CEE 697M: Probabilistic Machine Learning

M4 Nonparametric Methods: L4c: Support Vector Machines

Jimi Oke

UMass Amherst

College of Engineering

Wed, Apr 26, 2023

Outline

1. Introduction
2. Maximal margin classifier
3. Support vector classifier
4. Optimization for SVC
5. Nonlinearity in SVC
6. Support Vector Machines
7. Summary

Key concepts

Key concepts

- The*maximal margin classifier* partitions perfectly separable observations using a *separating hyperplane*.

Key concepts

- The*maximal margin classifier* partitions perfectly separable observations using a *separating hyperplane*.
- *Support vector classifiers* extend the *maximal margin classifier* by allowing for observations to overlap the linear decision boundary.

Key concepts

- The*maximal margin classifier* partitions perfectly separable observations using a *separating hyperplane*.
- *Support vector classifiers* extend the *maximal margin classifier* by allowing for observations to overlap the linear decision boundary.
- *Support vector machines (SVM)* generalize the *support vector classifier* by using kernels to allow for nonlinear decision boundaries.

Key concepts

- The*maximal margin classifier* partitions perfectly separable observations using a *separating hyperplane*.
- *Support vector classifiers* extend the *maximal margin classifier* by allowing for observations to overlap the linear decision boundary.
- *Support vector machines (SVM)* generalize the *support vector classifier* by using kernels to allow for nonlinear decision boundaries.

Key concepts

- The*maximal margin classifier* partitions perfectly separable observations using a *separating hyperplane*.
- *Support vector classifiers* extend the *maximal margin classifier* by allowing for observations to overlap the linear decision boundary.
- *Support vector machines (SVM)* generalize the *support vector classifier* by using kernels to allow for nonlinear decision boundaries.

Extensions

- SVM can be modified for regression problems (SVM regression)

Key concepts

- The*maximal margin classifier* partitions perfectly separable observations using a *separating hyperplane*.
- *Support vector classifiers* extend the *maximal margin classifier* by allowing for observations to overlap the linear decision boundary.
- *Support vector machines (SVM)* generalize the *support vector classifier* by using kernels to allow for nonlinear decision boundaries.

Extensions

- SVM can be modified for regression problems (SVM regression)
- Can also be used for multiclass problems

Key concepts

- The*maximal margin classifier* partitions perfectly separable observations using a *separating hyperplane*.
- *Support vector classifiers* extend the *maximal margin classifier* by allowing for observations to overlap the linear decision boundary.
- *Support vector machines (SVM)* generalize the *support vector classifier* by using kernels to allow for nonlinear decision boundaries.

Extensions

- SVM can be modified for regression problems (SVM regression)
- Can also be used for multiclass problems
- Can be regularized for improved performance

Key concepts

- The*maximal margin classifier* partitions perfectly separable observations using a *separating hyperplane*.
- *Support vector classifiers* extend the *maximal margin classifier* by allowing for observations to overlap the linear decision boundary.
- *Support vector machines (SVM)* generalize the *support vector classifier* by using kernels to allow for nonlinear decision boundaries.

Extensions

- SVM can be modified for regression problems (SVM regression)
- Can also be used for multiclass problems
- Can be regularized for improved performance
- Outputs can be converted to probabilities (Platt scaling)

Hyperplane

Hyperplane

Definition

A hyperplane is an affine[a] subspace of dimension $D-1$ in a D-dimensional affine space.

Hyperplane

Definition

A hyperplane is an affine[a] subspace of dimension $D-1$ in a D-dimensional affine space. It partitions a space in two.

[a] Generalization of Euclidean

Hyperplane

Definition

A hyperplane is an affine[a] subspace of dimension $D-1$ in a D-dimensional affine space. It partitions a space in two.

[a] Generalization of Euclidean

Hyperplane

Definition

A hyperplane is an affine[a] subspace of dimension $D-1$ in a D-dimensional affine space. It partitions a space in two.

[a] Generalization of Euclidean

Hyperplane

Definition

A hyperplane is an affine[a] subspace of dimension $D-1$ in a D-dimensional affine space. It partitions a space in two.

[a] Generalization of Euclidean

Hyperplane

Definition

A hyperplane is an affine[a] subspace of dimension $D-1$ in a D-dimensional affine space. It partitions a space in two.

[a] Generalization of Euclidean

Figure: Separating hyperplane in 2D

Hyperplane

Definition

A hyperplane is an affine[a] subspace of dimension $D-1$ in a D-dimensional affine space. It partitions a space in two.

[a] Generalization of Euclidean

Figure: Separating hyperplane in 2D

Hyperplane

Definition

A hyperplane is an affine[a] subspace of dimension $D-1$ in a D-dimensional affine space. It partitions a space in two.

[a] Generalization of Euclidean

Figure: Separating hyperplane in 2D

Hyperplane

Definition

A hyperplane is an affine[a] subspace of dimension $D-1$ in a D-dimensional affine space. It partitions a space in two.

[a] Generalization of Euclidean

Figure: Separating hyperplane in 2D

Hyperplane

Definition

A hyperplane is an affine[a] subspace of dimension $D-1$ in a D-dimensional affine space. It partitions a space in two.

[a] Generalization of Euclidean

Figure: Separating hyperplane in 2D

Figure: Separating hyperplane in 3D

Hyperplane — generalization

Hyperplane — generalization

In D dimensions, a hyperplane is then given by:

Hyperplane — generalization

In D dimensions, a hyperplane is then given by:

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D = 0 \tag{1}$$

(4)

Hyperplane — generalization

In D dimensions, a hyperplane is then given by:

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D = 0 \tag{1}$$

(4)

Hyperplane — generalization

In D dimensions, a hyperplane is then given by:

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D = 0 \tag{1}$$

(4)

Hyperplane — generalization

In D dimensions, a hyperplane is then given by:

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D = 0 \tag{1}$$

- $D + 1$ parameters are required to define hyperplane

(4)

Hyperplane — generalization

In D dimensions, a hyperplane is then given by:

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D = 0 \tag{1}$$

- $D + 1$ parameters are required to define hyperplane
- Relative to a hyperplane, points can be located in three distinct regions:

$$\tag{4}$$

Hyperplane — generalization

In D dimensions, a hyperplane is then given by:

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D = 0 \tag{1}$$

- $D + 1$ parameters are required to define hyperplane
- Relative to a hyperplane, points can be located in three distinct regions:

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D \quad = \quad 0 \quad \text{(on hyperplane itself)} \tag{2}$$

$$\tag{4}$$

Hyperplane — generalization

In D dimensions, a hyperplane is then given by:

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D = 0 \tag{1}$$

- $D + 1$ parameters are required to define hyperplane
- Relative to a hyperplane, points can be located in three distinct regions:

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D \quad = \quad 0 \quad \text{(on hyperplane itself)} \tag{2}$$

$$\tag{4}$$

Hyperplane — generalization

In D dimensions, a hyperplane is then given by:

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D = 0 \tag{1}$$

- $D + 1$ parameters are required to define hyperplane
- Relative to a hyperplane, points can be located in three distinct regions:

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D = 0 \quad \text{(on hyperplane itself)} \tag{2}$$

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D < 0 \quad \text{(\textit{one} side of hyperplane)} \tag{3}$$

$$\tag{4}$$

Hyperplane — generalization

In D dimensions, a hyperplane is then given by:

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D = 0 \tag{1}$$

- $D + 1$ parameters are required to define hyperplane
- Relative to a hyperplane, points can be located in three distinct regions:

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D = 0 \quad \text{(on hyperplane itself)} \tag{2}$$

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D < 0 \quad \text{(\textit{one} side of hyperplane)} \tag{3}$$

$$\tag{4}$$

Hyperplane — generalization

In D dimensions, a hyperplane is then given by:

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D = 0 \tag{1}$$

- $D + 1$ parameters are required to define hyperplane
- Relative to a hyperplane, points can be located in three distinct regions:

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D = 0 \quad \text{(on hyperplane itself)} \tag{2}$$

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D < 0 \quad \text{(\textit{one} side of hyperplane)} \tag{3}$$

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D > 0 \quad \text{(\textit{other} side of hyperplane)} \tag{4}$$

Hyperplane — generalization

In D dimensions, a hyperplane is then given by:

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D = 0 \tag{1}$$

- $D + 1$ parameters are required to define hyperplane
- Relative to a hyperplane, points can be located in three distinct regions:

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D = 0 \quad \text{(on hyperplane itself)} \tag{2}$$

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D < 0 \quad \text{(\textit{one} side of hyperplane)} \tag{3}$$

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D > 0 \quad \text{(\textit{other} side of hyperplane)} \tag{4}$$

Hyperplane — generalization

In D dimensions, a hyperplane is then given by:

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D = 0 \tag{1}$$

- $D + 1$ parameters are required to define hyperplane
- Relative to a hyperplane, points can be located in three distinct regions:

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D = 0 \quad \text{(on hyperplane itself)} \tag{2}$$

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D < 0 \quad \text{(\textit{one} side of hyperplane)} \tag{3}$$

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D > 0 \quad \text{(\textit{other} side of hyperplane)} \tag{4}$$

Figure: Separating hyperplane in three dimensions

Separating hyperplanes

Separating hyperplanes

Supposing a $\widetilde{y} \in \{-1, +1\}$ class coding for a D-dimensional dataset, then a separating hyperplane satisfies the following conditions:

Separating hyperplanes

Supposing a $\widetilde{y} \in \{-1, +1\}$ class coding for a D-dimensional dataset, then a separating hyperplane satisfies the following conditions:

$$\begin{cases} w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D > 0, \\ \end{cases}$$

Separating hyperplanes

Supposing a $\widetilde{y} \in \{-1, +1\}$ class coding for a D-dimensional dataset, then a separating hyperplane satisfies the following conditions:

$$\begin{cases} w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D > 0, & \widetilde{y}_n = +1 \\ \end{cases}$$

Separating hyperplanes

Supposing a $\widetilde{y} \in \{-1, +1\}$ class coding for a D-dimensional dataset, then a separating hyperplane satisfies the following conditions:

$$\begin{cases} w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D > 0, & \widetilde{y}_n = +1 \\ w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D < 0, & \end{cases}$$

Separating hyperplanes

Supposing a $\widetilde{y} \in \{-1, +1\}$ class coding for a D-dimensional dataset, then a separating hyperplane satisfies the following conditions:

$$\begin{cases} w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D > 0, & \widetilde{y}_n = +1 \\ w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D < 0, & \widetilde{y}_n = -1 \end{cases} \tag{5}$$

Separating hyperplanes

Supposing a $\widetilde{y} \in \{-1, +1\}$ class coding for a D-dimensional dataset, then a separating hyperplane satisfies the following conditions:

$$\begin{cases} w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D > 0, & \widetilde{y}_n = +1 \\ w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D < 0, & \widetilde{y}_n = -1 \end{cases} \tag{5}$$

Let

Separating hyperplanes

Supposing a $\widetilde{y} \in \{-1, +1\}$ class coding for a D-dimensional dataset, then a separating hyperplane satisfies the following conditions:

$$\begin{cases} w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D > 0, & \widetilde{y}_n = +1 \\ w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D < 0, & \widetilde{y}_n = -1 \end{cases} \tag{5}$$

Let

$$f(\boldsymbol{x}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D \tag{6}$$

Separating hyperplanes

Supposing a $\widetilde{y} \in \{-1, +1\}$ class coding for a D-dimensional dataset, then a separating hyperplane satisfies the following conditions:

$$\begin{cases} w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D > 0, & \widetilde{y}_n = +1 \\ w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D < 0, & \widetilde{y}_n = -1 \end{cases} \tag{5}$$

Let

$$f(\boldsymbol{x}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D \tag{6}$$

Multiplying $f(\boldsymbol{x})$ by $\widetilde{y}_n$,

Separating hyperplanes

Supposing a $\widetilde{y} \in \{-1, +1\}$ class coding for a D-dimensional dataset, then a separating hyperplane satisfies the following conditions:

$$\begin{cases} w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D > 0, & \widetilde{y}_n = +1 \\ w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D < 0, & \widetilde{y}_n = -1 \end{cases} \tag{5}$$

Let

$$f(\boldsymbol{x}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D \tag{6}$$

Multiplying $f(\boldsymbol{x})$ by $\widetilde{y}_n$, we see that Eq. (5) is equivalent to:

Separating hyperplanes

Supposing a $\widetilde{y} \in \{-1, +1\}$ class coding for a D-dimensional dataset, then a separating hyperplane satisfies the following conditions:

$$\begin{cases} w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D > 0, & \widetilde{y}_n = +1 \\ w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D < 0, & \widetilde{y}_n = -1 \end{cases} \tag{5}$$

