FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
NATURWISSENSCHAFTLICHE
FAKULTÄT

Support Vector Machines

Lecture “Mathematics of Learning” 2022

Andreas Bärmann
Friedrich-Alexander-Universität Erlangen-Nürnberg

WIRTSCHAFTS
MATHEMATIK

Support Vector Machines SVM

based on Elements Statistical Learning, ch. 4, 12

- generalizations of linear decision boundaries for classification
- will see later: SVM can produce *non-linear* boundaries by constructing a linear boundary in a large, transformed version of the feature space via kernels.
- The corresponding optimization problems are often solved via (variants of) gradient descent algorithms.

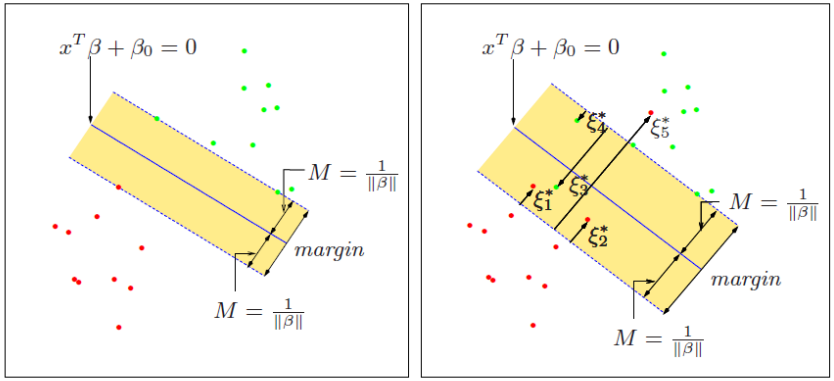


FIGURE 12.1. *Support vector classifiers. The left panel shows the separable case. The decision boundary is the solid line, while broken lines bound the shaded maximal margin of width $2M = 2/\|\beta\|$. The right panel shows the nonseparable (overlap) case. The points labeled ξ_j^* are on the wrong side of their margin by an amount $\xi_j^* = M\xi_j$; points on the correct side have $\xi_j^* = 0$. The margin is maximized subject to a total budget $\sum \xi_i \leq$ constant. Hence $\sum \xi_j^*$ is the total distance of points on the wrong side of their margin.*

An easy example

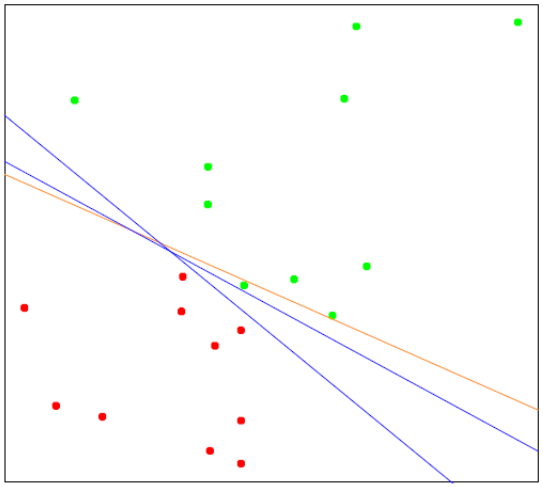


FIGURE 4.14. *A toy example with two classes separable by a hyperplane. The orange line is the least squares solution, which misclassifies one of the training points. Also shown are two blue separating hyperplanes found by the* perceptron learning algorithm *with different random starts.*

- 20 data points, two classes can be separated by linear boundary.
- blue lines: two of the infinitely many possible separating hyperplanes.
- orange line: least squares solution, obtained by regressing the $\{\pm 1\}$ response Y on X (with intercept); line given by $\{x \mid \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 = 0\}$.
- Orange line makes an error in separating the points.

