CS 181 Spring 2017 Section 5
Margin-Based Classification, SVMs
Solution

1 Motivation

The idea for Support Vector Machines is that, for all the linear hyperplanes that exist, we want one that will create the largest distance, or "margin", with the training data. Larger margins tend to improve generalization error. To define the margin, we consider a hyperplane of the form

$$\mathbf{w}^\top \mathbf{x} + w_0 = 0$$

What is the perpendicular distance from an example $\mathbf{x}_i$ to the decision boundary $\mathbf{w}^\top \mathbf{x} + w_0 = 0$?.

For two points $\mathbf{x}_1$ and $\mathbf{x}_2$ on the hyperplane, consider the projection with $\mathbf{w}$:

$$\mathbf{w}^\top(\mathbf{x}_1 - \mathbf{x}_2) = \mathbf{w}^\top \mathbf{x}_1 - \mathbf{w}^\top \mathbf{x}_2 = -w_0 - (-w_0) = 0$$

Therefore, $\mathbf{w}$ is orthogonal to the hyperplane. So to get the distance from a hyperplane and an arbitrary example $\mathbf{x}$, we just need the length in the direction of $\mathbf{w}$ between the point and the hyperplane. We let r signify the distance between a point and the hyperplane. Then $\mathbf{x}_\perp$ is the projection of the point onto the hyperplane, so that we can decompose a point $\mathbf{x}$ as

$$\mathbf{x}_\perp + r\frac{\mathbf{w}}{||\mathbf{w}||} = \mathbf{x}$$

Left multiply by $\mathbf{w}^\top$:

$$\mathbf{w}^\top \mathbf{x}_\perp + r\frac{\mathbf{w}^\top \mathbf{w}}{||\mathbf{w}||} = \mathbf{w}^\top \mathbf{x} \Rightarrow r = \frac{\mathbf{w}^\top \mathbf{x} + w_0}{||\mathbf{w}||}$$

Scalar r then gives the signed, normalized distance between a point and the hyperplane. For correctly classified data, we have $y_i = +1$ when this distance is positive and $y_i = -1$ when it is negative. Based on this, we can obtain a positive distance for both kinds of examples by multiplying by y_i. We define the margin of the dataset as the minimum such distance over all examples:

$$\min_i \frac{y_i(\mathbf{w}^\top \mathbf{x}_i + w_0)}{||\mathbf{w}||}$$

What is the training problem for support vector machines?

We want the $\mathbf{w}$ and w_0 that maximize the margin:

$$\arg\max_{\mathbf{w}, w_0} \frac{1}{||\mathbf{w}||} \min_i y_i(\mathbf{w}^\top \mathbf{x}_i + w_0)$$

In the hard-margin training problem, we know that the data is linearly separable and therefore any margin (including the optimal) must be greater than 0:

$$\min_i \frac{y_i(\mathbf{w}^\top \mathbf{x}_i + w_0)}{||\mathbf{w}||} > 0$$

We can observe that $\mathbf{w}$ and w_0 are invariant to changes of scale. Because of this, it is without loss of generality to impose $\min_i \frac{y_i(\mathbf{w}^\top \mathbf{x}_i + w_0)}{||\mathbf{w}||} > 1$. This lets us write the optimization problem as:

$$\arg\max_{\mathbf{w}, w_0} \frac{1}{||\mathbf{w}||} \quad \text{s.t. } \forall i \; y_i(\mathbf{w}^\top \mathbf{x}_i + w_0) \geq 1$$

What is the corresponding minimization problem for hard-margin training for SVMs? Explain at a high level why the minimization problem for SVMs is equivalent to the max-margin problem.

We can invert $\mathbf{w}$ to change the max to a min:

$$\arg\min_{\mathbf{w}, w_0} \frac{1}{2}||\mathbf{w}||^2 \quad \text{s.t. } \forall i \; y_i(\mathbf{w}^\top \mathbf{x}_i + w_0) \geq 1$$

Informally, this is the same as max-margin training because the constraint is binding for the examples closest to the decision boundary. For these examples we have $y_i(\mathbf{w}^\top \mathbf{x}_i + w_0) = 1$. The distance on these examples is $1/||\mathbf{w}||$, and is maximized by minimizing $||\mathbf{w}||^2$.