Let

$$f(\boldsymbol{x}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D \tag{6}$$

Multiplying $f(\boldsymbol{x})$ by $\widetilde{y}_n$, we see that Eq. (5) is equivalent to:

$$\boxed{\widetilde{y}_n \cdot f(\boldsymbol{x}_n) > 0} \tag{7}$$

Separating hyperplanes

Supposing a $\widetilde{y} \in \{-1, +1\}$ class coding for a D-dimensional dataset, then a separating hyperplane satisfies the following conditions:

$$\begin{cases} w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D > 0, & \widetilde{y}_n = +1 \\ w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D < 0, & \widetilde{y}_n = -1 \end{cases} \tag{5}$$

Let

$$f(\boldsymbol{x}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D \tag{6}$$

Multiplying $f(\boldsymbol{x})$ by $\widetilde{y}_n$, we see that Eq. (5) is equivalent to:

$$\boxed{\widetilde{y}_n \cdot f(\boldsymbol{x}_n) > 0} \tag{7}$$

Figure: Selected separating hyperplane used as decision rule for classification

Separating hyperplane (summary)

Separating hyperplane (summary)

- Using a $\widetilde{y} \in \{-1, +1\}$ class coding, we assign a point x^* by:

Separating hyperplane (summary)

- Using a $\widetilde{y} \in \{-1, +1\}$ class coding, we assign a point x^* by:

Separating hyperplane (summary)

- Using a $\widetilde{y} \in \{-1, +1\}$ class coding, we assign a point x^* by:

$$\widetilde{y}_{x^*} = \text{sign}\{f(\boldsymbol{x}^*)\} \tag{8}$$

Separating hyperplane (summary)

- Using a $\widetilde{y} \in \{-1, +1\}$ class coding, we assign a point x^* by:

$$\widetilde{y}_{x^*} = \text{sign}\{f(\boldsymbol{x}^*)\} \tag{8}$$

where

$$f(\boldsymbol{x}^*) = w_0 + w_1 x_1^* + w_2 x_2^* + \cdots + w_D x_D^* \tag{9}$$

  - If $f(\boldsymbol{x}^*)$ is negative, the nth observation is assigned to Class -1.

Separating hyperplane (summary)

- Using a $\widetilde{y} \in \{-1, +1\}$ class coding, we assign a point x^* by:

$$\widetilde{y}_{x^*} = \text{sign}\{f(\boldsymbol{x}^*)\} \tag{8}$$

where

$$f(\boldsymbol{x}^*) = w_0 + w_1 x_1^* + w_2 x_2^* + \cdots + w_D x_D^* \tag{9}$$

  - If $f(\boldsymbol{x}^*)$ is negative, the nth observation is assigned to Class -1.
  - If $f(\boldsymbol{x}^*)$ is positive, the nth observation is assigned to Class $+1$.

Separating hyperplane (summary)

- Using a $\widetilde{y} \in \{-1, +1\}$ class coding, we assign a point x^* by:

$$\widetilde{y}_{x^*} = \text{sign}\{f(\boldsymbol{x}^*)\} \tag{8}$$

where

$$f(\boldsymbol{x}^*) = w_0 + w_1 x_1^* + w_2 x_2^* + \cdots + w_D x_D^* \tag{9}$$

  - If $f(\boldsymbol{x}^*)$ is negative, the nth observation is assigned to Class -1.
  - If $f(\boldsymbol{x}^*)$ is positive, the nth observation is assigned to Class $+1$.
- The magnitude of $f(\boldsymbol{x}^*)$ indicates the certainty of the assignment

Separating hyperplane (summary)

- Using a $\widetilde{y} \in \{-1, +1\}$ class coding, we assign a point x^* by:

$$\widetilde{y}_{x^*} = \text{sign}\{f(\boldsymbol{x}^*)\} \tag{8}$$

where

$$f(\boldsymbol{x}^*) = w_0 + w_1 x_1^* + w_2 x_2^* + \cdots + w_D x_D^* \tag{9}$$

  - If $f(\boldsymbol{x}^*)$ is negative, the nth observation is assigned to Class -1.
  - If $f(\boldsymbol{x}^*)$ is positive, the nth observation is assigned to Class $+1$.
- The magnitude of $f(\boldsymbol{x}^*)$ indicates the certainty of the assignment
  - If $f(\boldsymbol{x}^*)$ is large, then $\boldsymbol{x}^*$ is further away from the hyperplane and thus there is less uncertainty about its class prediction

Separating hyperplane (summary)

- Using a $\widetilde{y} \in \{-1, +1\}$ class coding, we assign a point x^* by:

$$\widetilde{y}_{x^*} = \text{sign}\{f(\boldsymbol{x}^*)\} \tag{8}$$

where

$$f(\boldsymbol{x}^*) = w_0 + w_1 x_1^* + w_2 x_2^* + \cdots + w_D x_D^* \tag{9}$$

  - If $f(\boldsymbol{x}^*)$ is negative, the nth observation is assigned to Class -1.
  - If $f(\boldsymbol{x}^*)$ is positive, the nth observation is assigned to Class $+1$.
- The magnitude of $f(\boldsymbol{x}^*)$ indicates the certainty of the assignment
  - If $f(\boldsymbol{x}^*)$ is large, then $\boldsymbol{x}^*$ is further away from the hyperplane and thus there is less uncertainty about its class prediction
- Using a separating hyperplane as a decision rule is equivalent to estimating a *linear* decision boundary.

Distance between a point and hyperplane

Distance between a point and hyperplane

The unit normal vector to the surface of a hyperplane L is given by:

Distance between a point and hyperplane

The unit normal vector to the surface of a hyperplane L is given by:

$$\boldsymbol{w}^* =$$

Distance between a point and hyperplane

The unit normal vector to the surface of a hyperplane L is given by:

$$\boldsymbol{w}^* = \frac{\boldsymbol{w}}{||\boldsymbol{w}||} \tag{10}$$

Distance between a point and hyperplane

The unit normal vector to the surface of a hyperplane L is given by:

$$\boldsymbol{w}^* = \frac{\boldsymbol{w}}{||\boldsymbol{w}||} \tag{10}$$

where we define the vector $\boldsymbol{w}$ as

$$\boldsymbol{w} = (w_1, w_2, \ldots, w_D) \tag{11}$$

Distance between a point and hyperplane

The unit normal vector to the surface of a hyperplane L is given by:

$$\boldsymbol{w}^* = \frac{\boldsymbol{w}}{||\boldsymbol{w}||} \tag{10}$$

where we define the vector $\boldsymbol{w}$ as

$$\boldsymbol{w} = (w_1, w_2, \ldots, w_D) \tag{11}$$

For any point $\boldsymbol{x}_n$ in L:

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

Distance between a point and hyperplane

The unit normal vector to the surface of a hyperplane L is given by:

$$\boldsymbol{w}^* = \frac{\boldsymbol{w}}{||\boldsymbol{w}||} \tag{10}$$

where we define the vector $\boldsymbol{w}$ as

$$\boldsymbol{w} = (w_1, w_2, \ldots, w_D) \tag{11}$$

For any point $\boldsymbol{x}_n$ in L:

$$w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n = 0 \implies \boldsymbol{w}^\top \boldsymbol{x}_n =$$

Distance between a point and hyperplane

The unit normal vector to the surface of a hyperplane L is given by:

$$\boldsymbol{w}^* = \frac{\boldsymbol{w}}{||\boldsymbol{w}||} \tag{10}$$

where we define the vector $\boldsymbol{w}$ as

$$\boldsymbol{w} = (w_1, w_2, \ldots, w_D) \tag{11}$$

For any point $\boldsymbol{x}_n$ in L:

$$w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n = 0 \implies \boldsymbol{w}^\top \boldsymbol{x}_n = -w_0 \tag{12}$$

Distance between a point and hyperplane

The unit normal vector to the surface of a hyperplane L is given by:

$$\boldsymbol{w}^* = \frac{\boldsymbol{w}}{||\boldsymbol{w}||} \tag{10}$$

where we define the vector $\boldsymbol{w}$ as

$$\boldsymbol{w} = (w_1, w_2, \ldots, w_D) \tag{11}$$

For any point $\boldsymbol{x}_n$ in L:

$$w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n = 0 \implies \boldsymbol{w}^\top \boldsymbol{x}_n = -w_0 \tag{12}$$

Finally, the **signed distance** of a point $\boldsymbol{x}_n$ to the hyperplane L is given by:

Distance between a point and hyperplane

The unit normal vector to the surface of a hyperplane L is given by:

$$\boldsymbol{w}^* = \frac{\boldsymbol{w}}{||\boldsymbol{w}||} \tag{10}$$

where we define the vector $\boldsymbol{w}$ as

$$\boldsymbol{w} = (w_1, w_2, \ldots, w_D) \tag{11}$$

For any point $\boldsymbol{x}_n$ in L:

$$w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n = 0 \implies \boldsymbol{w}^\top \boldsymbol{x}_n = -w_0 \tag{12}$$

Finally, the **signed distance** of a point $\boldsymbol{x}_n$ to the hyperplane L is given by:

$$\boldsymbol{w}^{*T}(\boldsymbol{x} - \boldsymbol{x}_n) =$$

Distance between a point and hyperplane

The unit normal vector to the surface of a hyperplane L is given by:

$$\boldsymbol{w}^* = \frac{\boldsymbol{w}}{||\boldsymbol{w}||} \tag{10}$$

where we define the vector $\boldsymbol{w}$ as

$$\boldsymbol{w} = (w_1, w_2, \ldots, w_D) \tag{11}$$

For any point $\boldsymbol{x}_n$ in L:

$$w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n = 0 \implies \boldsymbol{w}^\top \boldsymbol{x}_n = -w_0 \tag{12}$$

Finally, the **signed distance** of a point $\boldsymbol{x}_n$ to the hyperplane L is given by:

$$\boldsymbol{w}^{*T}(\boldsymbol{x} - \boldsymbol{x}_n) = \frac{1}{||\boldsymbol{w}||}\boldsymbol{w}^\top(\boldsymbol{x} - \boldsymbol{x}_n) =$$

Distance between a point and hyperplane

The unit normal vector to the surface of a hyperplane L is given by:

$$\boldsymbol{w}^* = \frac{\boldsymbol{w}}{||\boldsymbol{w}||} \tag{10}$$

where we define the vector $\boldsymbol{w}$ as

$$\boldsymbol{w} = (w_1, w_2, \ldots, w_D) \tag{11}$$

For any point $\boldsymbol{x}_n$ in L:

$$w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n = 0 \quad \Longrightarrow \quad \boldsymbol{w}^\top \boldsymbol{x}_n = -w_0 \tag{12}$$

Finally, the **signed distance** of a point $\boldsymbol{x}_n$ to the hyperplane L is given by:

$$\boldsymbol{w}^{*T}(\boldsymbol{x} - \boldsymbol{x}_n) = \frac{1}{||\boldsymbol{w}||}\boldsymbol{w}^\top(\boldsymbol{x} - \boldsymbol{x}_n) = \frac{1}{||\boldsymbol{w}||}\left(\boldsymbol{w}^\top \boldsymbol{x} - \boldsymbol{w}^\top \boldsymbol{x}_n\right)$$

Distance between a point and hyperplane

The unit normal vector to the surface of a hyperplane L is given by:

$$\boldsymbol{w}^* = \frac{\boldsymbol{w}}{||\boldsymbol{w}||} \tag{10}$$

where we define the vector $\boldsymbol{w}$ as

$$\boldsymbol{w} = (w_1, w_2, \ldots, w_D) \tag{11}$$

For any point $\boldsymbol{x}_n$ in L:

$$w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n = 0 \quad \Longrightarrow \quad \boldsymbol{w}^\top \boldsymbol{x}_n = -w_0 \tag{12}$$

Finally, the **signed distance** of a point $\boldsymbol{x}_n$ to the hyperplane L is given by:

$$\boldsymbol{w}^{*T}(\boldsymbol{x} - \boldsymbol{x}_n) = \frac{1}{||\boldsymbol{w}||}\boldsymbol{w}^\top(\boldsymbol{x} - \boldsymbol{x}_n) = \frac{1}{||\boldsymbol{w}||}\left(\boldsymbol{w}^\top \boldsymbol{x} - \boldsymbol{w}^\top \boldsymbol{x}_n\right)$$

$$=$$

Distance between a point and hyperplane

The unit normal vector to the surface of a hyperplane L is given by:

$$\boldsymbol{w}^* = \frac{\boldsymbol{w}}{||\boldsymbol{w}||} \tag{10}$$

where we define the vector $\boldsymbol{w}$ as

$$\boldsymbol{w} = (w_1, w_2, \ldots, w_D) \tag{11}$$

For any point $\boldsymbol{x}_n$ in L:

$$w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n = 0 \implies \boldsymbol{w}^\top \boldsymbol{x}_n = -w_0 \tag{12}$$

