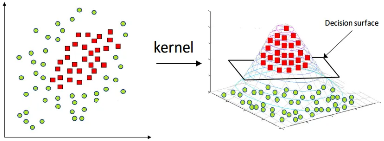
Kernel: Definition

- Consider a transformation $\Phi : \mathcal{X} \to \mathcal{Z}$
  - E.g., transform features in space $\mathcal{X}$ non-linearly into higher-dimensional space $\mathcal{Z}$
- **Kernel of transformation** Φ yields inner product of two points $\underline{\boldsymbol{x}}, \underline{\boldsymbol{x}}' \in \mathcal{X}$ in transformed space $\mathcal{Z}$

$$K_\Phi(\underline{\boldsymbol{x}}, \underline{\boldsymbol{x}}') \triangleq \langle \Phi(\underline{\boldsymbol{x}}), \Phi(\underline{\boldsymbol{x}}') \rangle = \Phi(\underline{\boldsymbol{x}})^T \Phi(\underline{\boldsymbol{x}}') = \underline{\boldsymbol{z}}^T \underline{\boldsymbol{z}}'$$

- Why doing this?



SCIENCE ACADEMY

Kernel: Expression From the Transform

- If you have an expression for Φ, compute a closed formula for the kernel
- E.g., if transformation is $\Phi : \mathbb{R}^2 \to \mathbb{R}^6$, it introduces interaction terms:

$$\underline{\boldsymbol{z}} = \Phi(\underline{\boldsymbol{x}}) = \Phi(x_1, x_2) = (1, x_1, x_2, x_1^2, x_2^2, x_1 x_2)$$

- Kernel of Φ is:

$$\begin{aligned} K_\Phi(\underline{\boldsymbol{x}}, \underline{\boldsymbol{x}}') &= (1, x_1, x_2, x_1^2, x_2^2, x_1 x_2)^T (1, x'_1, x'_2, x'^2_1, x'^2_2, x'_1 x'_2) \\ &= 1 + x_1 x'_1 + x_2 x'_2 + x_1^2 x'^2_1 + x_2^2 x'^2_2 + x_1 x_2 x'_1 x'_2 \end{aligned}$$

SCIENCE ACADEMY

Gaussian Kernel

- Aka "exponential kernel" or "Radial Basis Function" (RBF) kernel
- A **Gaussian kernel** has the form:

$$K(\underline{\boldsymbol{x}}, \underline{\boldsymbol{x}}') = \exp(-\gamma \|\underline{\boldsymbol{x}} - \underline{\boldsymbol{x}}'\|^2) = \exp(-\frac{\|\underline{\boldsymbol{x}} - \underline{\boldsymbol{x}}'\|^2}{\sigma^2})$$

- It can be shown to be an inner product in an infinite dimension $\mathcal{Z}$

Kernel as Way to Measure Similarity

- **Intuition**: The Gaussian kernel

$$K(\underline{\boldsymbol{x}}, \underline{\boldsymbol{x}}') = \exp(-\gamma \|\underline{\boldsymbol{x}} - \underline{\boldsymbol{x}}'\|^2)$$

  measures "similarity" of point $\underline{\boldsymbol{x}}$ to point $\underline{\boldsymbol{x}}_i$:
  - $K(\underline{\boldsymbol{x}}, \underline{\boldsymbol{x}}')$ is 1 when points are the same
  - Value is 0 when points are distant
  - Effect strength depends on γ
- Using kernels to compute features:
  - Kernels often rely on distance between vectors
    - E.g., euclidean norm $\|\underline{\boldsymbol{x}} - \underline{\boldsymbol{x}}'\|^2$
  - Need to scale features for similar effects among coordinates

UNIVERSITY OF MARYLAND 1856 SCIENCE ACADEMY

Linear Kernel

- Consider the transformation Φ as the identity function $\Phi(\underline{\boldsymbol{x}}) = \underline{\boldsymbol{x}}$
- The kernel function is:

$$K_{\Phi}(\underline{\boldsymbol{x}}, \underline{\boldsymbol{x}}') = \underline{\boldsymbol{x}}^T \underline{\boldsymbol{x}}'$$