The Linear Algebra Behind

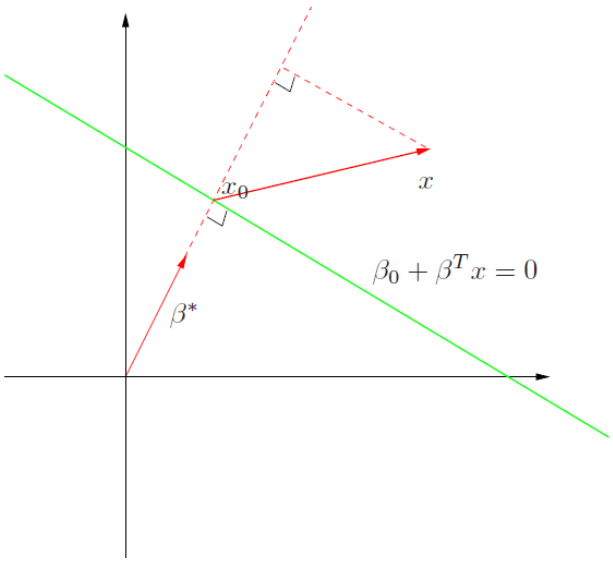


FIGURE 4.15. *The linear algebra of a hyperplane (affine set).*

green hyperplane L given by $f(x) = \beta_0 + \beta^\top x = 0$, with appropriately chosen β_0, β, i.e., a line in $\mathbb{R}^2$.
some properties:

- For any two points x_1 and x_2 lying in L, $\beta^\top(x_1 - x_2) = 0$, i.e., $x_1 - x_2$ is orthogonal to β. $x_1 - x_2$ lies in L, hence $\beta^\star = \frac{\beta}{||\beta||}$ is the vector normal to the surface of L.
- For any point x_0 in L, $\beta^\top x_0 = -\beta_0$.

The Linear Algebra Behind

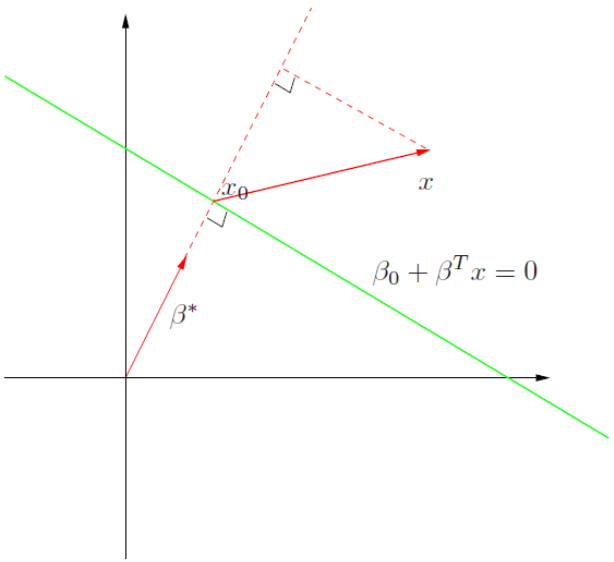


FIGURE 4.15. *The linear algebra of a hyperplane (affine set).*

in addition:

- The signed distance of any point x to L is given by $\beta^\star(x - x_0) = \frac{\beta}{||\beta||}(\beta^\top x + \beta_0) = \frac{1}{||f'(x)||} f(x)$. (see Hesse normal form linear algebra.)

Hence, the value $f(x)$ is proportional to the signed distance from x to the hyperplane $f(x) = 0$.

Optimal Separating Hyperplane for Classification

Standard Notation

- training data: N pairs $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$, with $x_i \in \mathbb{R}^p$, $y_i \in \{\pm 1\}$.
- hyperplane: $\{x \mid f(x) = x^\top \beta + \beta_0 = 0\}$
- β unit vector, i.e., $||\beta|| = 1$.
- We have just seen that $f(x)$ gives the signed distance from a point x to the hyperplane. Thus, if a new data point x comes in, the classification rule is

$$G(x) = \text{sign}(x^\top \beta + \beta_0).$$

Since the classes are separable, we can find a hyperplane

$$f(x) = x^\top \beta + \beta_0 \text{ with } y_i f(x_i) > 0 \; \forall i.$$

Next: We determine a hyperplane that creates the *biggest margin* between the training points for class 1 and -1 (see fig. next slide). Model this task as optimization problem

$$\max_{\beta, \beta_0, ||\beta||=1} M \quad \text{s.t. } y_i(x_i^\top \beta + \beta_0) \geq M, i = 1, \ldots, N \tag{1}$$

