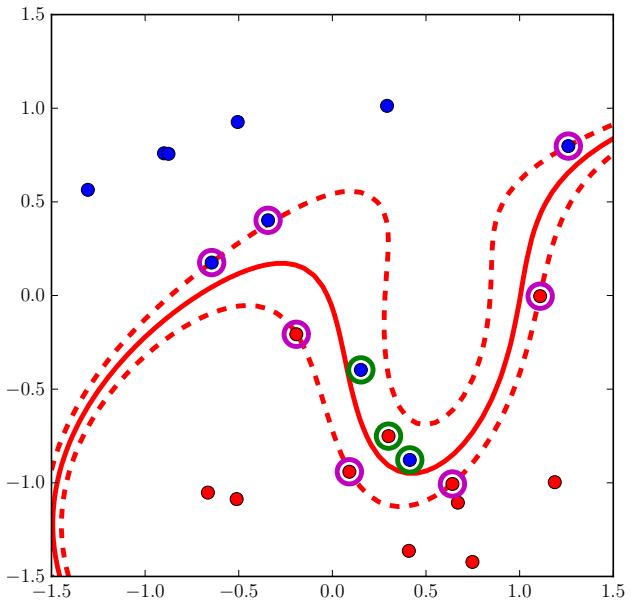
$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \left(\boldsymbol{x}_1^T \boldsymbol{x}_2 + 1\right)^d$$

Note that the expanded space, although finite, grows very quickly

Poly — $d = 2$, $C = 1.0$



Poly — $d = 4$, $C = 10.0$



Support Vector Machines

It is clear that, if we know Φ, we can define a corresponding kernel function

In general, we want to know for which kernel function we can actually find Φ and $\mathcal{H}$

Mercer's condition provides a sufficient condition for k to define a dot–product in an expanded space

Let $k(\boldsymbol{u}, \boldsymbol{v})$ be a symmetrical function in L_2

If, for all functions $g(\boldsymbol{u})$ such that

$$\int g(\boldsymbol{u})^2 d\boldsymbol{u} < \infty$$

we have that

$$\int k(\boldsymbol{u}, \boldsymbol{v}) g(\boldsymbol{u}) g(\boldsymbol{v}) d\boldsymbol{u} d\boldsymbol{v} > 0$$

then k defines a dot–product in some expanded space

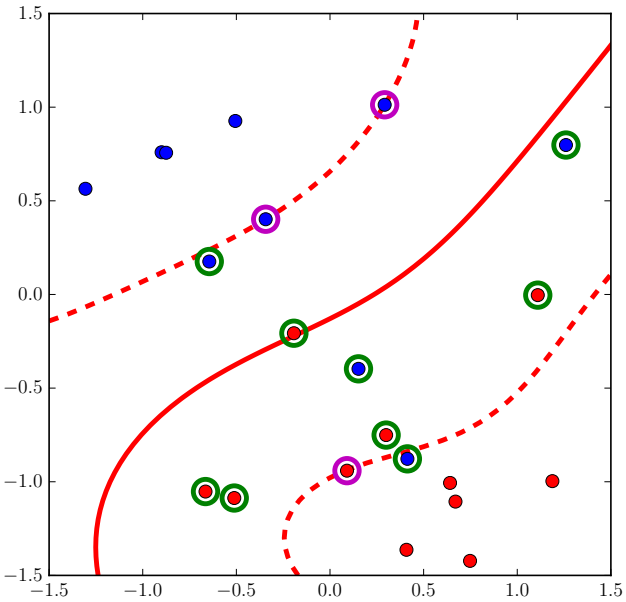
This condition does not tell us how to construct suitable kernels

In general, we can make use of well known kernels, or use rules to combine kernel functions that guarantee that the combination is still a kernel (e.g., summing kernel functions)