Finally, the **signed distance** of a point $\boldsymbol{x}_n$ to the hyperplane L is given by:

$$\begin{aligned} \boldsymbol{w}^{*T}(\boldsymbol{x} - \boldsymbol{x}_n) &= \frac{1}{||\boldsymbol{w}||}\boldsymbol{w}^\top(\boldsymbol{x} - \boldsymbol{x}_n) = \frac{1}{||\boldsymbol{w}||}\left(\boldsymbol{w}^\top \boldsymbol{x} - \boldsymbol{w}^\top \boldsymbol{x}_n\right) \\ &= \frac{1}{||\boldsymbol{w}||}\left(\boldsymbol{w}^\top \boldsymbol{x} - (-w_0)\right) = \end{aligned}$$

Distance between a point and hyperplane

The unit normal vector to the surface of a hyperplane L is given by:

$$\boldsymbol{w}^* = \frac{\boldsymbol{w}}{||\boldsymbol{w}||} \tag{10}$$

where we define the vector $\boldsymbol{w}$ as

$$\boldsymbol{w} = (w_1, w_2, \ldots, w_D) \tag{11}$$

For any point $\boldsymbol{x}_n$ in L:

$$w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n = 0 \quad \Longrightarrow \quad \boldsymbol{w}^\top \boldsymbol{x}_n = -w_0 \tag{12}$$

Finally, the **signed distance** of a point $\boldsymbol{x}_n$ to the hyperplane L is given by:

$$\begin{aligned} \boldsymbol{w}^{*T}(\boldsymbol{x} - \boldsymbol{x}_n) &= \frac{1}{||\boldsymbol{w}||}\boldsymbol{w}^\top(\boldsymbol{x} - \boldsymbol{x}_n) = \frac{1}{||\boldsymbol{w}||}\left(\boldsymbol{w}^\top \boldsymbol{x} - \boldsymbol{w}^\top \boldsymbol{x}_n\right) \\ &= \frac{1}{||\boldsymbol{w}||}\left(\boldsymbol{w}^\top \boldsymbol{x} - (-w_0)\right) = \frac{1}{||\boldsymbol{w}||}\left(\boldsymbol{w}^\top \boldsymbol{x} + w_0\right) = \end{aligned}$$

Distance between a point and hyperplane

The unit normal vector to the surface of a hyperplane L is given by:

$$\boldsymbol{w}^* = \frac{\boldsymbol{w}}{||\boldsymbol{w}||} \tag{10}$$

where we define the vector $\boldsymbol{w}$ as

$$\boldsymbol{w} = (w_1, w_2, \ldots, w_D) \tag{11}$$

For any point $\boldsymbol{x}_n$ in L:

$$w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n = 0 \implies \boldsymbol{w}^\top \boldsymbol{x}_n = -w_0 \tag{12}$$

Finally, the **signed distance** of a point $\boldsymbol{x}_n$ to the hyperplane L is given by:

$$\begin{aligned} \boldsymbol{w}^{*T}(\boldsymbol{x} - \boldsymbol{x}_n) &= \frac{1}{||\boldsymbol{w}||}\boldsymbol{w}^\top(\boldsymbol{x} - \boldsymbol{x}_n) = \frac{1}{||\boldsymbol{w}||}\left(\boldsymbol{w}^\top \boldsymbol{x} - \boldsymbol{w}^\top \boldsymbol{x}_n\right) \\ &= \frac{1}{||\boldsymbol{w}||}\left(\boldsymbol{w}^\top \boldsymbol{x} - (-w_0)\right) = \frac{1}{||\boldsymbol{w}||}\left(\boldsymbol{w}^\top \boldsymbol{x} + w_0\right) = \frac{1}{||f'(\boldsymbol{x})||} f(\boldsymbol{x}) \end{aligned} \tag{13}$$

Distance between a point and a hyperplane (illustrated)

Distance between a point and a hyperplane (illustrated)

Distance between a point and a hyperplane (illustrated)

The perpendicular distance between a point $\boldsymbol{x}_n$ and a hyperplane is given by:

Distance between a point and a hyperplane (illustrated)

The perpendicular distance between a point $\boldsymbol{x}_n$ and a hyperplane is given by:

$$||(x - x_0)|| \cos\theta =$$

Distance between a point and a hyperplane (illustrated)

The perpendicular distance between a point $\boldsymbol{x}_n$ and a hyperplane is given by:

$$||(x - x_0)|| \cos\theta = \frac{\boldsymbol{w}^\top(\boldsymbol{x} - \boldsymbol{x}_n)}{||\boldsymbol{w}||}$$

Distance between a point and a hyperplane (illustrated)

The perpendicular distance between a point $\boldsymbol{x}_n$ and a hyperplane is given by:

$$||(x - x_0)|| \cos\theta = \frac{\boldsymbol{w}^\top(\boldsymbol{x} - \boldsymbol{x}_n)}{||\boldsymbol{w}||} = \boldsymbol{w}^{*T}(\boldsymbol{x} - \boldsymbol{x}_n) \tag{14}$$

Margin of a separating hyperplane

Margin of a separating hyperplane

The **margin** of a separating hyperplane is the smallest [perpendicular] distance from the hyperplane to any of the observations.

Margin of a separating hyperplane

The **margin** of a separating hyperplane is the smallest [perpendicular] distance from the hyperplane to any of the observations.

$$M = \min_{n=1}^{N} \boldsymbol{w}^{*\top}(\boldsymbol{x} - \boldsymbol{x}_n)$$

Margin of a separating hyperplane

The **margin** of a separating hyperplane is the smallest [perpendicular] distance from the hyperplane to any of the observations.

$$M = \min_{n=1}^{N} \boldsymbol{w}^{*\top}(\boldsymbol{x} - \boldsymbol{x}_n) = \min_{n=1}^{N}[\widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n)] \tag{15}$$

Margin of a separating hyperplane

The **margin** of a separating hyperplane is the smallest [perpendicular] distance from the hyperplane to any of the observations.

$$M = \min_{n=1}^{N} \boldsymbol{w}^{*\top}(\boldsymbol{x} - \boldsymbol{x}_n) = \min_{n=1}^{N}[\widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n)] \tag{15}$$

- The datapoints defining the margin are called **support vectors**

Margin of a separating hyperplane

The **margin** of a separating hyperplane is the smallest [perpendicular] distance from the hyperplane to any of the observations.

$$M = \min_{n=1}^{N} \boldsymbol{w}^{*\top}(\boldsymbol{x} - \boldsymbol{x}_n) = \min_{n=1}^{N}[\widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n)] \tag{15}$$

- The datapoints defining the margin are called **support vectors**
- In the figure on the right the support vectors for hyperplane B are shown

Margin of a separating hyperplane

The **margin** of a separating hyperplane is the smallest [perpendicular] distance from the hyperplane to any of the observations.

$$M = \min_{n=1}^{N} \boldsymbol{w}^{*\top}(\boldsymbol{x} - \boldsymbol{x}_n) = \min_{n=1}^{N}[\widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n)] \tag{15}$$

- The datapoints defining the margin are called **support vectors**
- In the figure on the right the support vectors for hyperplane B are shown
- Can you identify the support vectors for A? How many are there?

Margin of a separating hyperplane

The **margin** of a separating hyperplane is the smallest [perpendicular] distance from the hyperplane to any of the observations.

$$M = \min_{n=1}^{N} \boldsymbol{w}^{*\top}(\boldsymbol{x} - \boldsymbol{x}_n) = \min_{n=1}^{N}[\widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n)] \tag{15}$$

- The datapoints defining the margin are called **support vectors**
- In the figure on the right the support vectors for hyperplane B are shown
- Can you identify the support vectors for A? How many are there?

Margin of a separating hyperplane

The **margin** of a separating hyperplane is the smallest [perpendicular] distance from the hyperplane to any of the observations.

$$M = \min_{n=1}^{N} \boldsymbol{w}^{*\top}(\boldsymbol{x} - \boldsymbol{x}_n) = \min_{n=1}^{N}[\widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n)] \tag{15}$$

- The datapoints defining the margin are called **support vectors**
- In the figure on the right the support vectors for hyperplane B are shown
- Can you identify the support vectors for A? How many are there?

Figure: Margins of two separating hyperplanes

Optimal separating hyperplane

Optimal separating hyperplane

- A perfectly separable dataset has an infinite number of separating hyperplanes

Optimal separating hyperplane

- A perfectly separable dataset has an infinite number of separating hyperplanes
- We can use **maximal margin** criterion to select the optimal separating hyperplane:

Optimal separating hyperplane

- A perfectly separable dataset has an infinite number of separating hyperplanes
- We can use **maximal margin** criterion to select the optimal separating hyperplane:
  - Define the margin as M

Optimal separating hyperplane

- A perfectly separable dataset has an infinite number of separating hyperplanes
- We can use **maximal margin** criterion to select the optimal separating hyperplane:
  - Define the margin as M
  - Find/choose the separating hyperplane for which M is the largest

Optimal separating hyperplane

- A perfectly separable dataset has an infinite number of separating hyperplanes
- We can use **maximal margin** criterion to select the optimal separating hyperplane:
  - Define the margin as M
  - Find/choose the separating hyperplane for which M is the largest
- Thus the optimal separating hyperplane is also called the **maximal margin hyperplane**

Optimal separating hyperplane

- A perfectly separable dataset has an infinite number of separating hyperplanes
- We can use **maximal margin** criterion to select the optimal separating hyperplane:
  - Define the margin as M
  - Find/choose the separating hyperplane for which M is the largest
- Thus the optimal separating hyperplane is also called the **maximal margin hyperplane**

Optimal separating hyperplane

- A perfectly separable dataset has an infinite number of separating hyperplanes
- We can use **maximal margin** criterion to select the optimal separating hyperplane:
  - Define the margin as M
  - Find/choose the separating hyperplane for which M is the largest
- Thus the optimal separating hyperplane is also called the **maximal margin hyperplane**

Figure: Margins of two separating hyperplanes.

Optimal separating hyperplane

- A perfectly separable dataset has an infinite number of separating hyperplanes
- We can use **maximal margin** criterion to select the optimal separating hyperplane:
  - Define the margin as M
  - Find/choose the separating hyperplane for which M is the largest
- Thus the optimal separating hyperplane is also called the **maximal margin hyperplane**

Figure: Margins of two separating hyperplanes. Which of the two is optimal? Why?

Optimal separating hyperplane

- A perfectly separable dataset has an infinite number of separating hyperplanes
- We can use **maximal margin** criterion to select the optimal separating hyperplane:
  - Define the margin as M
  - Find/choose the separating hyperplane for which M is the largest
- Thus the optimal separating hyperplane is also called the **maximal margin hyperplane**

Figure: Margins of two separating hyperplanes. Which of the two is optimal? Why?
Hyperplane A is the maximal margin hyperplane

Finding the maximal margin classifier

Finding the maximal margin classifier

Using the optimal separating hyperplane as a decision rule produces the **maximal margin classifier**.

Finding the maximal margin classifier

Using the optimal separating hyperplane as a decision rule produces the **maximal margin classifier**.

Formally, this is the solution to the optimization problem:

Finding the maximal margin classifier

Using the optimal separating hyperplane as a decision rule produces the **maximal margin classifier**.

Formally, this is the solution to the optimization problem:

$$\max_{\boldsymbol{w}, M} M \tag{16}$$

Finding the maximal margin classifier

Using the optimal separating hyperplane as a decision rule produces the **maximal margin classifier**.

Formally, this is the solution to the optimization problem:

$$\max_{\boldsymbol{w}, M} M \tag{16}$$

subject to

Finding the maximal margin classifier

Using the optimal separating hyperplane as a decision rule produces the **maximal margin classifier**.

Formally, this is the solution to the optimization problem:

$$\max_{\boldsymbol{w}, M} M \tag{16}$$

subject to

$$\sum_{d=1}^{D} w_d^2 = ||\boldsymbol{w}||$$

Finding the maximal margin classifier

Using the optimal separating hyperplane as a decision rule produces the **maximal margin classifier**.

Formally, this is the solution to the optimization problem:

$$\max_{\boldsymbol{w}, M} M \tag{16}$$

subject to

$$\sum_{d=1}^{D} w_d^2 = ||\boldsymbol{w}|| = 1 \tag{17}$$

Finding the maximal margin classifier

Using the optimal separating hyperplane as a decision rule produces the **maximal margin classifier**.