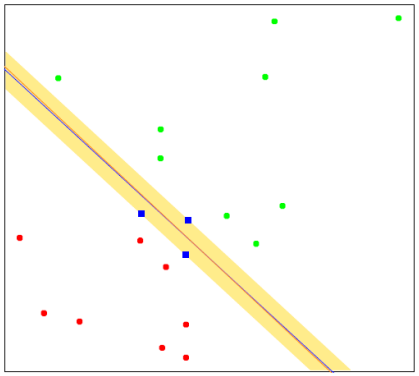


FIGURE 4.16. *The same data as in Figure 4.14. The shaded region delineates the maximum margin separating the two classes. There are three support points indicated, which lie on the boundary of the margin, and the optimal separating hyperplane (blue line) bisects the slab. Included in the figure is the boundary found using logistic regression (red line), which is very close to the optimal separating hyperplane (see Section 12.3.3).*

The band in the figure is M units away from the hyperplane on either side, and hence $2M$ units wide. It is called the *margin*.
Since for any β and β_0 satisfying these inequalities, any positively scaled multiple satisfies them too, we can arbitrarily set $||\beta|| = \frac{1}{M}$.

More convenient margin maximization problem

This leads to the following optimization problem instead of (1):

$$\min_{\beta,\beta_0,} \frac{1}{2}||\beta|| \quad \text{s.t. } y_i(x_i^\top \beta + \beta_0) \geq 1, i = 1, \ldots, N \tag{2}$$

The constraints define an empty slab or margin around the linear decision boundary of thickness $\frac{1}{||\beta||}$. By minimizing β and β_0, we maximize its thickness.

Lagrange Ansatz for Minimizing (2)

- In the linear regression problem that we have studied in the last chapter, optimality conditions were given by determine critical points where the gradient of the loss function w.r.t. the unknown parameters vanish.
- The SVM problem, however is a constrained optimization problem. We will nevertheless be able to write down optimality conditions.
- To this end, we remodel the margin maximization problem further such that we optimize an unconstrained problem.
- This can be achieved via the so-called *Lagrange function.*

Lagrange Ansatz for Minimizing (2)

- In the linear regression problem that we have studied in the last chapter, optimality conditions were given by determine critical points where the gradient of the loss function w.r.t. the unknown parameters vanish.
- The SVM problem, however is a constrained optimization problem. We will nevertheless be able to write down optimality conditions.
- To this end, we remodel the margin maximization problem further such that we optimize an unconstrained problem.
- This can be achieved via the so-called *Lagrange function.*

Lagrange Ansatz

Instead of (2), consider an unconstrained problem where the violation of the constraints is penalized in the objective. The Lagrange function models this: The Lagrange (primal) function, to be minimized w.r.t. β and β_0, is

$$L_P = \frac{1}{2}||\beta||^2 - \sum_{i=1}^{N} \alpha_i [y_i(x_i^\top \beta + \beta_0) - 1]$$

Karush-Kuhn Tucker conditions

Looking for critical points, we set the derivatives to zero and obtain:

$$\beta = \sum_{i=1}^{N} \alpha_i y_i x_i, 0 = \sum_{i=1}^{N} \alpha_i y_i \tag{3}$$

Substituting this in Lagrange function, we obtain

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k x_i^\top x_k, \alpha_i \geq 0$$

Now we need to use knowledge from non-linear optimization that is typically taught in bachelor analysis or calculus courses, namely the Karush-Kuhn Tucker conditions that are (first-order) optimality conditions for the Lagrange function. A solution has to satisfy the Karush-Kuhn Tucker conditions (3), $\alpha \geq 0$ and

$$\alpha_i [y_i (x_i^\top \beta + \beta_0) - 1] = 0 \forall i. \tag{4}$$

From this we can see

- if $\alpha_i > 0$, then $y_i(x_i^\top \beta + \beta_0) = 1$ or in other words, x_i is on the boundary of the slab;
- if $y_i(x_i^\top \beta + \beta_0) > 1$, x_i is not on the boundary of the slab, and $\alpha_i = 0$.

Geometrical Interpretation

We see that the solution vector β is defined in terms of a linear combination of the support points x_i. Those points defined to be on the boundary of the slab (margin) via $\alpha_i > 0$.