  - A **linear kernel** means using no kernel
  - It is just a "pass-through"

SCIENCE ACADEMY

Polynomial Kernel

- Given a point $\underline{\boldsymbol{x}} \in \mathbb{R}^n$, consider the function with two parameters k and d

$$K_\Phi(\underline{\boldsymbol{x}}, \underline{\boldsymbol{x}}') = (k + \underline{\boldsymbol{x}}^T \underline{\boldsymbol{x}}')^d$$

- It is called **polynomial** since if you expand the dot product you get a polynomial
- It can be proved that this is always a kernel

Kernel: Identifying a Function as a Kernel

- **Problem**:
  - You have a certain function $K(\underline{\boldsymbol{x}}, \underline{\boldsymbol{x}}')$ and you want to show that $K(\cdot)$ is an inner product in the form for some function $\Phi(\cdot)$

$$K(\underline{\boldsymbol{x}}, \underline{\boldsymbol{x}}') = \Phi(\underline{\boldsymbol{x}})^T \Phi(\underline{\boldsymbol{x}}') \quad \forall \underline{\boldsymbol{x}}, \underline{\boldsymbol{x}}'$$

  for a certain Φ and $\mathcal{Z}$
- In theory, a given function $K(\underline{\boldsymbol{x}}, \underline{\boldsymbol{x}}')$ is a valid kernel iff:
  - It is a symmetric, and
  - Satisfies the Mercer's condition: the matrix $K(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{x}}_j)$ is definite semi-positive

Kernel: Example of Identifying a Kernel

- Let's show that:

$$K(\underline{\boldsymbol{x}}, \underline{\boldsymbol{x}}') = (k + \underline{\boldsymbol{x}}^T \underline{\boldsymbol{x}}')^d$$

  is a **kernel** for any n, k, d

- According to the definition you need to show that there is always a transform Φ:

$$\Phi : \mathcal{X} = \mathbb{R}^n \rightarrow \mathcal{Z} = \mathbb{R}^q$$

  with $q \gg d$, such that $K_\Phi = (k + \underline{\boldsymbol{x}}^T \underline{\boldsymbol{x}}')^d$

- **Example**
  - $\mathcal{X} = \mathbb{R}^2$, $K(\underline{\boldsymbol{x}}, \underline{\boldsymbol{x}}') = (1 + \underline{\boldsymbol{x}}^T \underline{\boldsymbol{x}}')^2 = (1 + x_1 x_1' + x_2 x_2')^2$
  - Compute the full expression in terms of the coordinates:

$$K(\underline{\boldsymbol{x}}, \underline{\boldsymbol{x}}') = (1 + x_1^2 {x'}_1^2 + x_2^2 {x'}_2^2 + 2x_1 x_1' + 2x_2 x_2' + 2x_1 x_1' x_2 x_2')$$

  - Choose:
    - $\mathcal{Z} = \mathbb{R}^6$
    - $\Phi(x_1, x_2) = (1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2)$
  - This is a particular case of the polynomial kernel

SCIENCE ACADEMY

A Kernel Is a Computational Shortcut

- In literature, the **kernel trick** is a **computational shortcut** for the dot product of transformed vectors
- Compare 2 ways to compute the inner product of transformed vectors for a polynomial kernel
  1. **Using definition**: compute images of vectors, then inner product in transformed space:

     $$(1, x_1, x_2, \sqrt{2}x_1x_2, x_1^2, x_2^2, \ldots)^T \cdot (1, x_1', \ldots)$$

     - Requires combinatorial powers and a large dot product
  2. **Kernel trick**: use kernel function for dot product in transformed space

     $$(k + \underline{\boldsymbol{x}}^T \underline{\boldsymbol{x}}')^d$$

     - Requires inner product of small vectors, then power of a number
- Kernel trick is more computationally efficient for inner product computation