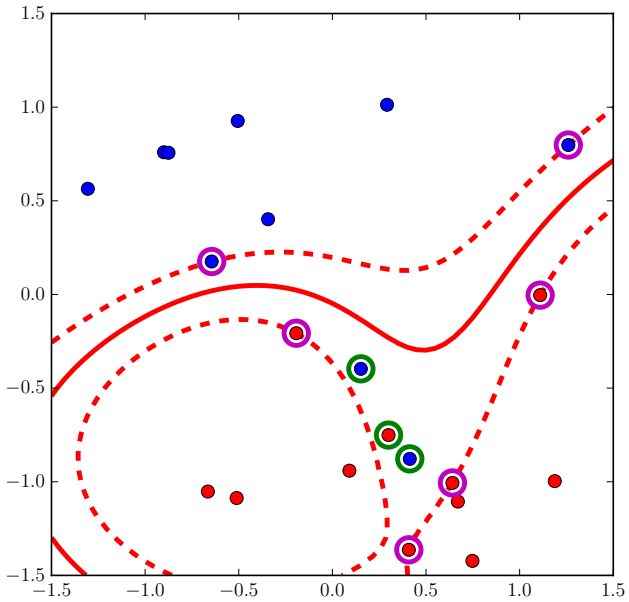
Another example of kernel function associated to an infinite–dimensional mapping (Gaussian Radial Basis Function kernel):

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = e^{-\gamma \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2}$$

RBF — $\gamma = 0.5$, $C = 1.0$



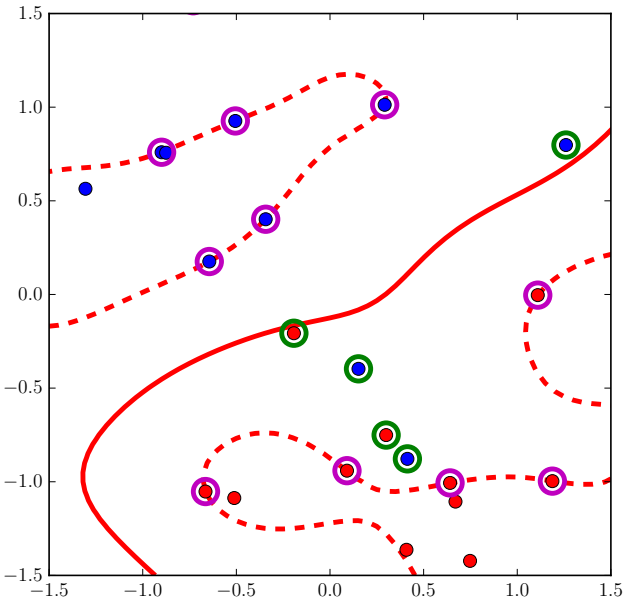
RBF — $\gamma = 0.5$, $C = 100.0$



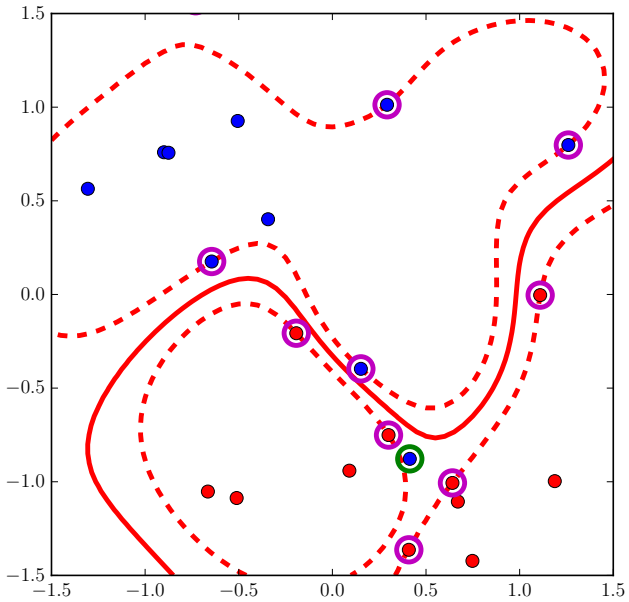
Another example of kernel function associated to an infinite–dimensional mapping (Gaussian Radial Basis Function kernel):

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = e^{-\gamma \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2}$$