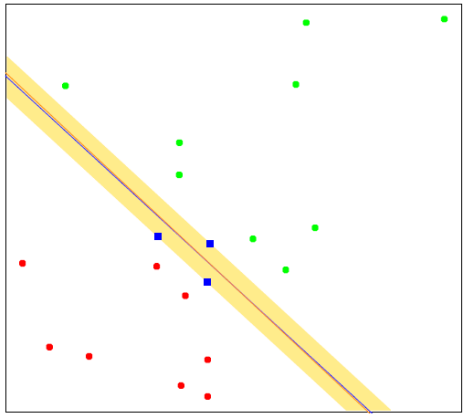


FIGURE 4.16. *The same data as in Figure 4.14. The shaded region delineates the maximum margin separating the two classes. There are three support points indicated, which lie on the boundary of the margin, and the optimal separating hyperplane (blue line) bisects the slab. Included in the figure is the boundary found using logistic regression (red line), which is very close to the optimal separating hyperplane (see Section 12.3.3).*

Figure shows the optimal separating hyperplane for our toy example; there are three support points. Likewise, β_0 is obtained by solving (4) for any of the support points.

Geometrical Interpretation

The optimal separating hyperplane produces a function $f(x) = x^\top \beta + \beta_0$ for classifying new observations: $G(x) = \text{sign} f(x)$

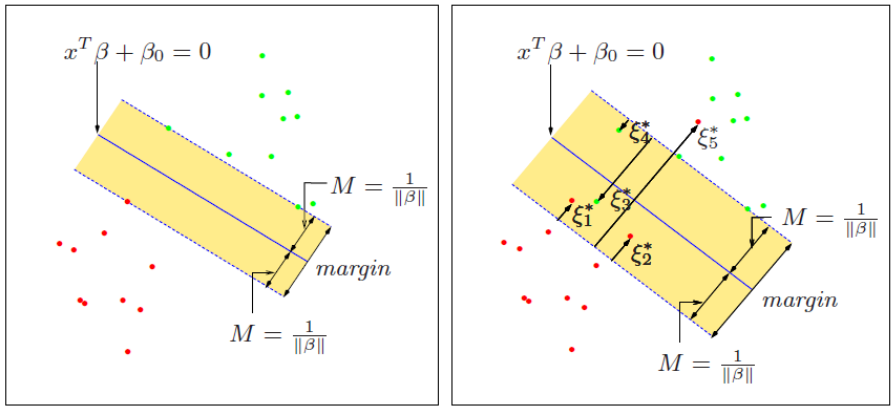


FIGURE 12.1. *Support vector classifiers. The left panel shows the separable case. The decision boundary is the solid line, while broken lines bound the shaded maximal margin of width $2M = 2/\|\beta\|$. The right panel shows the nonseparable (overlap) case. The points labeled ξ_j^* are on the wrong side of their margin by an amount $\xi_j^* = M\xi_j$; points on the correct side have $\xi_j^* = 0$. The margin is maximized subject to a total budget $\sum \xi_i \leq$ constant. Hence $\sum \xi_j^*$ is the total distance of points on the wrong side of their margin.*

Problem is convex (quadratic objective and only linear inequality constraints).

Geometrical Interpretation

The optimal separating hyperplane produces a function $f(x) = x^\top \beta + \beta_0$ for classifying new observations: $G(x) = \text{sign} f(x)$

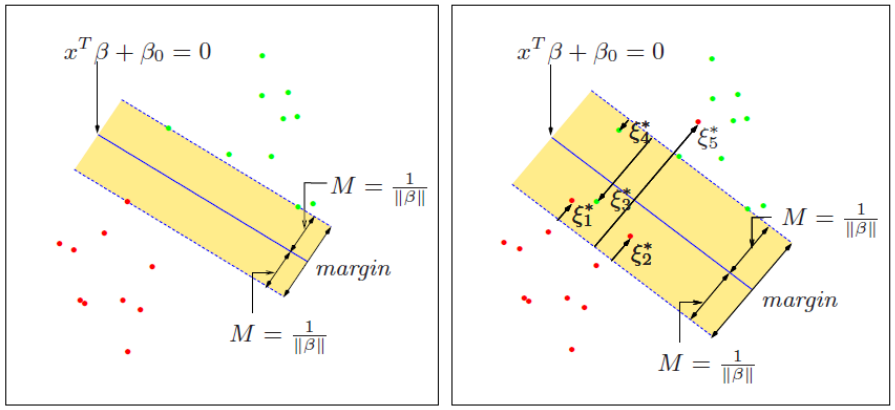


FIGURE 12.1. *Support vector classifiers. The left panel shows the separable case. The decision boundary is the solid line, while broken lines bound the shaded maximal margin of width $2M = 2/\|\beta\|$. The right panel shows the nonseparable (overlap) case. The points labeled ξ_j^* are on the wrong side of their margin by an amount $\xi_j^* = M\xi_j$; points on the correct side have $\xi_j^* = 0$. The margin is maximized subject to a total budget $\sum \xi_i \leq$ constant. Hence $\sum \xi_j^*$ is the total distance of points on the wrong side of their margin.*