Formally, this is the solution to the optimization problem:

$$\max_{\boldsymbol{w}, M} M \tag{16}$$

subject to

$$\begin{aligned} \sum_{d=1}^{D} w_d^2 &= ||\boldsymbol{w}|| = 1 \\ \widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) &\geq M \end{aligned} \tag{17}$$

Finding the maximal margin classifier

Using the optimal separating hyperplane as a decision rule produces the **maximal margin classifier**.

Formally, this is the solution to the optimization problem:

$$\max_{\boldsymbol{w}, M} M \tag{16}$$

subject to

$$\sum_{d=1}^{D} w_d^2 = ||\boldsymbol{w}|| = 1 \tag{17}$$

$$\widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq M \quad \forall n = 1, \ldots, N \tag{18}$$

Finding the maximal margin classifier

Using the optimal separating hyperplane as a decision rule produces the **maximal margin classifier**.
Formally, this is the solution to the optimization problem:

$$\max_{\boldsymbol{w}, M} M \tag{16}$$

subject to

$$\sum_{d=1}^{D} w_d^2 = ||\boldsymbol{w}|| = 1 \tag{17}$$

$$\widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq M \quad \forall n = 1, \ldots, N \tag{18}$$

Recall that the distance between a point x and a hyperplane is $\frac{1}{||\boldsymbol{w}||}\left(\boldsymbol{w}^\top \boldsymbol{x}_n + w_0\right)$.

Finding the maximal margin classifier

Using the optimal separating hyperplane as a decision rule produces the **maximal margin classifier**.
Formally, this is the solution to the optimization problem:

$$\max_{\boldsymbol{w}, M} M \tag{16}$$

subject to

$$\sum_{d=1}^{D} w_d^2 = ||\boldsymbol{w}|| = 1 \tag{17}$$

$$\widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq M \quad \forall n = 1, \ldots, N \tag{18}$$

Recall that the distance between a point x and a hyperplane is $\frac{1}{||\boldsymbol{w}||}\left(\boldsymbol{w}^\top \boldsymbol{x}_n + w_0\right)$. Thus, Constraint (17) ensures that we can obtain the distance between a point $\boldsymbol{x}_n$ and the hyperplane by

Finding the maximal margin classifier

Using the optimal separating hyperplane as a decision rule produces the **maximal margin classifier**.
Formally, this is the solution to the optimization problem:

$$\max_{\boldsymbol{w}, M} M \tag{16}$$

subject to

$$\sum_{d=1}^{D} w_d^2 = ||\boldsymbol{w}|| = 1 \tag{17}$$

$$\widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq M \quad \forall n = 1, \ldots, N \tag{18}$$

Recall that the distance between a point x and a hyperplane is $\frac{1}{||\boldsymbol{w}||}\left(\boldsymbol{w}^\top \boldsymbol{x}_n + w_0\right)$. Thus, Constraint (17) ensures that we can obtain the distance between a point $\boldsymbol{x}_n$ and the hyperplane by

$$d_n = \widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \tag{19}$$

Finding the maximal margin classifier

Using the optimal separating hyperplane as a decision rule produces the **maximal margin classifier**.
Formally, this is the solution to the optimization problem:

$$\max_{\boldsymbol{w}, M} M \tag{16}$$

subject to

$$\sum_{d=1}^{D} w_d^2 = ||\boldsymbol{w}|| = 1 \tag{17}$$

$$\widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq M \quad \forall n = 1, \ldots, N \tag{18}$$

Recall that the distance between a point x and a hyperplane is $\frac{1}{||\boldsymbol{w}||}\left(\boldsymbol{w}^\top \boldsymbol{x}_n + w_0\right)$. Thus, Constraint (17) ensures that we can obtain the distance between a point $\boldsymbol{x}_n$ and the hyperplane by

$$d_n = \widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \tag{19}$$

And Constraint (18) assures that the distance between all training observations and the classifier will be no less than M.

Overlapping of classes

Overlapping of classes

- When observations do not overlap, then separating hyperplanes exist and a maximal margin classifier can be found

Figure: (Left) **Maximal margin classifier:** only possible if data are perfectly separable. (Right) **Support vector classifier:** relaxes the margin and allows observations to fall on the wrong side of margin (M) and hyperplane (H). Source: https://www.surveypractice.org/article/2715-using-support-vector-machines-for-survey-research

Overlapping of classes

- When observations do not overlap, then separating hyperplanes exist and a maximal margin classifier can be found
- If observations overlap, there is no solution to the maximal margin classifier

Figure: (Left) **Maximal margin classifier:** only possible if data are perfectly separable. (Right) **Support vector classifier:** relaxes the margin and allows observations to fall on the wrong side of margin (M) and hyperplane (H). Source: https://www.surveypractice.org/article/2715-using-support-vector-machines-for-survey-research

Overlapping of classes

- When observations do not overlap, then separating hyperplanes exist and a maximal margin classifier can be found
- If observations overlap, there is no solution to the maximal margin classifier
- We must relax the optimization problem to allow for minimal observations to fall on the wrong side of the margin

Figure: (Left) **Maximal margin classifier:** only possible if data are perfectly separable. (Right) **Support vector classifier:** relaxes the margin and allows observations to fall on the wrong side of margin (M) and hyperplane (H). Source: https://www.surveypractice.org/article/2715-using-support-vector-machines-for-survey-research

Overlapping of classes

- When observations do not overlap, then separating hyperplanes exist and a maximal margin classifier can be found
- If observations overlap, there is no solution to the maximal margin classifier
- We must relax the optimization problem to allow for minimal observations to fall on the wrong side of the margin
- We do this using *slack variables* (producing a **soft margin**)

Figure: (Left) **Maximal margin classifier:** only possible if data are perfectly separable. (Right) **Support vector classifier:** relaxes the margin and allows observations to fall on the wrong side of margin (M) and hyperplane (H). Source: https://www.surveypractice.org/article/2715-using-support-vector-machines-for-survey-research

Overlapping of classes

- When observations do not overlap, then separating hyperplanes exist and a maximal margin classifier can be found
- If observations overlap, there is no solution to the maximal margin classifier
- We must relax the optimization problem to allow for minimal observations to fall on the wrong side of the margin
- We do this using *slack variables* (producing a **soft margin**)
- The resulting optimization problem results in the **support vector classifier**

Figure: (Left) **Maximal margin classifier:** only possible if data are perfectly separable. (Right) **Support vector classifier:** relaxes the margin and allows observations to fall on the wrong side of margin (M) and hyperplane (H). Source: https://www.surveypractice.org/article/2715-using-support-vector-machines-for-survey-research

Overlapping of classes

- When observations do not overlap, then separating hyperplanes exist and a maximal margin classifier can be found
- If observations overlap, there is no solution to the maximal margin classifier
- We must relax the optimization problem to allow for minimal observations to fall on the wrong side of the margin
- We do this using *slack variables* (producing a **soft margin**)
- The resulting optimization problem results in the **support vector classifier**

Figure: (Left) **Maximal margin classifier:** only possible if data are perfectly separable. (Right) **Support vector classifier:** relaxes the margin and allows observations to fall on the wrong side of margin (M) and hyperplane (H). Source: https://www.surveypractice.org/article/2715-using-support-vector-machines-for-survey-research

Finding the support vector classifier

Finding the support vector classifier

The support vector classifier is the solution to:

Finding the support vector classifier

The support vector classifier is the solution to:

$$\max_{w_0, \boldsymbol{w}, \xi} M \tag{20}$$

Finding the support vector classifier

The support vector classifier is the solution to:

$$\max_{w_0, \boldsymbol{w}, \xi} M \tag{20}$$

subject to

Finding the support vector classifier

The support vector classifier is the solution to:

$$\max_{w_0, \boldsymbol{w}, \xi} M \tag{20}$$

subject to

$$\sum_{d=1}^{D} w_d^2 = ||\boldsymbol{w}|| = 1 \tag{21}$$

Finding the support vector classifier

The support vector classifier is the solution to:

$$\max_{w_0, \boldsymbol{w}, \xi} M \tag{20}$$

subject to

$$\sum_{d=1}^{D} w_d^2 = ||\boldsymbol{w}|| = 1 \tag{21}$$

$$\widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq M(1 - \xi_n) \quad \forall n = 1, \ldots, N \tag{22}$$

Finding the support vector classifier

The support vector classifier is the solution to:

$$\max_{w_0, \boldsymbol{w}, \xi} M \tag{20}$$

subject to

$$\sum_{d=1}^{D} w_d^2 = ||\boldsymbol{w}|| = 1 \tag{21}$$

$$\widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq M(1 - \xi_n) \quad \forall n = 1, \ldots, N \tag{22}$$

$$\xi_n \geq 0 \tag{23}$$

Finding the support vector classifier

The support vector classifier is the solution to:

$$\max_{w_0, \boldsymbol{w}, \xi} M \tag{20}$$

subject to

$$\sum_{d=1}^{D} w_d^2 = ||\boldsymbol{w}|| = 1 \tag{21}$$

$$\widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq M(1 - \xi_n) \quad \forall n = 1, \ldots, N \tag{22}$$

$$\xi_n \geq 0 \tag{23}$$

$$\sum_{n=1}^{N} \xi_n \leq B \tag{24}$$

Finding the support vector classifier

The support vector classifier is the solution to:

$$\max_{w_0, \boldsymbol{w}, \xi} M \tag{20}$$

subject to

$$\sum_{d=1}^{D} w_d^2 = ||\boldsymbol{w}|| = 1 \tag{21}$$

$$\widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq M(1 - \xi_n) \quad \forall n = 1, \ldots, N \tag{22}$$

$$\xi_n \geq 0 \tag{23}$$

$$\sum_{n=1}^{N} \xi_n \leq B \tag{24}$$

- ξ_n are **slack variables** allowing the nth observation to be on the wrong side (overlap)

Finding the support vector classifier

The support vector classifier is the solution to:

$$\max_{w_0, \boldsymbol{w}, \xi} M \tag{20}$$

subject to

$$\sum_{d=1}^{D} w_d^2 = ||\boldsymbol{w}|| = 1 \tag{21}$$

$$\widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq M(1 - \xi_n) \quad \forall n = 1, \ldots, N \tag{22}$$

$$\xi_n \geq 0 \tag{23}$$

$$\sum_{n=1}^{N} \xi_n \leq B \tag{24}$$

- ξ_n are **slack variables** allowing the nth observation to be on the wrong side (overlap)
- B is a nonnegative tuning parameter ("**budget**"), which moderates the size of the allowable overlap.

Finding the support vector classifier

The support vector classifier is the solution to:

$$\max_{w_0, \boldsymbol{w}, \xi} M \tag{20}$$

subject to

$$\sum_{d=1}^{D} w_d^2 = ||\boldsymbol{w}|| = 1 \tag{21}$$

$$\widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq M(1 - \xi_n) \quad \forall n = 1, \ldots, N \tag{22}$$

$$\xi_n \geq 0 \tag{23}$$

$$\sum_{n=1}^{N} \xi_n \leq B \tag{24}$$

- ξ_n are **slack variables** allowing the nth observation to be on the wrong side (overlap)
- B is a nonnegative tuning parameter ("**budget**"), which moderates the size of the allowable overlap.
- If $\xi_n = 0$ for all i, then the problem solves the **maximal margin classifier**

Slack variables in SVC

Slack variables in SVC

Slack variable ξ_n behavior for nth observation:

Slack variables in SVC

Slack variable ξ_n behavior for nth observation:

$$\xi_n$$

Slack variables in SVC

Slack variable ξ_n behavior for nth observation:

$$\xi_n \begin{cases} = 0, \\ \\ \end{cases}$$

Slack variables in SVC

Slack variable ξ_n behavior for nth observation:

$$\xi_n \begin{cases} = 0, & \text{correct side of margin} \\ & \\ & \end{cases}$$

Slack variables in SVC

Slack variable ξ_n behavior for nth observation:

$$\xi_n \begin{cases} = 0, & \text{correct side of margin} \\ > 0, & \\ & \end{cases}$$

Slack variables in SVC

Slack variable ξ_n behavior for nth observation:

$$\xi_n \begin{cases} = 0, & \text{correct side of margin} \\ > 0, & \text{wrong side of margin} \end{cases}$$

Slack variables in SVC

Slack variable ξ_n behavior for nth observation:

$$\xi_n \begin{cases} = 0, & \text{correct side of margin} \\ > 0, & \text{wrong side of margin} \\ > 1, & \end{cases}$$

Slack variables in SVC

Slack variable ξ_n behavior for nth observation:

$$\xi_n \begin{cases} = 0, & \text{correct side of margin} \\ > 0, & \text{wrong side of margin} \\ > 1, & \text{wrong side of hyperplane (overlapping)} \end{cases} \tag{25}$$