2 Soft Margin Formulation

For the hard margin formulation, we have been assuming that the data is linearly separable. However, this is not always true, and even if the data is linearly separable, it may not be best to find a separating hyperplane. In optimizing generalization error, there is a tradeoff between the size of the margin and the number of mistakes on the training data.

For the soft margin formulation, we introduce a slack variable $\xi_i \geq 0$ for each i to relax the constraints on each example.

$$\xi_i \begin{cases} = 0 & \text{if correctly classified} \\ \in (0, 1] & \text{correctly classified but inside margin region} \\ > 1 & \text{if incorrectly classified} \end{cases}$$

We can now rewrite the training problem for a soft margin formulation to be

$$\arg\min_{\mathbf{w}, w_0} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n} \xi_i$$

$$\text{s.t. } \forall i \; y_i(\mathbf{w}^\top \mathbf{x}_i + w_0) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

We add a regularization parameter C, that controls how much we penalize violating the hard margin constraints. A large C penalizes these violations and thus "respects" the data closely and has small regularization. A small C does not penalize the sum of slack variables as heavily, relaxing the constraint. This is increasing the regularization.

3 Dual Form of the Support Vector Machine Training Problem

Use Lagrange multipliers to convert the new minimization problem into dual form.

Our original hard-margin training problem, which looks for weights that maximize the margin on the training data, is

$$\mathbf{w}^\star, w_0^\star = \arg\min_{\mathbf{w},w_0} \frac{1}{2}||\mathbf{w}||^2 \quad \text{s.t.} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + w_0) \geq 1, \quad \forall i \in \{1,\ldots,n\}\,. \tag{1}$$

We introduce Lagrange multipliers, $\alpha_1, \ldots, \alpha_n \geq 0$, one for each inequality in Equation 1, i.e., one per example, to obtain the Lagrangian function (Bishop appendix E does a good job of explaining Lagrange multipliers):

$$L(\mathbf{w}, w_0, \boldsymbol{\alpha}) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{n} \alpha_i (y_i(\mathbf{w}^\top \mathbf{x}_i + w_0) - 1) \tag{2}$$

Based on this, we now equivalently solve:

$$\mathbf{w}^*, w_0^* = \arg\min_{\mathbf{w},w_0} \max_{\boldsymbol{\alpha}\geq 0} L(\mathbf{w}, w_0, \boldsymbol{\alpha})$$

By strong duality (out of scope!), we can equivalently write this as:

$$\max_{\boldsymbol{\alpha}\geq 0} \min_{\mathbf{w},w_0} L(\mathbf{w}, w_0, \boldsymbol{\alpha})$$

This is useful because we can now solve analytically for the $\min_{\mathbf{w},w_0}[\cdot]$ part of this expression. Taking derivatives, we see:

$$\frac{\partial L}{\partial w_j} = w_j - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = 0 \Rightarrow \mathbf{w}^* = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

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

Plugging these optimal values into the Lagrangian function, we get (you should understand but not memorize this):

$$\begin{aligned} L(\mathbf{w}, w_0, \boldsymbol{\alpha}) &= \frac{1}{2}||\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i||^2 - \sum_{i=1}^{n} \alpha_i y_i (\sum_{i'=1}^{n} \alpha_{i'} y_{i'} \mathbf{x}_{i'})^\top \mathbf{x}_i - \sum_{i=1}^{n} \alpha_i y_i w_0 + \sum_{i=1}^{n} \alpha_i \\ &= \frac{1}{2} \sum_{i=1}^{n}\sum_{i'=1}^{n} \alpha_i \alpha_{i'} y_i y_{i'} \mathbf{x}_i^\top \mathbf{x}_{i'} - \sum_{i=1}^{n}\sum_{i'=1}^{n} \alpha_i \alpha_{i'} y_i y_{i'} \mathbf{x}_i^\top \mathbf{x}_{i'} - w_0 \sum_{i=1}^{n} \alpha_i y_i + \sum_{i=1}^{n} \alpha_i \\ &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n}\sum_{i'=1}^{n} \alpha_i \alpha_{i'} y_i y_{i'} \mathbf{x}_i^\top \mathbf{x}_{i'} \end{aligned}$$

We then solve for $\boldsymbol{\alpha}$, maximizing L, subject to $\boldsymbol{\alpha} \geq 0$. The support vectors are defined to be $Q = \{i : \alpha_i > 0\}$. For the soft-margin variation, these include examples that lie in the margin region in addition to on the margin boundary.