Problem is convex (quadratic objective and only linear inequality constraints). Next, we consider a more realistic setting where the classes are allowed to overlap in feature space. One way to deal with the overlap is to still maximize M, but allow for some points to be on the wrong side of the margin.

SVM Model Allowing Misclassifications

(Linear) SVM optimization problem

Define slack variables $\xi = (\xi_1, \xi_2, \ldots, \xi_N)$ with $\forall i, \xi_i \geq 0, \sum_1^N \xi_i \leq$ constant and relax the constraints:

$$y_i(x_i^\top \beta + \beta_0) \geq M - \xi_i.$$

This measures overlap in actual distance from the margin. However, it will lead to a non-convex problem after dropping the norm constraint (see later).
An alternative model is:

$$y_i(x_i^\top \beta + \beta_0) \geq M(1 - \xi_i)$$

This constraint measures the overlap in relative distance, which changes with the width of the margin M.

This model is convex and is the *standard* support vector classifier, which we use from here on, as we see next:

Standard SVM Model allowing Misclassifications

- Its idea: value ξ_i in the constraint

$$y_i(x_i^\top \beta + \beta_0) \geq M(1 - \xi_i)$$

  is the proportional amount by which the prediction $f(x_i) = x_i^\top \beta + \beta_0$ is on the wrong side of its margin.
- By bounding the sum $\sum_{i=1}^N \xi_i$, we bound the total proportional amount by which predictions fall on the wrong side of their margin.
- Misclassifications occur when $\xi_i > 1$, so bounding $\sum_{i=1}^N \xi_i \leq K$ with some value K bounds the total number of training misclassifications at K.
- As in our simple example before, we can drop the norm constraint on β, define $M = \frac{1}{||\beta||}$ and write the model in the equivalent form

Standard (Linear) SVM Optimization Problem

$$\min_\beta ||\beta|| \quad \text{s.t. } y_i(x_i^\top \beta + \beta_0) \geq 1 - \xi_i, i = 1, \ldots, N, \xi_i \geq 0, \sum_{i=1}^N \xi_i \leq \text{const} \qquad (5)$$

(Please double-check...) This is convex!

Lagrange Ansatz for (5)

Again, we want to write down optimality conditions for solving the SVM optimization problem. We thus rewrite it as unconstrained problems via the Lagrange function again. Rewrite SVM optimization problem:

$$\min_{\beta} \frac{1}{2}||\beta||^2 + C\sum_{i=1}^{N}\xi \quad \text{s.t. } x_i \geq 0, y_i(x_i^\top \beta + \beta_0) \geq 1 - \xi_i, i = 1, \ldots, N, \xi_i \geq 0 \tag{6}$$

with appropriately chosen 'cost' parameter C that measures the impact for allowing misclassifications.
Again Lagrange Ansatz: instead of having constraints, consider an unconstrained problem where the violation of the constraints is penalized in the objective.

Lagrange function

The Lagrange (primal) function, to be minimized w.r.t. β and β_0, is

$$L_P = \frac{1}{2}||\beta||^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i[y_i(x_i^\top \beta + \beta_0) - (1 - \xi_i)] - \sum_{i=1}^{N}\mu_i\xi_i \tag{7}$$

KKT Conditions for SVM with Misclassifications

In order to minimize the Lagrange function w.r.t. β, β_0, ξ_i, we look for critical points and set the derivatives to zero.
We obtain:

$$\beta = \sum_{i=1}^{N} \alpha_i y_i x_i, 0 = \sum_{i=1}^{N} \alpha_i y_i, \alpha_i = C - \mu_i \tag{8}$$

together with $\alpha_i, \mu_i, \xi_i \geq 0$.
Substituting these equations (8) into the Lagrange function, we obtain the Lagrange dual