Slack variables in SVC

Slack variable ξ_n behavior for nth observation:

$$\xi_n \begin{cases} = 0, & \text{correct side of margin} \\ > 0, & \text{wrong side of margin} \\ > 1, & \text{wrong side of hyperplane (overlapping)} \end{cases} \tag{25}$$

- Observations directly on the margin ($\xi_n = 0$) or on the wrong side of the margin ($\xi_1 > 0$) are the **support vectors**

Slack variables in SVC

Slack variable ξ_n behavior for nth observation:

$$\xi_n \begin{cases} = 0, & \text{correct side of margin} \\ > 0, & \text{wrong side of margin} \\ > 1, & \text{wrong side of hyperplane (overlapping)} \end{cases} \tag{25}$$

- Observations directly on the margin ($\xi_n = 0$) or on the wrong side of the margin ($\xi_1 > 0$) are the **support vectors**

Slack variables in SVC

Slack variable ξ_n behavior for nth observation:

$$\xi_n \begin{cases} = 0, & \text{correct side of margin} \\ > 0, & \text{wrong side of margin} \\ > 1, & \text{wrong side of hyperplane (overlapping)} \end{cases} \tag{25}$$

- Observations directly on the margin ($\xi_n = 0$) or on the wrong side of the margin ($\xi_1 > 0$) are the **support vectors**

Slack variables in SVC

Slack variable ξ_n behavior for nth observation:

$$\xi_n \begin{cases} = 0, & \text{correct side of margin} \\ > 0, & \text{wrong side of margin} \\ > 1, & \text{wrong side of hyperplane (overlapping)} \end{cases} \tag{25}$$

- Observations directly on the margin ($\xi_n = 0$) or on the wrong side of the margin ($\xi_1 > 0$) are the **support vectors**

SVC for a small dataset

Slack variables in SVC

Slack variable ξ_n behavior for nth observation:

$$\xi_n \begin{cases} = 0, & \text{correct side of margin} \\ > 0, & \text{wrong side of margin} \\ > 1, & \text{wrong side of hyperplane (overlapping)} \end{cases} \tag{25}$$

- Observations directly on the margin ($\xi_n = 0$) or on the wrong side of the margin ($\xi_1 > 0$) are the **support vectors**

SVC for a small dataset

- Class Violet:

Slack variables in SVC

Slack variable ξ_n behavior for nth observation:

$$\xi_n \begin{cases} = 0, & \text{correct side of margin} \\ > 0, & \text{wrong side of margin} \\ > 1, & \text{wrong side of hyperplane (overlapping)} \end{cases} \tag{25}$$

- Observations directly on the margin ($\xi_n = 0$) or on the wrong side of the margin ($\xi_1 > 0$) are the **support vectors**

SVC for a small dataset

- Class Violet:
  - Point 11 is on wrong side of hyperplane ($\xi_{11} > 1$)

Slack variables in SVC

Slack variable ξ_n behavior for nth observation:

$$\xi_n \begin{cases} = 0, & \text{correct side of margin} \\ > 0, & \text{wrong side of margin} \\ > 1, & \text{wrong side of hyperplane (overlapping)} \end{cases} \tag{25}$$

- Observations directly on the margin ($\xi_n = 0$) or on the wrong side of the margin ($\xi_1 > 0$) are the **support vectors**

SVC for a small dataset

- Class Violet:
  - Point 11 is on wrong side of hyperplane ($\xi_{11} > 1$)
  - Pt. 1 is on wrong side of margin ($\xi_1 > 0$)

Slack variables in SVC

Slack variable ξ_n behavior for nth observation:

$$\xi_n \begin{cases} = 0, & \text{correct side of margin} \\ > 0, & \text{wrong side of margin} \\ > 1, & \text{wrong side of hyperplane (overlapping)} \end{cases} \tag{25}$$

- Observations directly on the margin ($\xi_n = 0$) or on the wrong side of the margin ($\xi_1 > 0$) are the **support vectors**

SVC for a small dataset

- Class Violet:
  - Point 11 is on wrong side of hyperplane ($\xi_{11} > 1$)
  - Pt. 1 is on wrong side of margin ($\xi_1 > 0$)
  - Pt. 2 is on margin ($\xi_2 = 0$)

Slack variables in SVC

Slack variable ξ_n behavior for nth observation:

$$\xi_n \begin{cases} = 0, & \text{correct side of margin} \\ > 0, & \text{wrong side of margin} \\ > 1, & \text{wrong side of hyperplane (overlapping)} \end{cases} \tag{25}$$

- Observations directly on the margin ($\xi_n = 0$) or on the wrong side of the margin ($\xi_1 > 0$) are the **support vectors**

SVC for a small dataset

- Class Violet:
  - Point 11 is on wrong side of hyperplane ($\xi_{11} > 1$)
  - Pt. 1 is on wrong side of margin ($\xi_1 > 0$)
  - Pt. 2 is on margin ($\xi_2 = 0$)
- Class Sky Blue:

Slack variables in SVC

Slack variable ξ_n behavior for nth observation:

$$\xi_n \begin{cases} = 0, & \text{correct side of margin} \\ > 0, & \text{wrong side of margin} \\ > 1, & \text{wrong side of hyperplane (overlapping)} \end{cases} \tag{25}$$

- Observations directly on the margin ($\xi_n = 0$) or on the wrong side of the margin ($\xi_1 > 0$) are the **support vectors**

SVC for a small dataset

- Class Violet:
  - Point 11 is on wrong side of hyperplane ($\xi_{11} > 1$)
  - Pt. 1 is on wrong side of margin ($\xi_1 > 0$)
  - Pt. 2 is on margin ($\xi_2 = 0$)
- Class Sky Blue:
  - Pt. 12 is on wrong side of hyperplane

Slack variables in SVC

Slack variable ξ_n behavior for nth observation:

$$\xi_n \begin{cases} = 0, & \text{correct side of margin} \\ > 0, & \text{wrong side of margin} \\ > 1, & \text{wrong side of hyperplane (overlapping)} \end{cases} \tag{25}$$

- Observations directly on the margin ($\xi_n = 0$) or on the wrong side of the margin ($\xi_1 > 0$) are the **support vectors**

SVC for a small dataset

- Class Violet:
  - Point 11 is on wrong side of hyperplane ($\xi_{11} > 1$)
  - Pt. 1 is on wrong side of margin ($\xi_1 > 0$)
  - Pt. 2 is on margin ($\xi_2 = 0$)
- Class Sky Blue:
  - Pt. 12 is on wrong side of hyperplane
  - Pt. 8 is on wrong side of margin

Slack variables in SVC

Slack variable ξ_n behavior for nth observation:

$$\xi_n \begin{cases} = 0, & \text{correct side of margin} \\ > 0, & \text{wrong side of margin} \\ > 1, & \text{wrong side of hyperplane (overlapping)} \end{cases} \tag{25}$$

- Observations directly on the margin ($\xi_n = 0$) or on the wrong side of the margin ($\xi_1 > 0$) are the **support vectors**

SVC for a small dataset

- Class Violet:
  - Point 11 is on wrong side of hyperplane ($\xi_{11} > 1$)
  - Pt. 1 is on wrong side of margin ($\xi_1 > 0$)
  - Pt. 2 is on margin ($\xi_2 = 0$)
- Class Sky Blue:
  - Pt. 12 is on wrong side of hyperplane
  - Pt. 8 is on wrong side of margin
  - Pt. 7 and 9 are on margin

Slack variables in SVC

Slack variable ξ_n behavior for nth observation:

$$\xi_n \begin{cases} = 0, & \text{correct side of margin} \\ > 0, & \text{wrong side of margin} \\ > 1, & \text{wrong side of hyperplane (overlapping)} \end{cases} \tag{25}$$

- Observations directly on the margin ($\xi_n = 0$) or on the wrong side of the margin ($\xi_1 > 0$) are the **support vectors**

SVC for a small dataset

- Class Violet:
  - Point 11 is on wrong side of hyperplane ($\xi_{11} > 1$)
  - Pt. 1 is on wrong side of margin ($\xi_1 > 0$)
  - Pt. 2 is on margin ($\xi_2 = 0$)
- Class Sky Blue:
  - Pt. 12 is on wrong side of hyperplane
  - Pt. 8 is on wrong side of margin
  - Pt. 7 and 9 are on margin
- The **support vectors/points** are:

Slack variables in SVC

Slack variable ξ_n behavior for nth observation:

$$\xi_n \begin{cases} = 0, & \text{correct side of margin} \\ > 0, & \text{wrong side of margin} \\ > 1, & \text{wrong side of hyperplane (overlapping)} \end{cases} \tag{25}$$

- Observations directly on the margin ($\xi_n = 0$) or on the wrong side of the margin ($\xi_1 > 0$) are the **support vectors**

SVC for a small dataset

- Class Violet:
  - Point 11 is on wrong side of hyperplane ($\xi_{11} > 1$)
  - Pt. 1 is on wrong side of margin ($\xi_1 > 0$)
  - Pt. 2 is on margin ($\xi_2 = 0$)
- Class Sky Blue:
  - Pt. 12 is on wrong side of hyperplane
  - Pt. 8 is on wrong side of margin
  - Pt. 7 and 9 are on margin
- The **support vectors/points** are: 1, 2, 11,

Slack variables in SVC

Slack variable ξ_n behavior for nth observation:

$$\xi_n \begin{cases} = 0, & \text{correct side of margin} \\ > 0, & \text{wrong side of margin} \\ > 1, & \text{wrong side of hyperplane (overlapping)} \end{cases} \tag{25}$$

- Observations directly on the margin ($\xi_n = 0$) or on the wrong side of the margin ($\xi_1 > 0$) are the **support vectors**

SVC for a small dataset

- Class Violet:
  - Point 11 is on wrong side of hyperplane ($\xi_{11} > 1$)
  - Pt. 1 is on wrong side of margin ($\xi_1 > 0$)
  - Pt. 2 is on margin ($\xi_2 = 0$)
- Class Sky Blue:
  - Pt. 12 is on wrong side of hyperplane
  - Pt. 8 is on wrong side of margin
  - Pt. 7 and 9 are on margin
- The **support vectors/points** are: 1, 2, 11, 7, 8, 9 and 12.

Budget parameter and the bias-variance tradeoff

Budget parameter and the bias-variance tradeoff

- B determines the width of the margin (tolerance of overlap)

Budget parameter and the bias-variance tradeoff

- B determines the width of the margin (tolerance of overlap)
- Typically, B is chosen via k-fold cross-validation

Budget parameter and the bias-variance tradeoff

- B determines the width of the margin (tolerance of overlap)
- Typically, B is chosen via k-fold cross-validation
- As B becomes larger, the percentage of training observations that are support vectors increases (i.e. more observations are on the wrong side of margin or hyperplane)

Budget parameter and the bias-variance tradeoff

- B determines the width of the margin (tolerance of overlap)
- Typically, B is chosen via k-fold cross-validation
- As B becomes larger, the percentage of training observations that are support vectors increases (i.e. more observations are on the wrong side of margin or hyperplane)
- Large B increases bias (more misclassifications) but reduces variance (greater stability)

Budget parameter and the bias-variance tradeoff

- B determines the width of the margin (tolerance of overlap)
- Typically, B is chosen via k-fold cross-validation
- As B becomes larger, the percentage of training observations that are support vectors increases (i.e. more observations are on the wrong side of margin or hyperplane)
- Large B increases bias (more misclassifications) but reduces variance (greater stability)

Budget parameter and the bias-variance tradeoff

- B determines the width of the margin (tolerance of overlap)
- Typically, B is chosen via k-fold cross-validation
- As B becomes larger, the percentage of training observations that are support vectors increases (i.e. more observations are on the wrong side of margin or hyperplane)
- Large B increases bias (more misclassifications) but reduces variance (greater stability)

Figure: Support vector classifier: B decreasing from Left to Right.

Budget parameter and the bias-variance tradeoff

- B determines the width of the margin (tolerance of overlap)
- Typically, B is chosen via k-fold cross-validation
- As B becomes larger, the percentage of training observations that are support vectors increases (i.e. more observations are on the wrong side of margin or hyperplane)
- Large B increases bias (more misclassifications) but reduces variance (greater stability)

Figure: Support vector classifier: B decreasing from Left to Right.

Budget parameter and the bias-variance tradeoff

- B determines the width of the margin (tolerance of overlap)
- Typically, B is chosen via k-fold cross-validation
- As B becomes larger, the percentage of training observations that are support vectors increases (i.e. more observations are on the wrong side of margin or hyperplane)
- Large B increases bias (more misclassifications) but reduces variance (greater stability)

Figure: Support vector classifier: B decreasing from Left to Right.