4 Why bother with the dual form? *very important*

The dual form is useful because the number of variables to maximize is linear in n (one for each training example), and does not depend on the dimension of the feature space! Moreover, there tends to be a small number of support vectors (data points that define the boundary) and thus the trained classifier can be interpretable.

To classify a new example $\mathbf{x}$, we compute

$$\sum_{i=1}^{n} \alpha_i^\star \, y_i \mathbf{x}_i^\top \mathbf{x} + w_0^\star$$

where $\alpha^\star$ and $w_0^\star$ solve the training problem. Based on this, we classify the example as $+1$ if this discriminant value is > 0, and -1 otherwise.
In addition, the dual has the very nice property that if we use a basis function to map $\mathbf{x}$ to a higher dimensional space, this only comes in through the "kernel function." The training problem is as explained earlier, except where $\mathbf{x}_i^\top \mathbf{x}_{i'}$ appears, we use

$$K(\mathbf{x}_i, \mathbf{x}_{i'}) = \boldsymbol{\phi}(\mathbf{x}_i)^\top \boldsymbol{\phi}(\mathbf{x}_{i'})$$

in its place. Similarly, we classify a new example $\mathbf{x}$ based on the value of discriminant $\sum_{i=1}^{n} \alpha_i^\star \, y_i K(\mathbf{x}_i, \mathbf{x}) + w_0^\star$. The reason that this is interesting is because we can directly compute the dot product $\boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}')$ without projecting to the higher-dimensional space! This is known as the "kernel trick." As long as $K()$ is a valid kernel (see practice question below) the dual training problem can be solved without actually computing $\boldsymbol{\phi}$.

5 Practice Problems (cover some but not all at section)

1. **Removing Support Vectors and Retraining (Berkeley, Fall '11)**

   Suppose that we train two SVMs, the first containing all of the training data and the second trained on a data set constructed by removing some of the support vectors from the first training set. How does the size of the optimal margin change between the first and second training data? What is a downside to doing this?

   If we remove some of the examples on the margin (the examples that are closest to the best separator), then we can either leave the size of the optimal margin the same or we can increase the size of the optimal margin. This is because we have removed the difficult examples, and thus we can either leave the optimal solution unchanged or make it "better" (corresponding to increasing the size of the optimal margin). A potential downside is that we might see equally difficult points at test time and mis-classify them.

2. **Proof that margin is invariant to scalar multiplication**

   In reformulating our max margin
   $$\frac{y_i(\mathbf{w}^\top \mathbf{x}_i + w_0)}{||\mathbf{w}||}$$
   training problem, we use the fact that the margin is invariant to multiplying $(\mathbf{w}, w_0)$ by any $\beta > 0$. Show that this property is true.

   $$\frac{y_i((\beta\mathbf{w})^\top \mathbf{x}_i + (\beta w_0))}{||(\beta\mathbf{w})||} = \frac{\beta}{\beta} \cdot \frac{y_i(\mathbf{w}^\top \mathbf{x}_i + w_0)}{||\mathbf{w}||} = \frac{y_i(\mathbf{w}^\top \mathbf{x}_i + w_0)}{||\mathbf{w}||}$$

3. **(Berkeley, Fall '11)**

   Consider $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^2$. Show that the polynomial kernel of degree 2, $K(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^\top \mathbf{x}')^2$ is equivalent to a dot product $\boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}')$ where $\boldsymbol{\phi}(\mathbf{x}) = (x_1^2, x_2^2, x_1, x_2, x_1 x_2, 1)$.

   The dot product of the features $\boldsymbol{\phi}(\mathbf{x})$ and $\boldsymbol{\phi}(\mathbf{x}')$ is $x_1^2 {x_1'}^2 + x_2^2 {x_2'}^2 + x_1 x_1' + x_2 x_2' + x_1 x_2 x_1' x_2' + 1$. The kernel function gives:

   $$\begin{aligned} K(\mathbf{x}, \mathbf{x}') &= (1 + \mathbf{x}^\top \mathbf{x}')^2 \\ &= (1 + x_1 x_1' + x_2 x_2')^2 \\ &= 1 + 2x_1 x_1' + 2x_2 x_2' + 2x_1 x_1' x_2 x_2' + x_1^2 {x_1'}^2 + x_2^2 {x_2'}^2 \end{aligned}$$

   So the dot product and kernel function give the same terms (up to scalar multiples in terms), meaning they are equivalent.