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k x_i^\top x_k, \alpha_i, \mu_i, \xi_i \geq 0 \tag{9}$$

L_D gives a lower bound on the objective function (7) for any feasible point.
The parameters α for relaxing the constraints into the Lagrangean objective function (7) need to be $\alpha \geq 0$.
In the KKT conditions, we have $\alpha_i = C_i - \mu_i$, where $\mu_i \geq 0$, thus $\alpha \leq C$.
We thus maximize L_D subject to $0 \leq \alpha_i \leq C$ and $\sum_{i=1}^{N} \alpha_i y_i = 0$.

KKT Conditions for SVM allowing misclassifications

In addition to (8), the Karush–Kuhn–Tucker conditions include the constraints

$$\alpha_i[y_i(x_i^\top \beta + \beta_0) - (1 - \xi_i)] = 0$$

$$\mu_i \xi_i = 0$$

$$y_i[(x_i^\top \beta + \beta_0) - (1 - \xi_i)] \geq 0, i = 1, \ldots, N$$

that together with (8) uniquely characterize the solution to the primal and dual problem.

This is where the name Support Vector comes from...

From the KKT conditions, we see that the solution for beta has the form

$$\beta = \sum_{i=1}^{N} \alpha_i y_i x_i$$

.

The coefficients α_i are only non-zero for those observations *i* for which the constraints

$$y_i(x_i^\top \beta + \beta_0) - (1 - \xi_i)] \geq 0$$

are exactly met with equality (due to $\alpha_i[y_i(x_i^\top \beta + \beta_0) - (1 - \xi_i)] = 0$)
These observations are called the *support vectors*, since β is represented in terms of them alone. Among these support points, some will lie on the edge of the margin $\xi_i = 0$, and will be characterized by $0 < \alpha_i < C$; the remainder ($\xi_i > 0$) have $\alpha_i = C$.
From $\alpha_i[y_i(x_i^\top \beta + \beta_0) - (1 - \xi_i)] = 0$ we can see that any of these margin points ($0 < \alpha_i, \xi_i = 0$) can be used to solve for β_0, and we typically use an average of all the solutions for numerical stability.

SVM Classification

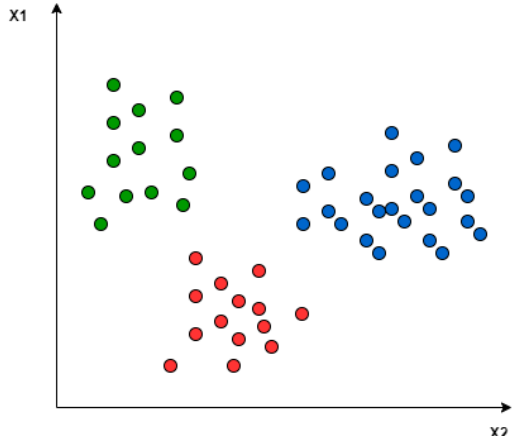
Given the solutions β_0 and β, the classification prediction for a data point x is

$$G(x) = \text{sign}[f(x)] = \text{sign}[x^\top \beta + \beta_0]$$

Multiclass Classification Using SVM

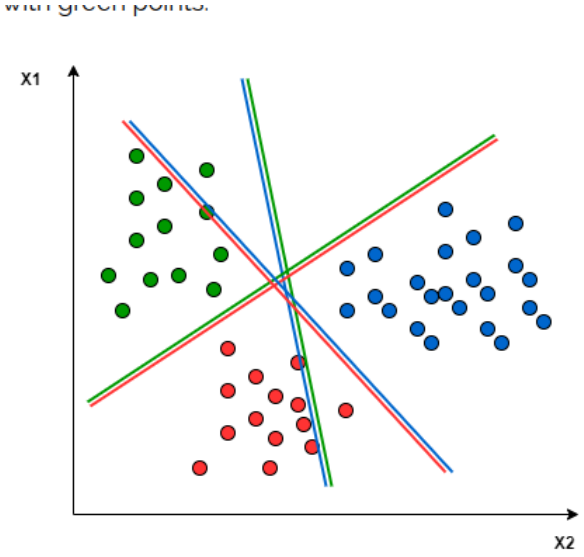
For multiclass classification, the same principle is utilized after breaking down the multiclassification problem into multiple binary classification problems.
The idea is to map data points to high dimensional space to gain mutual linear separation between every two classes.
One-to-One approach: breaks down the multiclass problem into multiple binary classification problems. A binary classifier per each pair of classes. The classifier can use $\frac{m(m-1)}{2}$ SVMs.
One-to-Rest: The breakdown is set to a binary classifier per each class. The classifier can use m SVMs. Each SVM would predict membership in one of the m classes.