RBF — $\gamma = 2.0$, $C = 1.0$



RBF — $\gamma = 2.0$, $C = 100.0$



Support Vector Machines

Another example of kernel function associated to an infinite–dimensional mapping (Gaussian Radial Basis Function kernel):

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = e^{-\gamma \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2}$$

The kernel depends on the distance of the points

"Similar" (i.e. closer) points have higher kernel value

Another example of kernel function associated to an infinite–dimensional mapping (Gaussian Radial Basis Function kernel):

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = e^{-\gamma \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2}$$

γ defines the width of the kernel

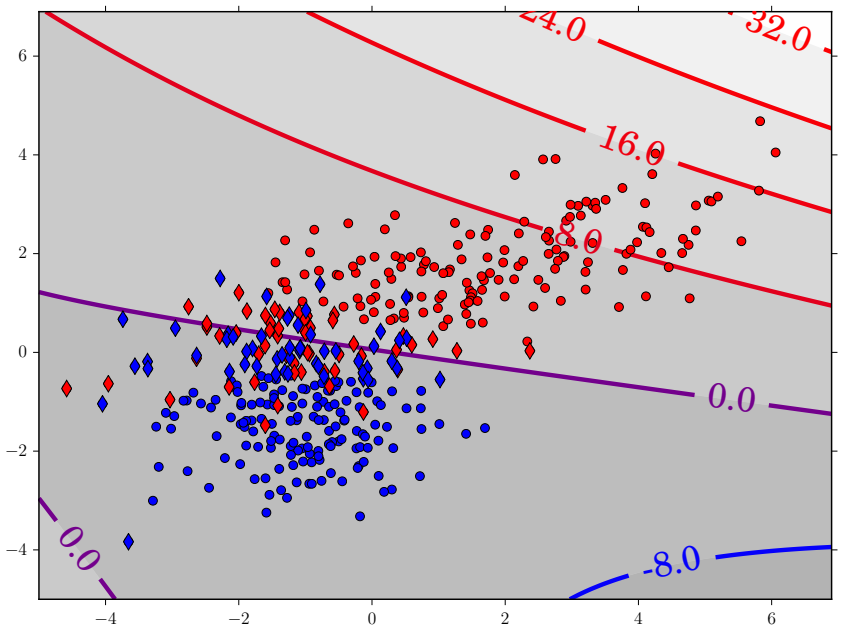
- Small γ: the kernel is wide, a S.V. influences most other points
- Large γ: the kernel is narrow, a S.V. has very small influence on points that are not close. Assuming[4] $b = 0$, we obtain the same classification rule of 1-NN[5]

[4] Remember that in general we assign labels comparing scores with a threshold that depends on our application, and is not necessarily 0

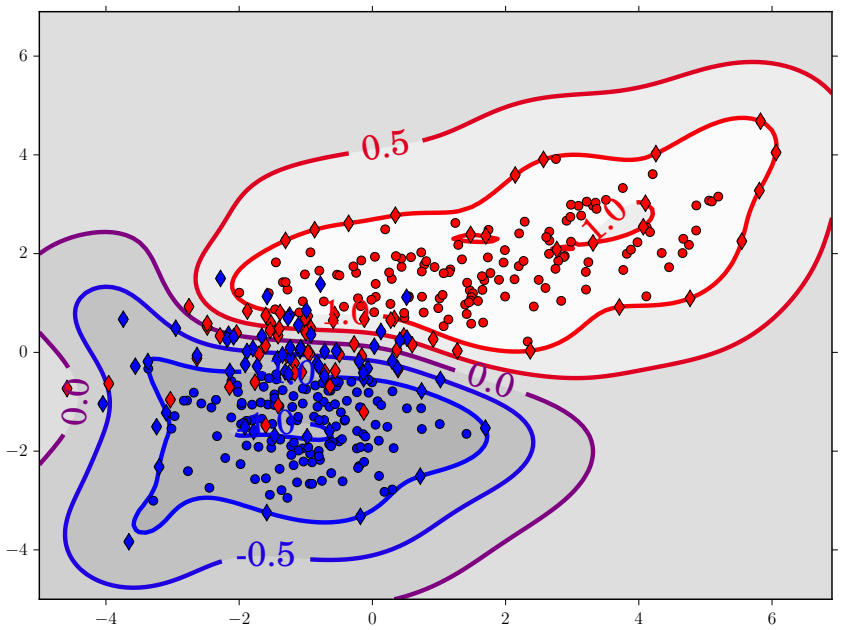
[5] In practice we should not use SVMs as replacement for K-NN, since we will incur into numerical problems due to test points all having scores $s \approx 0$