4. **Composing Kernels**

A key benefit of SVM training is the ability to use kernel functions $K(\mathbf{x}, \mathbf{x}')$ as opposed to explicit basis functions $\boldsymbol{\phi}(\mathbf{x})$. Kernels make it possible to implicitly express large or even infinite dimensional basis features. We do this by computing $\boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}')$ directly, without ever computing $\boldsymbol{\phi}(\mathbf{x})$.

When training SVMs, we begin by computing the kernel matrix $\mathbf{K}$, over our training data $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. The kernel matrix, defined as $K_{i,i'} = K(\mathbf{x}_i, \mathbf{x}_{i'})$, expresses the kernel function applied between all pairs of training points.

In class, we saw Mercer's theorem (maybe), which tells us that any function K that yields a positive semi-definite kernel matrix forms a valid kernel, i.e. corresponds to a matrix of dot-products under *some* basis $\boldsymbol{\phi}$. Therefore instead of using an explicit basis, we can build kernel functions directly that fulfill this property.

A particularly nice coralary of this theorem is that it allows us to build more expressive kernels by composition. In this problem, you are tasked with using Mercer's theorem and the definition of a kernel matrix to prove that the following compositions are valid kernels, assuming $K^{(1)}$ and $K^{(2)}$ are valid kernels. Recall that a positive semi-definite matrix $\mathbf{K}$ requires $\mathbf{z}^\top \mathbf{K} \mathbf{z} \geq 0, \ \forall \ \mathbf{z} \in \mathbb{R}^n$.

(a) $K(\mathbf{x}, \mathbf{x}') = c\, K^{(1)}(\mathbf{x}, \mathbf{x}') \quad$ for $c > 0$

(b) $K(\mathbf{x}, \mathbf{x}') = K^{(1)}(\mathbf{x}, \mathbf{x}') + K^{(2)}(\mathbf{x}, \mathbf{x}')$

(c) $K(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})\, K^{(1)}(\mathbf{x}, \mathbf{x}')\, f(\mathbf{x}') \quad$ where f is any function from $\mathbb{R}^m$ to $\mathbb{R}$

(d) $K(\mathbf{x}, \mathbf{x}') = K^{(1)}(\mathbf{x}, \mathbf{x}')\, K^{(2)}(\mathbf{x}, \mathbf{x}')$
[Hint: Use the property that for any $\boldsymbol{\phi}(\mathbf{x})$, $K(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}')$ forms a positive semi-definite kernel matrix.]

(e) i. The $\exp$ function can be written as,

$$\exp(x) = \lim_{i \to \infty} \left(1 + x + \cdots + \frac{x^i}{i!}\right).$$

Use this to show that $\exp(xx')$ (here $x, x' \in \mathbb{R}$)) can be written as $\boldsymbol{\phi}(x)^\top \boldsymbol{\phi}(x')$ for some basis function $\boldsymbol{\phi}(x)$. Derive this basis function, and explain why this would be hard to use as a basis in standard logistic regression.

ii. Using the previous identities, show that $K(\mathbf{x}, \mathbf{x}') = \exp(K^{(1)}(\mathbf{x}, \mathbf{x}'))$ is a valid kernel.

(f) Finally use this analysis and previous identities to prove the validity of the Gaussian kernel:

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(\frac{-||\mathbf{x} - \mathbf{x}'||_2^2}{2\sigma^2}\right)$$

For parts 1-5, it suffices to show that the kernel is valid either by finding a particular $\boldsymbol{\phi}(\mathbf{x})$ that produces it, or by showing that the kernel matrix is positive semi-definite. In these solutions, let $\boldsymbol{\phi}^{(i)}(\mathbf{x})$ and $\mathbf{K}^{(i)}$ denote the underlying basis function and kernel matrix for kernel $K^{(i)}$, respectively.

(a) We have the kernel matrix $\mathbf{K} = c\mathbf{K}^{(1)}$. We need $\mathbf{v}^\top \mathbf{K}\mathbf{v} = c\mathbf{v}^\top \mathbf{K}^{(1)}\mathbf{v} \geq 0$, which we know to be true because $\mathbf{K}^{(1)}$ is positive semi-definite and $c > 0$.
Alternatively, take $\boldsymbol{\phi}(\mathbf{x}) = \sqrt{c}\,\boldsymbol{\phi}^{(1)}(\mathbf{x})$.