Budget parameter and the bias-variance tradeoff

- B determines the width of the margin (tolerance of overlap)
- Typically, B is chosen via k-fold cross-validation
- As B becomes larger, the percentage of training observations that are support vectors increases (i.e. more observations are on the wrong side of margin or hyperplane)
- Large B increases bias (more misclassifications) but reduces variance (greater stability)

Figure: Support vector classifier: B decreasing from Left to Right.

Reformulating [OP1]

Reformulating [OP1]

Recall that the *signed* distance between a point $\boldsymbol{x}_n$ and a separating hyperplane $f(\boldsymbol{x}) = w_0 + \boldsymbol{w}^\top x$ is given by:

$$\frac{1}{||\boldsymbol{w}||}(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \tag{26}$$

Since $\widetilde{y}_n = \{-1, 1\}$, then the absolute distance is given by:

Reformulating [OP1]

Recall that the *signed* distance between a point $\boldsymbol{x}_n$ and a separating hyperplane $f(\boldsymbol{x}) = w_0 + \boldsymbol{w}^\top x$ is given by:

$$\frac{1}{||\boldsymbol{w}||}(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \tag{26}$$

Since $\widetilde{y}_n = \{-1, 1\}$, then the absolute distance is given by:

$$\widetilde{y}_n \times \frac{1}{||\boldsymbol{w}||}(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \tag{27}$$

Reformulating [OP1]

Recall that the *signed* distance between a point $\boldsymbol{x}_n$ and a separating hyperplane $f(\boldsymbol{x}) = w_0 + \boldsymbol{w}^\top x$ is given by:

$$\frac{1}{||\boldsymbol{w}||}(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \tag{26}$$

Since $\widetilde{y}_n = \{-1, 1\}$, then the absolute distance is given by:

$$\widetilde{y}_n \times \frac{1}{||\boldsymbol{w}||}(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \tag{27}$$

Reformulating [OP1]

Recall that the *signed* distance between a point $\boldsymbol{x}_n$ and a separating hyperplane $f(\boldsymbol{x}) = w_0 + \boldsymbol{w}^\top x$ is given by:

$$\frac{1}{||\boldsymbol{w}||}(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \tag{26}$$

Since $\widetilde{y}_n = \{-1, 1\}$, then the absolute distance is given by:

$$\widetilde{y}_n \times \frac{1}{||\boldsymbol{w}||}(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \tag{27}$$

In OP1, constraining $||\boldsymbol{w}|| = 1$ allowed us to write the boundary constraint as:

$$\widetilde{y}_n \times \frac{1}{||\boldsymbol{w}||}(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) =$$

Reformulating [OP1]

Recall that the *signed* distance between a point $\boldsymbol{x}_n$ and a separating hyperplane $f(\boldsymbol{x}) = w_0 + \boldsymbol{w}^\top x$ is given by:

$$\frac{1}{||\boldsymbol{w}||}(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \tag{26}$$

Since $\widetilde{y}_n = \{-1, 1\}$, then the absolute distance is given by:

$$\widetilde{y}_n \times \frac{1}{||\boldsymbol{w}||}(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \tag{27}$$

In OP1, constraining $||\boldsymbol{w}|| = 1$ allowed us to write the boundary constraint as:

$$\widetilde{y}_n \times \frac{1}{||\boldsymbol{w}||}(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) = \widetilde{y}_n \times \frac{1}{1}(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n)$$

Reformulating [OP1]

Recall that the *signed* distance between a point $\boldsymbol{x}_n$ and a separating hyperplane $f(\boldsymbol{x}) = w_0 + \boldsymbol{w}^\top x$ is given by:

$$\frac{1}{||\boldsymbol{w}||}(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \tag{26}$$

Since $\widetilde{y}_n = \{-1, 1\}$, then the absolute distance is given by:

$$\widetilde{y}_n \times \frac{1}{||\boldsymbol{w}||}(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \tag{27}$$

In OP1, constraining $||\boldsymbol{w}|| = 1$ allowed us to write the boundary constraint as:

$$\widetilde{y}_n \times \frac{1}{||\boldsymbol{w}||}(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) = \widetilde{y}_n \times \frac{1}{1}(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq M(1 - \xi_n) \tag{28}$$

Reformulating [OP1]

Recall that the *signed* distance between a point $\boldsymbol{x}_n$ and a separating hyperplane $f(\boldsymbol{x}) = w_0 + \boldsymbol{w}^\top x$ is given by:

$$\frac{1}{||\boldsymbol{w}||}(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \tag{26}$$

Since $\widetilde{y}_n = \{-1, 1\}$, then the absolute distance is given by:

$$\widetilde{y}_n \times \frac{1}{||\boldsymbol{w}||}(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \tag{27}$$

In OP1, constraining $||\boldsymbol{w}|| = 1$ allowed us to write the boundary constraint as:

$$\widetilde{y}_n \times \frac{1}{||\boldsymbol{w}||}(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) = \widetilde{y}_n \times \frac{1}{1}(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq M(1 - \xi_n) \tag{28}$$

If instead we do not constrain $||\boldsymbol{w}||$, then the constraint can be written as:

Reformulating [OP1]

Recall that the *signed* distance between a point $\boldsymbol{x}_n$ and a separating hyperplane $f(\boldsymbol{x}) = w_0 + \boldsymbol{w}^\top x$ is given by:

$$\frac{1}{||\boldsymbol{w}||}(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \tag{26}$$

Since $\widetilde{y}_n = \{-1, 1\}$, then the absolute distance is given by:

$$\widetilde{y}_n \times \frac{1}{||\boldsymbol{w}||}(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \tag{27}$$

In OP1, constraining $||\boldsymbol{w}|| = 1$ allowed us to write the boundary constraint as:

$$\widetilde{y}_n \times \frac{1}{||\boldsymbol{w}||}(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) = \widetilde{y}_n \times \frac{1}{1}(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq M(1 - \xi_n) \tag{28}$$

If instead we do not constrain $||\boldsymbol{w}||$, then the constraint can be written as:

$$\widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq M||\boldsymbol{w}||(1 - \xi_n) \tag{29}$$

Reformulating [OP1] (cont.)

Reformulating [OP1] (cont.)

Now, let us set

$$M = \frac{1}{||\boldsymbol{w}||}$$

Reformulating [OP1] (cont.)

Now, let us set

$$M = \frac{1}{||\boldsymbol{w}||} \quad \Longrightarrow$$

Reformulating [OP1] (cont.)

Now, let us set

$$M = \frac{1}{||\boldsymbol{w}||} \implies ||\boldsymbol{w}|| = \frac{1}{M} \tag{30}$$

Reformulating [OP1] (cont.)

Now, let us set

$$M = \frac{1}{||\boldsymbol{w}||} \quad \Longrightarrow \quad ||\boldsymbol{w}|| = \frac{1}{M} \tag{30}$$

Two implications follow:

Reformulating [OP1] (cont.)

Now, let us set

$$M = \frac{1}{||\boldsymbol{w}||} \quad \Longrightarrow \quad ||\boldsymbol{w}|| = \frac{1}{M} \tag{30}$$

Two implications follow:

- *Maximizing* M (the margin) is equivalent to *minimizing* $||\boldsymbol{w}||$

Reformulating [OP1] (cont.)

Now, let us set

$$M = \frac{1}{||\boldsymbol{w}||} \quad \Longrightarrow \quad ||\boldsymbol{w}|| = \frac{1}{M} \tag{30}$$

Two implications follow:

- *Maximizing* M (the margin) is equivalent to *minimizing* $||\boldsymbol{w}||$

Reformulating [OP1] (cont.)

Now, let us set

$$M = \frac{1}{||\boldsymbol{w}||} \quad \Longrightarrow \quad ||\boldsymbol{w}|| = \frac{1}{M} \tag{30}$$

Two implications follow:

- *Maximizing* M (the margin) is equivalent to *minimizing* $||\boldsymbol{w}||$
- The boundary constraint becomes:

Reformulating [OP1] (cont.)

Now, let us set

$$M = \frac{1}{||\boldsymbol{w}||} \quad \Longrightarrow \quad ||\boldsymbol{w}|| = \frac{1}{M} \tag{30}$$

Two implications follow:

- *Maximizing* M (the margin) is equivalent to *minimizing* $||\boldsymbol{w}||$
- The boundary constraint becomes:

Reformulating [OP1] (cont.)

Now, let us set

$$M = \frac{1}{||\boldsymbol{w}||} \quad \Longrightarrow \quad ||\boldsymbol{w}|| = \frac{1}{M} \tag{30}$$

Two implications follow:

- *Maximizing* M (the margin) is equivalent to *minimizing* $||\boldsymbol{w}||$
- The boundary constraint becomes:

$$\widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq M||\boldsymbol{w}||(1 - \xi_n) =$$

Reformulating [OP1] (cont.)

Now, let us set

$$M = \frac{1}{||\boldsymbol{w}||} \quad \Longrightarrow \quad ||\boldsymbol{w}|| = \frac{1}{M} \tag{30}$$

Two implications follow:

- *Maximizing* M (the margin) is equivalent to *minimizing* $||\boldsymbol{w}||$
- The boundary constraint becomes:

$$\widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq M||\boldsymbol{w}||(1 - \xi_n) = M \cdot \frac{1}{M}(1 - \xi_n) =$$

Reformulating [OP1] (cont.)

Now, let us set

$$M = \frac{1}{||\boldsymbol{w}||} \quad \Longrightarrow \quad ||\boldsymbol{w}|| = \frac{1}{M} \tag{30}$$

Two implications follow:

- *Maximizing* M (the margin) is equivalent to *minimizing* $||\boldsymbol{w}||$
- The boundary constraint becomes:

$$\widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq M||\boldsymbol{w}||(1 - \xi_n) = M \cdot \frac{1}{M}(1 - \xi_n) = 1 - \xi_n \tag{31}$$

Given these, we can formulate a new optimization problem OP2 to solve for the support vector classifier.

Optimization problem 2 [OP2]

Optimization problem 2 [OP2]

We can write the optimization problem for a support vector classifier as:

Optimization problem 2 [OP2]

We can write the optimization problem for a support vector classifier as:

$$\min_{w_0, \boldsymbol{w}, \xi} \frac{1}{2}||\boldsymbol{w}||^2$$

Optimization problem 2 [OP2]

We can write the optimization problem for a support vector classifier as:

$$\min_{w_0, \boldsymbol{w}, \xi} \frac{1}{2}||\boldsymbol{w}||^2$$

subject to:

Optimization problem 2 [OP2]

We can write the optimization problem for a support vector classifier as:

$$\min_{w_0, \boldsymbol{w}, \xi} \frac{1}{2}||\boldsymbol{w}||^2$$

$$\text{subject to:} \quad \widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq 1 - \xi_n \quad \forall n = 1, \ldots, N$$

Optimization problem 2 [OP2]

We can write the optimization problem for a support vector classifier as:

$$\min_{w_0, \boldsymbol{w}, \xi} \frac{1}{2}||\boldsymbol{w}||^2$$

$$\text{subject to:} \quad \widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq 1 - \xi_n \quad \forall n = 1, \ldots, N$$

$$\sum_{n=1}^{N} \xi_n \leq B$$

Optimization problem 2 [OP2]

We can write the optimization problem for a support vector classifier as:

$$\begin{aligned} &\min_{w_0, \boldsymbol{w}, \xi} \frac{1}{2}||\boldsymbol{w}||^2 \\ \text{subject to:} \quad &\widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq 1 - \xi_n \quad \forall n = 1, \ldots, N \\ &\sum_{n=1}^{N} \xi_n \leq B \\ &\xi_n \geq 0 \end{aligned} \tag{32}$$

Optimization problem 2 [OP2]

We can write the optimization problem for a support vector classifier as:

$$\begin{aligned} &\min_{w_0, \boldsymbol{w}, \xi} \frac{1}{2}||\boldsymbol{w}||^2 \\ \text{subject to:} \quad &\widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq 1 - \xi_n \quad \forall n = 1, \ldots, N \\ &\sum_{n=1}^{N} \xi_n \leq B \\ &\xi_n \geq 0 \end{aligned} \tag{32}$$

- For computational convenience, we minimize $\frac{1}{2}||\boldsymbol{w}||^2$ (a quadratic programming problem)

Optimization problem 3 [OP3]

Optimization problem 3 [OP3]

The budget constraint can be included in the objective (penalty method) to obtain:

$$\min_{w_0, \boldsymbol{w}, \xi} \left(\frac{1}{2} ||\boldsymbol{w}||^2 + C \sum_{n=1}^{N} \xi_n \right)$$