Binary classification:

Poly — $d = 2$, $C = 1.0$

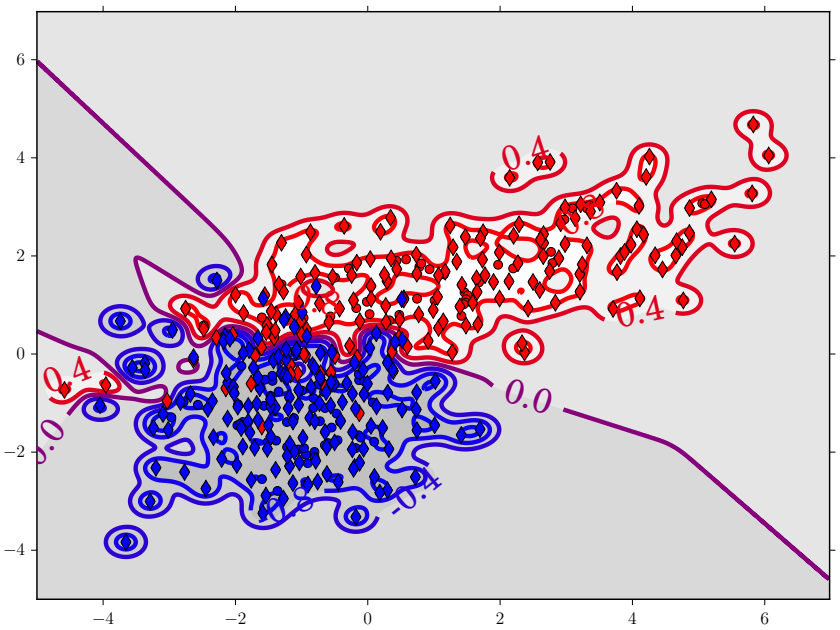


RBF — $\gamma = 0.5$, $C = 1.0$

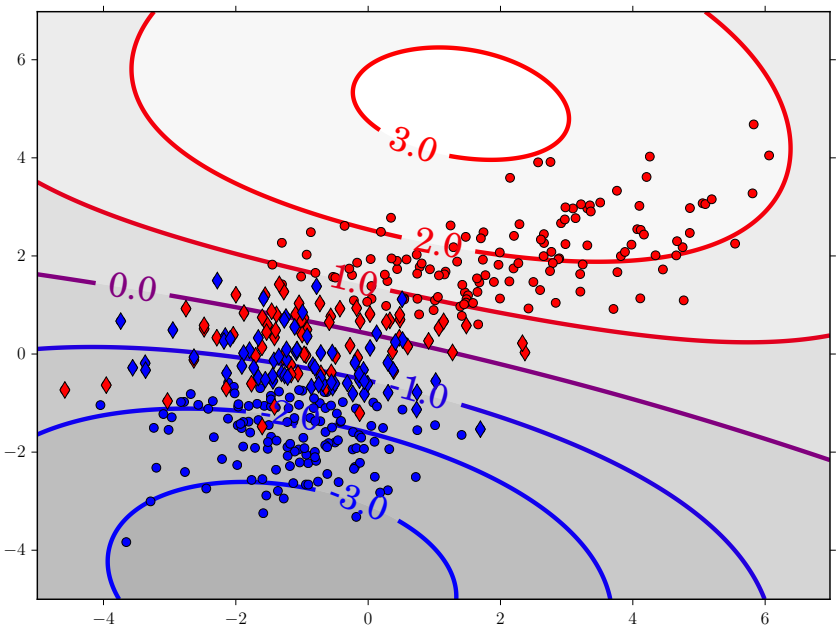


Binary classification:

RBF (no bias) — $\gamma = 10$, $C = 1.0$

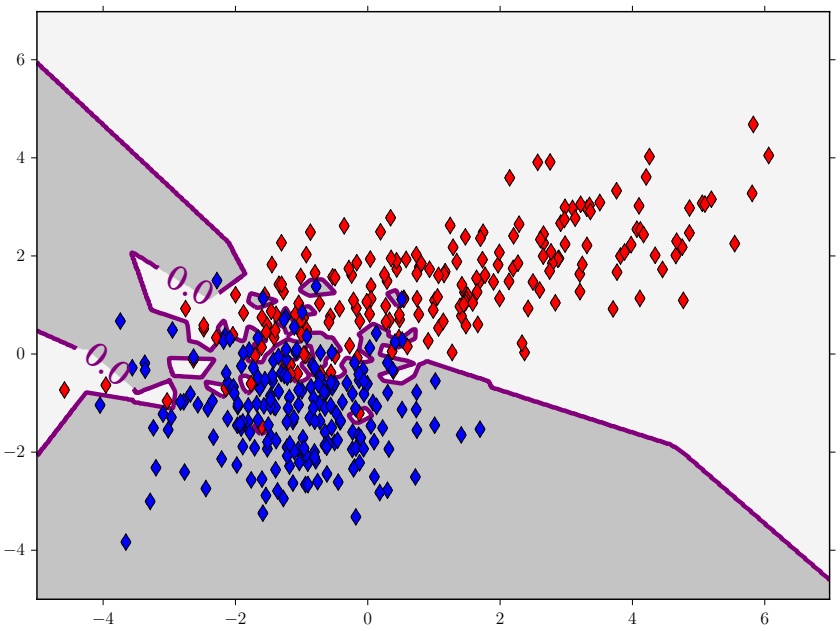


RBF (no bias) — $\gamma = 0.02$, $C = 1.0$

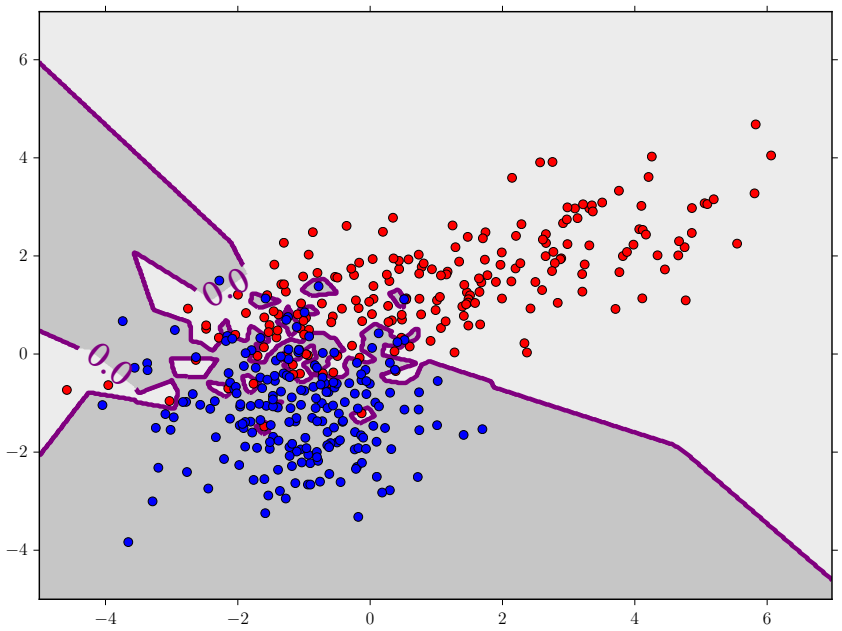


Binary classification:

RBF (no bias) — $\gamma = 200$, $C = 1.0$



1–NN



Support Vector Machines

Practical considerations:

- We need to take care in selecting the appropriate kernel
- Kernels often include hyper-parameters (e.g. γ for RBF)
  - Cross–validation can help selection of good values
- We need to select good values of the coefficient C
  - Again, cross–validation can help

Support Vector Machines

Practical considerations:

- In some cases linear classifiers are sufficient
  - Our feature set is of sufficient high dimension to make our classes almost linearly separable

  Fast algorithms for training the primal problem are available

  Furthermore, linear SVM scoring can be implemented as a simple dot–product
- SVM training is not invariant under affine transformations
  - Feature pre-processing can be relevant
  - Often it is useful to center and whiten data (see the Logistic Regression slides)

Support Vector Machines

Practical considerations:

- SVM scores have no probabilistic interpretation
  - Score post-processing can be applied to estimate class posterior probabilities (e.g. we can train a generative or discriminative probabilistic model over SVM scores using an held–out dataset)
- Finally, we have to take care when training with highly unbalanced datasets (or, more precisely, when the empirical training set prior is significantly different from the target application prior). In this case, weighting the costs of different errors may be beneficial (but we still don't have a probabilistic interpretation for the score)

To re-balance the classes we can use a different value of C for the different classes (or even for different samples):

$$\arg\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 + \sum_{i=1}^{n} C_i \left[1 - z_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)\right]_+$$

The corresponding dual problem is[6]

$$\arg\max_{\alpha} \boldsymbol{\alpha}^T\mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{H}\boldsymbol{\alpha}$$

$$\text{s.t. } \sum_{i=1}^{n} \alpha_i z_i = 0\,, \quad 0 \le \alpha_i \le C_i,\ i = 1,\ldots,n$$

[6] In the laboratory we will omit the constraint related to the bias, since we will not explicitly model the bias term

Support Vector Machines

For class balancing, we can set $C_i = C_T$ for samples of class $\mathcal{H}_T$ and $C_i = C_F$ for samples of class $\mathcal{H}_F$

We can select $C_T = C\frac{\pi_T}{\pi_T^{emp}}$ and $C_F = C\frac{\pi_F}{\pi_F^{emp}}$

π_T^{emp} and π_F^{emp} are the empirical priors (i.e. sample proportions) for the two classes computed over the training set

In practice, we are simulating the application class proportions at training time

MNIST — Comparison of average pairwise EER for different kernels (raw features)

Kernel	$C = 10$	$C = 1$	$C = 0.1$	LogReg ($\lambda = 1e^{-3}$)
Linear	1.6%	1.3%	1.1%	1.2%
Poly ($d = 2$)	0.3%	0.3%	0.3%	0.9%[7]
Poly ($d = 3$)	0.4%	0.4%	0.4%	—
RBF ($\gamma = 0.2$)	0.7%	0.7%	0.9%	—
RBF ($\gamma = 1.0$)	0.5%	0.5%	0.6%	—
RBF ($\gamma = 2.0$)	0.1%	0.1%	0.1%	—

[7] Quadratic expansion of PCA–reduced features

Support Vector Machines

SVMs provide good performance for binary classification problems

Difficult to extend to multiclass problems

Some possibilities:

- One–versus–all training
- Train over all possible pairs

In both cases we need some voting scheme to decide the actual class

Some possibilities (cont.):

- Introduce an objective function that maximizes the margin between each class and all the others. For example, (we ignore the bias terms)

$$\min_{\boldsymbol{W}} \frac{1}{2} \|\boldsymbol{W}\|_2^2 + C \sum_{i=1}^{n} \max_{y'} \left[(\boldsymbol{w}_{y'} - \boldsymbol{w}_{z_i})^T \boldsymbol{x}_i + \delta_{y',z_i} \right]$$

Compare with multiclass logistic regression

$$\min_{\boldsymbol{W}} \frac{\lambda}{2} \|\boldsymbol{W}\|_2^2 + \frac{1}{n} \sum_i \log \left(\sum_{y'} e^{(\boldsymbol{w}_{y'} - \boldsymbol{w}_{z_i}^T)^T \boldsymbol{x}_i} \right)$$

Support Vector Machines

Kernel methods are not restricted to SVMs

- Logistic Regression can be extended to incorporate kernels
- In general, we can apply the kernel tricks to problems which depend only on dot–products
  - Kernel PCA
  - Gaussian Processes
  - ...