(b) We have the kernel matrix $\mathbf{K} = \mathbf{K}^{(1)} + \mathbf{K}^{(2)}$. We need

$$\mathbf{v}^\top \mathbf{K}\mathbf{v} = \mathbf{v}^\top(\mathbf{K}^{(1)} + \mathbf{K}^{(2)})\mathbf{v} = \mathbf{v}^\top \mathbf{K}^{(1)}\mathbf{v} + \mathbf{v}^\top \mathbf{K}^{(2)}\mathbf{v} \geq 0,$$

which we know to be true because $\mathbf{K}^{(1)}, \mathbf{K}^{(2)}$ are positive semi-definite.
Alternatively, take $\boldsymbol{\phi}(\mathbf{x}) = \left[\phi_1^{(1)}(\mathbf{x}), \ldots, \phi_d^{(1)}(\mathbf{x}), \phi_1^{(2)}(\mathbf{x}), \ldots, \phi_d^{(2)}(\mathbf{x})\right]^\top$, the concatenation of $\boldsymbol{\phi}^{(1)}(\mathbf{x}), \boldsymbol{\phi}^{(2)}(\mathbf{x})$

(c) Take $\boldsymbol{\phi}(\mathbf{x}) = f(\mathbf{x})\boldsymbol{\phi}^{(1)}(\mathbf{x})$.

(d) Take $\boldsymbol{\phi}(\mathbf{x}) = \left[\phi_1^{(1)}(\mathbf{x})\phi_1^{(2)}(\mathbf{x}), \ldots, \phi_1^{(1)}(\mathbf{x})\phi_d^{(2)}(\mathbf{x}), \phi_2^{(1)}(\mathbf{x})\phi_1^{(2)}(\mathbf{x}), \ldots, \phi_d^{(1)}(\mathbf{x})\phi_d^{(2)}(\mathbf{x})\right]^\top$, the flattened vector for outer product $\boldsymbol{\phi}^{(1)}(\mathbf{x}) \otimes \boldsymbol{\phi}^{(2)}(\mathbf{x})$. The order of terms does not matter.

(e) i. Using the Taylor series expansion, we see that $\exp(xx') = \lim_{i\to\infty}\left(1 + xx' + \cdots + \frac{(xx')^i}{i!}\right)$. Take

$$\boldsymbol{\phi}(x) = \left[1, x, \ldots, \frac{x^i}{\sqrt{i!}}, \ldots\right]^\top.$$

This is infeasible because we have an infinite basis!
Note: don't forget the square root on the factorial in the denominator.

ii. We have

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(K^{(1)}(\mathbf{x}, \mathbf{x}')\right) = \exp\left(\boldsymbol{\phi}^{(1)}(\mathbf{x})^\top \boldsymbol{\phi}^{(1)}(\mathbf{x}')\right) = \prod_{i=1}^{d} \exp\left(\phi_i^{(1)}(\mathbf{x})\phi_i^{(1)}(\mathbf{x}')\right)$$

From here, we recognize the multiplicand to be a valid kernel, using our result from part (a). Then, recognize that a product of valid kernels is a valid kernel, per our results in part 4. Alternatively, write

$$K(\mathbf{x}, \mathbf{x}') = \exp(K^{(1)}(\mathbf{x}, \mathbf{x}')) = \lim_{i\to\infty}\left(1 + \boldsymbol{\phi}^{(1)}(\mathbf{x})^\top \boldsymbol{\phi}^{(1)}(\mathbf{x}') + \cdots + \frac{(\boldsymbol{\phi}^{(1)}(\mathbf{x})^\top \boldsymbol{\phi}^{(1)}(\mathbf{x}'))^i}{i!}\right)$$

and use properties 1, 2, and 4 to recognize that this is a valid kernel.