Visual Example with 3 Classes



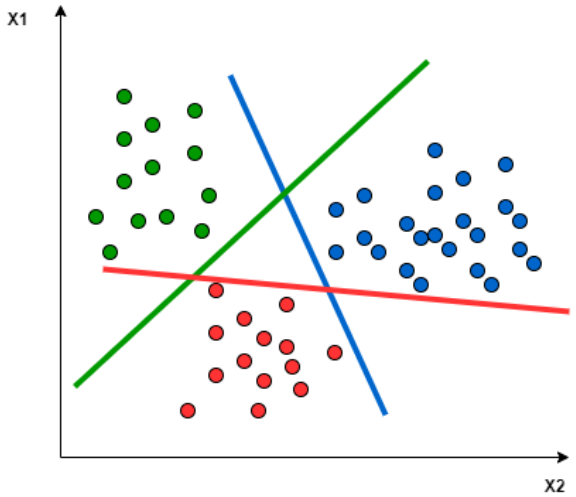
One-to-One Approach

We need a hyperplane to separate between every two classes, neglecting the points of the third class. This means the separation takes into account only the points of the two classes in the current split. For example, the red-blue line tries to maximize the separation only between blue and red points. It has nothing to do with green points:



One-to-Rest approach

We need a hyperplane to separate between a class and all others at once. This means the separation takes all points into account, dividing them into two groups; a group for the class points and a group for all other points. For example, the green line tries to maximize the separation between green points and all other points at once:



Non-Linear SVM: SVM with Kernels

- SVM as we have described it finds linear boundaries in input feature space
- similar to other methods (cmp what we did for, e.g., linear regression, (kernel) PCA, earlier): enlarge feature space using basis expansions such as polynomials or splines
- typically this leads to better separation in training data that is linear in an enlarged space. This corresponds to non-linear separation in original space.

approach:

- select basis functions (e.g., polynomials) $h_m(x), m = 1, \ldots, M$
- determine data points $h(x_i) = (h_1(x_i), h_2(x_i), \ldots, h_M(x_i)), i = 1, \ldots, N$
- produce (the now non-linear) function $f(x) = h(x)^\top \beta + \beta_0$.
- the classifier is $G(x) = \text{sign}(f(x))$ as before.

SVM with Kernels for Classification

We will write down the SVM optimization problem such that it only contains inner products of the kernels, denoted by $\langle ., . \rangle$ of the input features, and not the kernels itself.
Recall Lagrange dual function (9) that we express here with kernel functions:

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k \langle h(x_i), h(x_k) \rangle. \tag{10}$$

From (8) we see that the solution function $f(x)$ can be written as

$$f(x) = h(x)^\top \beta + \beta_0 = \sum_{i=1}^{N} \alpha_i y_i \langle h(x), h(x_i) \rangle + \beta_0 \tag{11}$$

As before, given α_i, then β_0 can be determined by solving for $y_i f(x_i) = 1$ in (11) for any (or all) x_i for which $0 < \alpha_i < C$.

SVM with Kernels

Thus, both the Lagrangean and also (11) involve $h(x)$ only via inner products (compare this with the similar situation we had in kernel PCA) Thus, we do not need to specify the functions $h(x)$ themselves, but only the kernel functions

$$K(x, x') = \langle h(x), h(x') \rangle$$

that computes the inner product in the transformed space. K should be a symmetric positive function.
Popular choices:

- polynomial of degree d: $K(x, x') = (1 + \langle x, x' \rangle)^d$
- radial basis: $K(x, x') = \exp(\gamma ||x - x'||^2)$, with parameter γ
- neural network: $K(x, x') = \tanh(\kappa_1)\langle x, x' \rangle + \kappa_2$, with parameters κ_1, κ_2.

Illustration

(from ESL chp.12)