SCIENCE ACADEMY

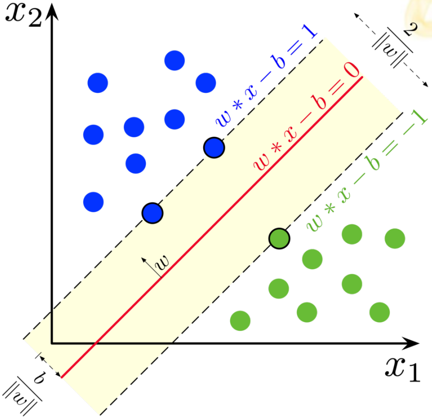
- ***Support Vector Machines (Optional)***

Support Vector Machines (SVM)

- Arguably one of the most successful classification algorithm, together with neural networks and random forests
- **Idea**: find a separating hyperplane that maximizes the distance from the class points (aka “margin”)
- **All the rage in 2005-2015**
  - Robust classifier handling outliers automatically
  - Strong theoretical justification of out-of-bound error
  - Strong link with VC dimension
  - Cool geometric interpretation
  - Solve a very complex optimization problem with some neat tricks
  - Works for both regression and classification
- SVM for classification:
  - Does not output probabilities (like logistic regression), but predicts directly the class
  - Has a notion of confidence, as distance from the margin

SVM Is a Large Margin Classifier

- Why **large margin** classifier is good?
- Given a linearly separable data set, the optimal separating line maximizes the margin:
    - More robust to noise
    - Large margin reduces VC dimension of hypothesis set



SCIENCE ACADEMY

SVM: Notation and Conventions

- Assume that:
  1. Outputs are encoded as $y_i \in \{-1, 1\}$
  2. Pull out w_0 from $\underline{\boldsymbol{w}}$
     - The bias $w_0 = b$ plays a different role
     - $\underline{\boldsymbol{w}} = (w_1, ..., w_d)$ and there is no $x_0 = 1$
     - $\underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}} + b = 0$ is the equation of the separating hyperplane
  3. $\underline{\boldsymbol{x}}_n$ is the closest point to the hyperplane
     - It can be multiple points from different classes
- Normalize $\underline{\boldsymbol{w}}$ and b to get a canonical representation of the hyperplane imposing $|\underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}}_n + b| = 1$

SVM: Original Form of Problem

- The SVM problem is:

$$
\begin{aligned}
&\text{find } \underline{\boldsymbol{w}}, b \\
&\text{maximize } \frac{1}{\|\underline{\boldsymbol{w}}\|} && \text{(max margin)} \\
&\text{subject to } \min_{i=1,\ldots,n} |\underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}}_i + b| = 1 && \text{(hyperplane)}
\end{aligned}
$$

- This problem is not friendly to optimization since it has norm, min, and absolute value

SCIENCE ACADEMY

Primal Form of SVM Problem

- You can rewrite it as:

$$\begin{array}{ll} \text{find } \underline{\boldsymbol{w}}, b & \\ \text{minimize} & \frac{1}{2}\underline{\boldsymbol{w}}^T\underline{\boldsymbol{w}} \\ \text{subject to} & y_i(\underline{\boldsymbol{w}}^T\underline{\boldsymbol{x}}_i + b) \geq 1 \ \forall i = 1, ..., n \end{array}$$

- Note that under $\underline{\boldsymbol{w}}$ minimal and linear separable classes, it is guaranteed that for at least one $\underline{\boldsymbol{x}}_i$ in the second equation will be equal to 1 (as in the original problem)
  - In fact otherwise we could scale down $\underline{\boldsymbol{w}}$ and b (which does not change the plane) to use the slack, against the hypothesis of minimality of $\underline{\boldsymbol{w}}$

Dual (Lagrangian) Form of SVM Problem

minimize with respect to $\underline{\boldsymbol{\alpha}}$ $\quad \mathcal{L}(\underline{\boldsymbol{\alpha}}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \underline{\boldsymbol{x}}_i^T \underline{\boldsymbol{x}}_j$

subject to $\quad \underline{\boldsymbol{\alpha}} \geq \underline{\mathbf{0}}, \sum_{i=1}^{N} \alpha_i y_i = 0$

$$\underline{\boldsymbol{w}} = \sum_{i=1}^{N} \alpha_i y_i \underline{\boldsymbol{x}}_i$$