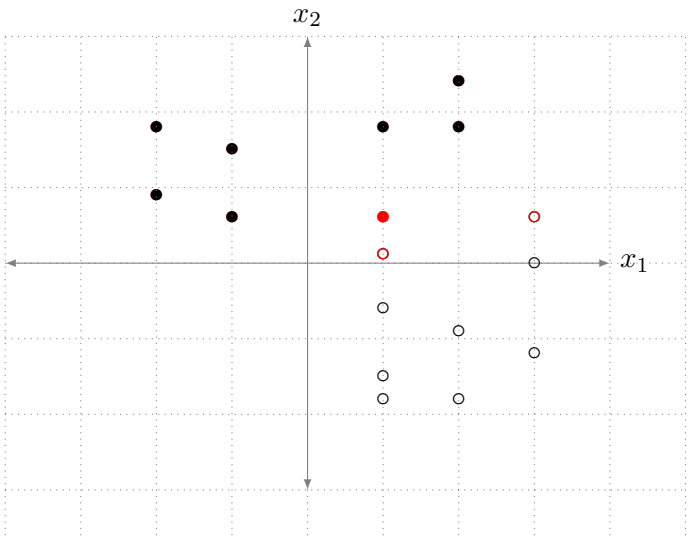
(f)
- $K_0(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}'$ is a valid kernel by definition of the kernel (it is the inner product of $\mathbf{x}$ and $\mathbf{x}'$).
- Thus $K_1(\mathbf{x}, \mathbf{x}') = \exp(2\mathbf{x}^\top \mathbf{x}')$ is also a valid kernel by property 1 and 5.
- Note that $K(\mathbf{x}, \mathbf{x}') = \exp(-\mathbf{x}^\top \mathbf{x})\exp(2\mathbf{x}^\top \mathbf{x}')\exp(-\mathbf{x}'^\top \mathbf{x}') = f(\mathbf{x})K_1 f(\mathbf{x}')$, where $f(\mathbf{x}) = \exp(-\mathbf{x}^\top \mathbf{x})$.
- Hence using property 3 in the last step, we proved $K(\mathbf{x}, \mathbf{x}')$ is a kernel.

Note: a common mistake is saying $\exp(-\mathbf{x}^\top \mathbf{x})$ is a kernel. It is not.

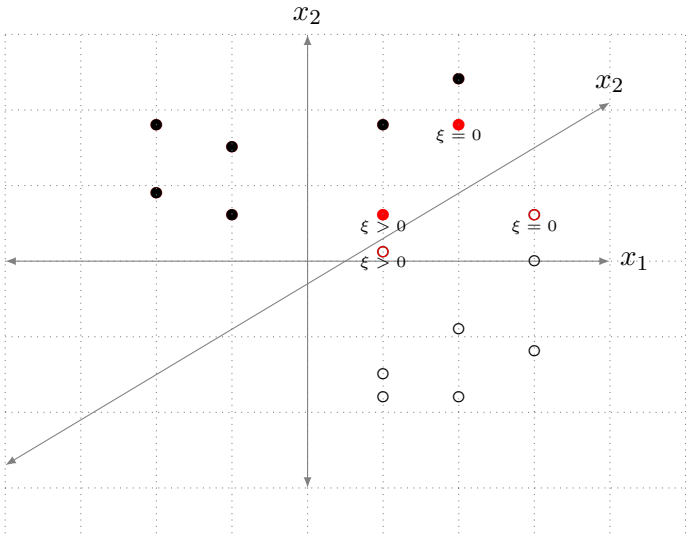
5. **Draw Margin Boundary**

In the figures below, the red examples represent the support vectors. All other examples can be assumed to have $\alpha_i = 0$. Draw the margins for the boundary given this information.
i) For the first example, you can assume the hard margin formulation. Draw the decision boundary as well as the two margin boundaries given the support vector.
ii) For the second example, you can assume the soft margin formulation and that all points are correctly classified with the optimal decision boundary. The decision boundary is already given. Draw the two margin boundaries given the support vector.