Optimization problem 3 [OP3]

The budget constraint can be included in the objective (penalty method) to obtain:

$$\min_{w_0, \boldsymbol{w}, \xi} \left(\frac{1}{2} ||\boldsymbol{w}||^2 + C \sum_{n=1}^{N} \xi_n \right)$$

subject to

Optimization problem 3 [OP3]

The budget constraint can be included in the objective (penalty method) to obtain:

$$\min_{w_0, \boldsymbol{w}, \xi} \left(\frac{1}{2} ||\boldsymbol{w}||^2 + C \sum_{n=1}^{N} \xi_n \right)$$
$$\text{subject to} \quad \widetilde{y}_n (w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq 1 - \xi_n \quad \forall n$$

Optimization problem 3 [OP3]

The budget constraint can be included in the objective (penalty method) to obtain:

$$\begin{aligned} \min_{w_0, \boldsymbol{w}, \xi} & \left(\frac{1}{2} ||\boldsymbol{w}||^2 + C \sum_{n=1}^{N} \xi_n \right) \\ \text{subject to} \quad & \widetilde{y}_n (w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq 1 - \xi_n \quad \forall n \\ & \xi_n \geq 0 \end{aligned} \tag{33}$$

Optimization problem 3 [OP3]

The budget constraint can be included in the objective (penalty method) to obtain:

$$\begin{aligned} \min_{w_0, \boldsymbol{w}, \xi} & \left(\frac{1}{2} ||\boldsymbol{w}||^2 + C \sum_{n=1}^{N} \xi_n \right) \\ \text{subject to} \quad & \widetilde{y}_n (w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq 1 - \xi_n \quad \forall n \\ & \xi_n \geq 0 \end{aligned} \tag{33}$$

- In this form, we can see that $C \sum_{n=1}^{N} \xi_n$ serves as an overlap penalty

Optimization problem 3 [OP3]

The budget constraint can be included in the objective (penalty method) to obtain:

$$\begin{aligned} \min_{w_0, \boldsymbol{w}, \xi} & \left(\frac{1}{2} ||\boldsymbol{w}||^2 + C \sum_{n=1}^{N} \xi_n \right) \\ \text{subject to} \quad & \widetilde{y}_n (w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq 1 - \xi_n \quad \forall n \\ & \xi_n \geq 0 \end{aligned} \tag{33}$$

- In this form, we can see that $C \sum_{n=1}^{N} \xi_n$ serves as an overlap penalty
- We can now refer to C as a "cost" parameter here

Optimization problem 3 [OP3]

The budget constraint can be included in the objective (penalty method) to obtain:

$$\begin{aligned} \min_{w_0, \boldsymbol{w}, \xi} & \left(\frac{1}{2} ||\boldsymbol{w}||^2 + C \sum_{n=1}^{N} \xi_n \right) \\ \text{subject to} \quad & \widetilde{y}_n (w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq 1 - \xi_n \quad \forall n \\ & \xi_n \geq 0 \end{aligned} \tag{33}$$

- In this form, we can see that $C \sum_{n=1}^{N} \xi_n$ serves as an overlap penalty
- We can now refer to C as a "cost" parameter here
- As $C \to \infty$,

Optimization problem 3 [OP3]

The budget constraint can be included in the objective (penalty method) to obtain:

$$\begin{aligned} \min_{w_0, \boldsymbol{w}, \xi} & \left(\frac{1}{2} ||\boldsymbol{w}||^2 + C \sum_{n=1}^{N} \xi_n \right) \\ \text{subject to} \quad & \widetilde{y}_n (w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq 1 - \xi_n \quad \forall n \\ & \xi_n \geq 0 \end{aligned} \tag{33}$$

- In this form, we can see that $C \sum_{n=1}^{N} \xi_n$ serves as an overlap penalty
- We can now refer to C as a "cost" parameter here
- As $C \to \infty$,

Optimization problem 3 [OP3]

The budget constraint can be included in the objective (penalty method) to obtain:

$$\begin{aligned} \min_{w_0, \boldsymbol{w}, \xi} & \left(\frac{1}{2} ||\boldsymbol{w}||^2 + C \sum_{n=1}^{N} \xi_n \right) \\ \text{subject to} \quad & \widetilde{y}_n (w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq 1 - \xi_n \quad \forall n \\ & \xi_n \geq 0 \end{aligned} \tag{33}$$

- In this form, we can see that $C \sum_{n=1}^{N} \xi_n$ serves as an overlap penalty
- We can now refer to C as a "cost" parameter here
- As $C \to \infty$, $\xi_n \to 0$
- Thus, a small C allows for more overlap (larger margin), while a large C leads to less overlap (smaller margin)

Method of Lagrange multipliers

Method of Lagrange multipliers

Using the method of Lagrange multipliers we can write the Lagrangian form of the problem [OP3] and minimize that instead.

Method of Lagrange multipliers

Using the method of Lagrange multipliers we can write the Lagrangian form of the problem [OP3] and minimize that instead.

This is done by:

- Rewriting each inequality constraint in the form $g(x) \geq 0$

Method of Lagrange multipliers

Using the method of Lagrange multipliers we can write the Lagrangian form of the problem [OP3] and minimize that instead.

This is done by:

- Rewriting each inequality constraint in the form $g(x) \geq 0$
- Augmenting the objective function by subtracting $\lambda g(x)$, where λ is the Lagrange multiplier

Method of Lagrange multipliers

Using the method of Lagrange multipliers we can write the Lagrangian form of the problem [OP3] and minimize that instead.

This is done by:

- Rewriting each inequality constraint in the form $g(x) \geq 0$
- Augmenting the objective function by subtracting $\lambda g(x)$, where λ is the Lagrange multiplier
- Solving the respective differential equations by setting the gradient of the Lagrangian $\mathcal{L}$ to 0 (w.r.t. each of the unknowns)

Method of Lagrange multipliers on [OP3]

Method of Lagrange multipliers on [OP3]

Given [OP3]:

$$\min_{w_0, \boldsymbol{w}, \xi} \left(\frac{1}{2} ||\boldsymbol{w}||^2 + C \sum_{n=1}^{N} \xi_n \right)$$

Method of Lagrange multipliers on [OP3]

Given [OP3]:

$$\min_{w_0, \boldsymbol{w}, \xi} \left(\frac{1}{2} ||\boldsymbol{w}||^2 + C \sum_{n=1}^{N} \xi_n \right)$$

subject to

Method of Lagrange multipliers on [OP3]

Given [OP3]:

$$\min_{w_0, \boldsymbol{w}, \xi} \left(\frac{1}{2} ||\boldsymbol{w}||^2 + C \sum_{n=1}^{N} \xi_n \right)$$
$$\text{subject to} \quad \widetilde{y}_n (w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq 1 - \xi_n \quad \forall n$$

Method of Lagrange multipliers on [OP3]

Given [OP3]:

$$\begin{aligned} \min_{w_0, \boldsymbol{w}, \xi} \left(\frac{1}{2} ||\boldsymbol{w}||^2 + C \sum_{n=1}^{N} \xi_n \right) \\ \text{subject to} \quad \widetilde{y}_n (w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq 1 - \xi_n \quad \forall n \\ \xi_n \geq 0 \end{aligned} \tag{34}$$

Method of Lagrange multipliers on [OP3]

Given [OP3]:

$$\begin{aligned} \min_{w_0, \boldsymbol{w}, \xi} & \left(\frac{1}{2} ||\boldsymbol{w}||^2 + C \sum_{n=1}^{N} \xi_n \right) \\ \text{subject to} \quad & \widetilde{y}_n (w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq 1 - \xi_n \quad \forall n \\ & \xi_n \geq 0 \end{aligned} \tag{34}$$

we rewrite the constraints as:

Method of Lagrange multipliers on [OP3]

Given [OP3]:

$$\begin{aligned} \min_{w_0, \boldsymbol{w}, \xi} & \left(\frac{1}{2} ||\boldsymbol{w}||^2 + C \sum_{n=1}^{N} \xi_n \right) \\ \text{subject to} \quad & \widetilde{y}_n (w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq 1 - \xi_n \quad \forall n \\ & \xi_n \geq 0 \end{aligned} \tag{34}$$

we rewrite the constraints as:

$$\widetilde{y}_n (w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) - (1 - \xi_n) \quad \geq \quad 0 \quad \forall n \tag{35}$$

Method of Lagrange multipliers on [OP3]

Given [OP3]:

$$\begin{aligned} &\min_{w_0, \boldsymbol{w}, \xi} \left(\frac{1}{2} ||\boldsymbol{w}||^2 + C \sum_{n=1}^{N} \xi_n \right) \\ &\text{subject to} \quad \widetilde{y}_n (w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq 1 - \xi_n \quad \forall n \\ &\qquad\qquad\qquad\qquad\qquad\qquad \xi_n \geq 0 \end{aligned} \tag{34}$$

we rewrite the constraints as:

$$\widetilde{y}_n (w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) - (1 - \xi_n) \geq 0 \quad \forall n \tag{35}$$

$$\xi_n \geq 0 \tag{36}$$

Method of Lagrange multipliers on [OP3]

Given [OP3]:

$$\begin{aligned} \min_{w_0, \boldsymbol{w}, \xi} & \left(\frac{1}{2} ||\boldsymbol{w}||^2 + C \sum_{n=1}^{N} \xi_n \right) \\ \text{subject to} \quad & \widetilde{y}_n (w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq 1 - \xi_n \quad \forall n \\ & \xi_n \geq 0 \end{aligned} \tag{34}$$

we rewrite the constraints as:

$$\widetilde{y}_n (w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) - (1 - \xi_n) \geq 0 \quad \forall n \tag{35}$$

$$\xi_n \geq 0 \tag{36}$$

Let the Lagrange multiplier of Constraint (35) be α_n and the Lagrange multiplier of Constraint (36) be μ_n.

Method of Lagrange multipliers on [OP3]

Given [OP3]:

$$\begin{aligned} \min_{w_0, \boldsymbol{w}, \xi} & \left(\frac{1}{2} ||\boldsymbol{w}||^2 + C \sum_{n=1}^{N} \xi_n \right) \\ \text{subject to} \quad & \widetilde{y}_n (w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) \geq 1 - \xi_n \quad \forall n \\ & \xi_n \geq 0 \end{aligned} \tag{34}$$

we rewrite the constraints as:

$$\widetilde{y}_n (w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) - (1 - \xi_n) \geq 0 \quad \forall n \tag{35}$$

$$\xi_n \geq 0 \tag{36}$$

Let the Lagrange multiplier of Constraint (35) be α_n and the Lagrange multiplier of Constraint (36) be μ_n.

Primal Lagrangian problem

[1] The subscript P denotes that this is the primal problem, as there exists a dual problem that can be maximized.

Primal Lagrangian problem

Then we augment the objective to obtain the **primal Lagrangian** $\mathcal{L}_P$ as:

[1] The subscript P denotes that this is the primal problem, as there exists a dual problem that can be maximized.

Primal Lagrangian problem

Then we augment the objective to obtain the **primal Lagrangian** $\mathcal{L}_P$ as:

$$\mathcal{L}_P = \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N}\alpha_n\left[\widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) - 1 + \xi_n\right] - \sum_{n=1}^{N}\mu_n\xi_n \quad (37)$$

[1] The subscript P denotes that this is the primal problem, as there exists a dual problem that can be maximized.

Primal Lagrangian problem

Then we augment the objective to obtain the **primal Lagrangian** $\mathcal{L}_P$ as:

$$\mathcal{L}_P = \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N}\alpha_n\left[\widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) - 1 + \xi_n\right] - \sum_{n=1}^{N}\mu_n\xi_n \quad (37)$$

To **minimize** $\mathcal{L}_P$, we find the gradient $\nabla\mathcal{L}_P$ (w.r.t. w_0, $\boldsymbol{w}$ and ξ) and set to zero.[1]

$$\nabla_{w_0} = -\sum_{n=1}^{N}\alpha_n\widetilde{y}_n = 0 \quad (38)$$

[1] The subscript P denotes that this is the primal problem, as there exists a dual problem that can be maximized.

Primal Lagrangian problem

Then we augment the objective to obtain the **primal Lagrangian** $\mathcal{L}_P$ as:

$$\mathcal{L}_P = \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N}\alpha_n\left[\widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) - 1 + \xi_n\right] - \sum_{n=1}^{N}\mu_n\xi_n \quad (37)$$

To **minimize** $\mathcal{L}_P$, we find the gradient $\nabla\mathcal{L}_P$ (w.r.t. w_0, $\boldsymbol{w}$ and ξ) and set to zero.[1]

$$\nabla_{w_0} = -\sum_{n=1}^{N}\alpha_n\widetilde{y}_n = 0 \quad (38)$$

$$\nabla_{\boldsymbol{w}} = \boldsymbol{w} - \sum_{n=1}^{N}\alpha_n\widetilde{y}_n\boldsymbol{x}_n = 0 \quad (39)$$

[1] The subscript P denotes that this is the primal problem, as there exists a dual problem that can be maximized.