- The equation for $\underline{\boldsymbol{w}}$ is not a constraint, but it computes $\underline{\boldsymbol{w}}$ (the plane) given $\underline{\boldsymbol{\alpha}}$, while b is given by $\min |\underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}}_i + b| = 1$

University of Maryland 1856 SCIENCE ACADEMY

Dual Form of SVM as QP Problem

- The dual form of SVM problem is a convex quadratic programming problem, in the form:

$$\begin{aligned} \text{minimize with respect to } \underline{\boldsymbol{\alpha}} \qquad & \underline{\mathbf{1}}^T \underline{\boldsymbol{\alpha}} - \frac{1}{2} \underline{\boldsymbol{\alpha}}^T \underline{\underline{\boldsymbol{Q}}} \underline{\boldsymbol{\alpha}} \\ \text{subject to} \qquad & \underline{\boldsymbol{\alpha}} \geq 0, \underline{\boldsymbol{y}}^T \underline{\boldsymbol{\alpha}} = 0 \end{aligned}$$

where:

- the matrix is $\underline{\underline{\boldsymbol{Q}}} = \{y_i y_j \underline{\boldsymbol{x}}_i^T \underline{\boldsymbol{x}}_j\}_{ij}$
- $\underline{\boldsymbol{\alpha}}$ is the column vector $(\alpha_1, \ldots, \alpha_N)$

SCIENCE ACADEMY

Solving Dual Formulation of SVM Problem (1/2)

- **Solving convex problem** for α
  - Feeding this problem to a QP solver, you get the optimal vector $\underline{\boldsymbol{\alpha}}$
- **Compute hyperplane $\underline{\boldsymbol{w}}$**
  - From $\underline{\boldsymbol{\alpha}}$ recover the plane $\underline{\boldsymbol{w}}$ from the equation: $\underline{\boldsymbol{w}} = \sum_{i=1}^{N} \alpha_i y_i \underline{\boldsymbol{x}}_i$
  - Looking at the optimal α_i, you can observe that many of them are 0
  - This is because when you applied the Lagrange multipliers to the inequalities: $y_i(\underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}}_i + b) \geq 1$, you got the KKT condition:

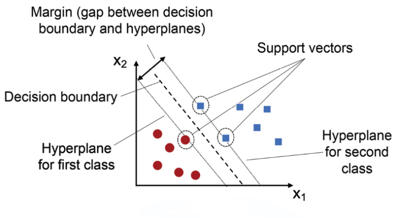
$$\alpha_i(y_i(\underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}}_i + b) - 1) = 0$$

  - From these equations, either
    - $\alpha_i = 0$ and $\underline{\boldsymbol{x}}_i$ is an *interior point* since it has non-null distance from the plane (i.e., slack) from the plane; or
    - $\alpha_i \neq 0$ and the slack is 0, which implies that the $\underline{\boldsymbol{x}}_i$ point touches the margin, i.e., it is a *support vector*

SCIENCE ACADEMY

Solving Dual Formulation of SVM Problem (2/2)

- Thus the hyperplane is only function of the support vectors:



$$\underline{\boldsymbol{w}} = \sum_{i=1}^{N} \alpha_i y_i \underline{\boldsymbol{x}}_i = \sum_{\underline{\boldsymbol{x}}_i \in SV} \alpha_i y_i \underline{\boldsymbol{x}}_i$$

since only for the support vectors $\alpha \neq 0$

  - The $\alpha_i \neq 0$ are the real degree of freedom

- **Compute** b
  - Once $\underline{\boldsymbol{w}}$ is known, you can use any support vector to compute b:

$$y_i(\underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}}_i + b) = 1$$

SCIENCE ACADEMY

Support Vectors and Degrees of Freedom for SVM

- The number of support vectors is related to the degrees of freedom of the model
- Because of the VC dimension, you have an in-sample quantity to bound the out-of-sample error:

$$E_{out} \leq E_{in} + c\frac{\text{num of SVs}}{N-1}$$

- You are "guaranteed" to not overfit

Non-Linear Transform for SVM

- $\Phi : \mathcal{X} \to \mathcal{Z}$ transforms $\underline{\boldsymbol{x}}_i$ into $\underline{\boldsymbol{z}}_i = \Phi(\underline{\boldsymbol{x}}_i) \in \mathbb{R}^{\tilde{d}}$ with $\tilde{d} > d$
- Transform vectors through Φ and apply SVM machinery
- Dual SVM formulation in $\mathcal{Z}$ space:

$$\mathcal{L}(\underline{\boldsymbol{\alpha}}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \underline{\boldsymbol{z}}_i^T \underline{\boldsymbol{z}}_j$$

- Note:
  - Optimization problem has same number of unknowns as original space (number of points N)
  - Support vectors live in $\mathcal{Z}$: they have $\alpha = 0$. In $\mathcal{X}$, they are pre-images of support vectors
  - Decision boundary and margin can be represented in original space (not linear)

SCIENCE ACADEMY

Non-Linear Transforms for SVM vs Others

- In SVM the non-linear transform does not change the number of unknowns and degrees of freedom of the model
- This is different from transforming the variables in a linear problem, since in that case the number of unknowns changes

SCIENCE ACADEMY

SVM in Higher Dimensional Space

- **Pros**
  - You don't pay the price in terms of complexity of optimization problem
    - Number of unknowns is still N (different than a linear problem)
  - You don't pay the price in terms of increased generalization bounds
    - Number of support vectors is $\leq N$
    - This is because each hypothesis h can be complex but the cardinality of the hypothesis set $\mathcal{H}$ is the same
- **Cons**
  - You pay a price to compute $\Phi(\underline{\boldsymbol{x}}_i)^T\Phi(\underline{\boldsymbol{x}}_j)$, since Φ could be very complex
    - The kernel trick will remove this extra complexity by doing $\Phi(\underline{\boldsymbol{x}}_i)^T\Phi(\underline{\boldsymbol{x}}_j) = K_\Phi(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{x}}_j)$

Non-Linear Transform in SVM vs Kernel Trick

- The trivial approach is
  - Transform vectors with $\Phi(\cdot)$
  - Apply all SVM machinery to the transformed vectors
- The issue is that Φ might be very complex, e.g., potentially exponential number of terms
- If you can express the SVM problem formulation and the prediction in terms of a kernel

$$K_\Phi(\underline{\boldsymbol{x}}, \underline{\boldsymbol{x}}') = \Phi(\underline{\boldsymbol{x}})^T \Phi(\underline{\boldsymbol{x}}') = \underline{\boldsymbol{z}}^T \underline{\boldsymbol{z}}'$$

you would need the kernel of the transformation $\Phi(\cdot)$ (and not $\Phi(\cdot)$) itself

SVM Formulation in Terms of Kernel: Optimization Step

- When you build the QP formulation for the Lagrangian to compute the α we can use $K_\Phi(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{x}}_j)$ instead of $\underline{\boldsymbol{z}}_i^T \underline{\boldsymbol{z}}_j$

$$\mathcal{L}(\underline{\boldsymbol{\alpha}}) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m K_\Phi(\underline{\boldsymbol{x}}_n, \underline{\boldsymbol{x}}_m)$$

- $\underline{\boldsymbol{z}}_n$ does not appear in the constraints

$$\underline{\boldsymbol{\alpha}} \geq \underline{\mathbf{0}}, \underline{\boldsymbol{\alpha}}^T \underline{\boldsymbol{y}} = 0$$