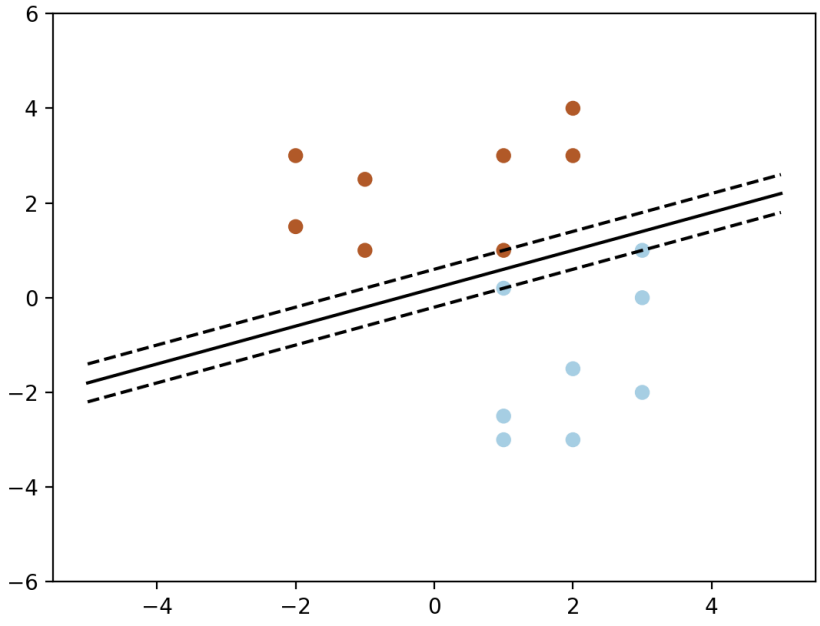
i)



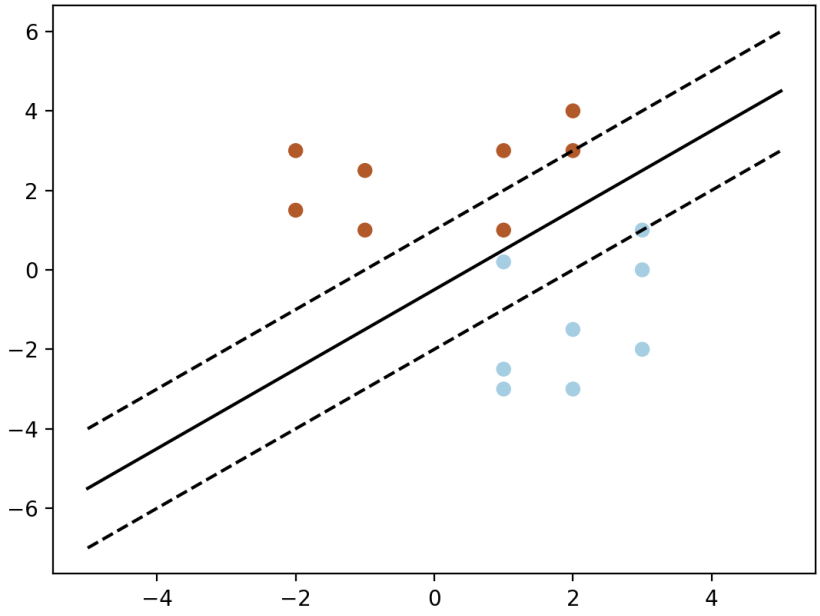
ii)



i)



ii)



6. **String Kernel**

Let $\mathbf{s}$ and $\mathbf{s}'$ be strings. To measure how similar $\mathbf{s}$ and $\mathbf{s}'$ are, consider the "string kernel" $K(\mathbf{s}, \mathbf{s}')$, which returns the total number of distinct substrings (of any length) that $\mathbf{s}$ and $\mathbf{s}'$ have in common. For example, $K(\texttt{'aa'}, \texttt{'aab'}) = 3$ because the substrings `''`, `'a'`, and `'aa'` are in common.
(i) Compute $K(\texttt{'aza'}, \texttt{'zaz'})$.
(ii) What is the number of possible substrings of length 1, 2, and 3 in strings that are composed from a 26-letter alphabet?
(iii) Suppose we wanted to project a string into a higher-dimensional space such that we could represent via a 0 or 1 each of all possible substrings of length ≤ 3. How many dimensions would we need?

(i) $K(\texttt{'aza'}, \texttt{'zaz'}) = 5$ because substrings `''`, `'a'`, `'z'`, `'az'`, `'za'` are in common.

(ii) There are $26^1 = 26$ possible substrings of length 1, $26^2 = 676$ of length 2, and $26^3 = 17576$ of length 3.

(iii) Then $26 + 676 + 17576 = 18278$ features are required to represent all substrings of length ≤ 3.

Note that in computing the kernel, we don't have to compute a feature representation for the data points (i.e. we don't have to find the presence/absence of each possible substring for $\mathbf{s}$ and $\mathbf{s}'$). Instead we can just write a program to find only the substrings that are in common. We avoid having to use costly representations to calculate the similarity between strings. This is the advantage of using kernel functions.