SCIENCE ACADEMY

SVM Formulation in Terms of Kernel: Prediction Step

- You need only inner products to compute a prediction for a given $\underline{\boldsymbol{z}}$
- In fact to make predictions, you replace the expression of $\underline{\tilde{\boldsymbol{w}}} = \sum_{i:\alpha_i>0} \alpha_i y_i \underline{\boldsymbol{z}}_i$ in $h(\underline{\boldsymbol{x}}) = \text{sign}(\underline{\boldsymbol{w}}^T \Phi(\underline{\boldsymbol{x}}) + b)$, yielding:

$$h(\underline{\boldsymbol{x}}) = \text{sign}(\sum_{i:\alpha_i>0} \alpha_i y_i K_\Phi(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{x}}) + b)$$

where b is given by $y_i(\underline{\boldsymbol{w}}^T \underline{\boldsymbol{z}}_i + b) = 1$ for any support vector $\underline{\boldsymbol{x}}_m$ and thus

$$b = \frac{1}{y_m} - \sum_{i:\alpha_i>0} \alpha_i y_i K_\Phi(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{x}}_m)$$

UNIVERSITY OF MARYLAND 1856 SCIENCE ACADEMY

Implications of Kernel Trick in SVM

- The "kernel trick" is a computational shortcut:
  - Use the kernel of the transformation instead of the transformation itself
- We have seen that in order to use SVMs we need only to be able to compute inner products between transformed vectors $\underline{\boldsymbol{z}}$
- The kernel trick implies:
  - No need to compute $\Phi()$: we just need the kernel of the transformation K_Φ and not the transformation Φ itself
  - No need to know Φ: if we have a function K_Φ and we know that is an inner product in some space, we can still use all the SVM machinery, even if we don't know what is the $\mathcal{Z}$ space or what is the transformation Φ
  - Φ can be impossible to compute: K_Φ can even correspond to a transformation Φ to an infinite dimensional space (e.g., Gaussian kernel)

Non-Linearly Separable SVM Problem

- In general there are 2 types of non-separable data sets:

1. Slightly non-separable
   - Few points crossing the boundary $\Longrightarrow$ use soft margin SVMs
2. Seriously non-separable
   - E.g., the class inside the circle $\Longrightarrow$ use non-linear transforms / kernels

- In practice, both issues are present and one can combine soft margin SVM and non-linear transforms

Soft-Margin SVM for Better Generalization on Linearly-Separable Data Sets

- Sometimes, even if the data is linearly separable, one can get better E_{out} using soft margin SVM at the cost of worst E_{in}
  - Usual trade off between in-sample and out-of-sample performance
  - E.g., in the data set there are a few of outliers that are forcing a smaller margin than what we could obtain if we ignore them, in order to get all the points classified correctly
- If C parameter is very large the SVM optimization requires to make the error very small, and this might trade off a large margin with getting all the classification right

Primal Formulation for Soft Margin SVM

- We want to introduce an error measure based on the margin violation for each point, so instead of the constraint:

$$y_i(\underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}}_i + b) \geq 1 \text{ (hard margin)}$$

we use:

$$y_i(\underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}}_i + b) \geq 1 - \xi_i, \text{ where } \xi_i \geq 0 \text{ (soft margin)}$$

- The cumulative margin violation is $C \sum_{i=1}^{N} \xi_i$
- The soft margin SVM optimization (primal form) is:

$$
\begin{aligned}
&\text{find } \underline{\boldsymbol{w}}, b, \underline{\boldsymbol{\xi}} \\
&\text{minimize} \quad \frac{1}{2}\underline{\boldsymbol{w}}^T \underline{\boldsymbol{w}} + C \sum_{i=1}^{N} \xi_i \\
&\text{subject to} \quad y_i(\underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}}_i + b) \geq 1 - \xi_i \ \forall i \\
&\qquad\qquad\quad\ \xi_i \geq 0
\end{aligned}
$$

Classes of Support Vectors for Soft Margin SVM

- There are 3 classes of points:
- *margin support vectors*: they are exactly on the margin defining it
  - In primal form: $y_i(\underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}}_i + b) = 1 \iff \xi_i = 0$
  - In dual form: $0 < \alpha_i < C$
- *non-margin support vectors*: they are inside the margin and classified correctly or not
  - In primal form: $y_i(\underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}}_i + b) < 1 \iff \xi_i > 0$
  - In dual form: $\alpha_i = C$
- *non-support vectors*, i.e., interior points:
  - In primal form: $y_i(\underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}}_i + b) > 1$
  - In dual form: $\alpha_i = 0$