Primal Lagrangian problem

Then we augment the objective to obtain the **primal Lagrangian** $\mathcal{L}_P$ as:

$$\mathcal{L}_P = \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N}\alpha_n\left[\widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) - 1 + \xi_n\right] - \sum_{n=1}^{N}\mu_n\xi_n \quad (37)$$

To **minimize** $\mathcal{L}_P$, we find the gradient $\nabla\mathcal{L}_P$ (w.r.t. w_0, $\boldsymbol{w}$ and ξ) and set to zero.[1]

$$\nabla_{w_0} = -\sum_{n=1}^{N}\alpha_n\widetilde{y}_n = 0 \quad (38)$$

$$\nabla_{\boldsymbol{w}} = \boldsymbol{w} - \sum_{n=1}^{N}\alpha_n\widetilde{y}_n\boldsymbol{x}_n = 0 \quad (39)$$

$$\nabla_{\xi} = C - \alpha_n - \mu_n = 0 \quad (40)$$

[1] The subscript P denotes that this is the primal problem, as there exists a dual problem that can be maximized.

Optimization problem 4 [OP4]: Dual Lagrangian

Optimization problem 4 [OP4]: Dual Lagrangian

Finally, we can substitute Constraints (38), (39) and (40) into $\mathcal{L}_D$ to obtain the dual Lagrangian objective function, which we **maximize**:

Optimization problem 4 [OP4]: Dual Lagrangian

Finally, we can substitute Constraints (38), (39) and (40) into $\mathcal{L}_D$ to obtain the dual Lagrangian objective function, which we **maximize**:

$$\max \mathcal{L}_D =$$

Optimization problem 4 [OP4]: Dual Lagrangian

Finally, we can substitute Constraints (38), (39) and (40) into $\mathcal{L}_D$ to obtain the dual Lagrangian objective function, which we **maximize**:

$$\max \mathcal{L}_D = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{n'=1}^{N} \alpha_n \alpha_{n'} \widetilde{y}_n y_{n'} \boldsymbol{x}_n^\top \boldsymbol{x}_{n'} \tag{41}$$

Optimization problem 4 [OP4]: Dual Lagrangian

Finally, we can substitute Constraints (38), (39) and (40) into $\mathcal{L}_D$ to obtain the dual Lagrangian objective function, which we **maximize**:

$$\max \mathcal{L}_D = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{n'=1}^{N} \alpha_n \alpha_{n'} \widetilde{y}_n y_{n'} \boldsymbol{x}_n^\top \boldsymbol{x}_{n'} \tag{41}$$

subject to the **Karush-Kuhn-Tucker** (KKT) conditions:

Optimization problem 4 [OP4]: Dual Lagrangian

Finally, we can substitute Constraints (38), (39) and (40) into $\mathcal{L}_D$ to obtain the dual Lagrangian objective function, which we **maximize**:

$$\max \mathcal{L}_D = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{n'=1}^{N} \alpha_n \alpha_{n'} \widetilde{y}_n y_{n'} \boldsymbol{x}_n^\top \boldsymbol{x}_{n'} \tag{41}$$

subject to the **Karush-Kuhn-Tucker** (KKT) conditions:

$$0 \leq \alpha_n \quad \leq \quad C \tag{42}$$

Optimization problem 4 [OP4]: Dual Lagrangian

Finally, we can substitute Constraints (38), (39) and (40) into $\mathcal{L}_D$ to obtain the dual Lagrangian objective function, which we **maximize**:

$$\max \mathcal{L}_D = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{n'=1}^{N} \alpha_n \alpha_{n'} \widetilde{y}_n y_{n'} \boldsymbol{x}_n^\top \boldsymbol{x}_{n'} \tag{41}$$

subject to the **Karush-Kuhn-Tucker** (KKT) conditions:

$$0 \leq \alpha_n \leq C \tag{42}$$

$$\sum_{n=1}^{N} \alpha_n \widetilde{y}_n = 0 \tag{43}$$

Optimization problem 4 [OP4]: Dual Lagrangian

Finally, we can substitute Constraints (38), (39) and (40) into $\mathcal{L}_D$ to obtain the dual Lagrangian objective function, which we **maximize**:

$$\max \mathcal{L}_D = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{n'=1}^{N} \alpha_n \alpha_{n'} \widetilde{y}_n y_{n'} \boldsymbol{x}_n^\top \boldsymbol{x}_{n'} \tag{41}$$

subject to the **Karush-Kuhn-Tucker** (KKT) conditions:

$$0 \leq \alpha_n \leq C \tag{42}$$

$$\sum_{n=1}^{N} \alpha_n \widetilde{y}_n = 0 \tag{43}$$

$$\alpha_n \left[\widetilde{y}_n (w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) - 1 + \xi_n \right] = 0 \tag{44}$$

Optimization problem 4 [OP4]: Dual Lagrangian

Finally, we can substitute Constraints (38), (39) and (40) into $\mathcal{L}_D$ to obtain the dual Lagrangian objective function, which we **maximize**:

$$\max \mathcal{L}_D = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{n'=1}^{N} \alpha_n \alpha_{n'} \widetilde{y}_n y_{n'} \boldsymbol{x}_n^\top \boldsymbol{x}_{n'} \tag{41}$$

subject to the **Karush-Kuhn-Tucker** (KKT) conditions:

$$0 \leq \alpha_n \leq C \tag{42}$$

$$\sum_{n=1}^{N} \alpha_n \widetilde{y}_n = 0 \tag{43}$$

$$\alpha_n \left[\widetilde{y}_n (w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) - 1 + \xi_n \right] = 0 \tag{44}$$

$$\mu_n \xi_n = 0 \tag{45}$$

Optimization problem 4 [OP4]: Dual Lagrangian

Finally, we can substitute Constraints (38), (39) and (40) into $\mathcal{L}_D$ to obtain the dual Lagrangian objective function, which we **maximize**:

$$\max \mathcal{L}_D = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{n'=1}^{N} \alpha_n \alpha_{n'} \widetilde{y}_n y_{n'} \boldsymbol{x}_n^\top \boldsymbol{x}_{n'} \tag{41}$$

subject to the **Karush-Kuhn-Tucker** (KKT) conditions:

$$\begin{aligned}
0 \leq \alpha_n &\leq C && (42)\\
\sum_{n=1}^{N} \alpha_n \widetilde{y}_n &= 0 && (43)\\
\alpha_n \left[\widetilde{y}_n (w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) - 1 + \xi_n\right] &= 0 && (44)\\
\mu_n \xi_n &= 0 && (45)\\
\widetilde{y}_n (w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) - 1 + \xi_n &\geq 0 && (46)
\end{aligned}$$

Support vector classifier solution form

Support vector classifier solution form

Recall constraint (39) of the primal Lagrangian $\mathcal{L}_P$:

Support vector classifier solution form

Recall constraint (39) of the primal Lagrangian $\mathcal{L}_P$:

$$\boldsymbol{w} - \sum_{n=1}^{N} \alpha_n \widetilde{y}_n \boldsymbol{x}_n = 0 \tag{47}$$

This indicates that the coefficients of fitted classifier (hyperplane) are given by:

Support vector classifier solution form

Recall constraint (39) of the primal Lagrangian $\mathcal{L}_P$:

$$\boldsymbol{w} - \sum_{n=1}^{N} \alpha_n \widetilde{y}_n \boldsymbol{x}_n = 0 \tag{47}$$

This indicates that the coefficients of fitted classifier (hyperplane) are given by:

$$\hat{\boldsymbol{w}} = \sum_{n=1}^{N} \hat{\alpha}_n \widetilde{y}_n \boldsymbol{x}_n \tag{48}$$

Support vector classifier solution form

Recall constraint (39) of the primal Lagrangian $\mathcal{L}_P$:

$$\boldsymbol{w} - \sum_{n=1}^{N} \alpha_n \widetilde{y}_n \boldsymbol{x}_n = 0 \tag{47}$$

This indicates that the coefficients of fitted classifier (hyperplane) are given by:

$$\hat{\boldsymbol{w}} = \sum_{n=1}^{N} \hat{\alpha}_n \widetilde{y}_n \boldsymbol{x}_n \tag{48}$$

We observe that:

Support vector classifier solution form

Recall constraint (39) of the primal Lagrangian $\mathcal{L}_P$:

$$\boldsymbol{w} - \sum_{n=1}^{N} \alpha_n \widetilde{y}_n \boldsymbol{x}_n = 0 \tag{47}$$

This indicates that the coefficients of fitted classifier (hyperplane) are given by:

$$\hat{\boldsymbol{w}} = \sum_{n=1}^{N} \hat{\alpha}_n \widetilde{y}_n \boldsymbol{x}_n \tag{48}$$

We observe that:

- When $\hat{\alpha}_n \neq 0$, then $\widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) - 1 + \xi_n = 0$

Support vector classifier solution form

Recall constraint (39) of the primal Lagrangian $\mathcal{L}_P$:

$$\boldsymbol{w} - \sum_{n=1}^{N} \alpha_n \widetilde{y}_n \boldsymbol{x}_n = 0 \tag{47}$$

This indicates that the coefficients of fitted classifier (hyperplane) are given by:

$$\hat{\boldsymbol{w}} = \sum_{n=1}^{N} \hat{\alpha}_n \widetilde{y}_n \boldsymbol{x}_n \tag{48}$$

We observe that:

- When $\hat{\alpha}_n \neq 0$, then $\widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) - 1 + \xi_n = 0$
- Observations for which $\hat{\alpha}$ is nonzero are the **support vectors**

Support vector classifier solution form

Recall constraint (39) of the primal Lagrangian $\mathcal{L}_P$:

$$\boldsymbol{w} - \sum_{n=1}^{N} \alpha_n \widetilde{y}_n \boldsymbol{x}_n = 0 \tag{47}$$

This indicates that the coefficients of fitted classifier (hyperplane) are given by:

$$\hat{\boldsymbol{w}} = \sum_{n=1}^{N} \hat{\alpha}_n \widetilde{y}_n \boldsymbol{x}_n \tag{48}$$

We observe that:

- When $\hat{\alpha}_n \neq 0$, then $\widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) - 1 + \xi_n = 0$
- Observations for which $\hat{\alpha}$ is nonzero are the **support vectors**
  - $\alpha_n < C$,

Support vector classifier solution form

Recall constraint (39) of the primal Lagrangian $\mathcal{L}_P$:

$$\boldsymbol{w} - \sum_{n=1}^{N} \alpha_n \widetilde{y}_n \boldsymbol{x}_n = 0 \tag{47}$$

This indicates that the coefficients of fitted classifier (hyperplane) are given by:

$$\hat{\boldsymbol{w}} = \sum_{n=1}^{N} \hat{\alpha}_n \widetilde{y}_n \boldsymbol{x}_n \tag{48}$$

We observe that:

- When $\hat{\alpha}_n \neq 0$, then $\widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) - 1 + \xi_n = 0$
- Observations for which $\hat{\alpha}$ is nonzero are the **support vectors**
  - $\alpha_n < C$,

Support vector classifier solution form

Recall constraint (39) of the primal Lagrangian $\mathcal{L}_P$:

$$\boldsymbol{w} - \sum_{n=1}^{N} \alpha_n \widetilde{y}_n \boldsymbol{x}_n = 0 \tag{47}$$

This indicates that the coefficients of fitted classifier (hyperplane) are given by:

$$\hat{\boldsymbol{w}} = \sum_{n=1}^{N} \hat{\alpha}_n \widetilde{y}_n \boldsymbol{x}_n \tag{48}$$

We observe that:

- When $\hat{\alpha}_n \neq 0$, then $\widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) - 1 + \xi_n = 0$
- Observations for which $\hat{\alpha}$ is nonzero are the **support vectors**
  - $\alpha_n < C$, $\xi_n = 0$ (observation on margin)
  - $\alpha_n = C$: $\xi_n \leq 1$: (obs. inside margin);

Support vector classifier solution form

Recall constraint (39) of the primal Lagrangian $\mathcal{L}_P$:

$$\boldsymbol{w} - \sum_{n=1}^{N} \alpha_n \widetilde{y}_n \boldsymbol{x}_n = 0 \tag{47}$$

This indicates that the coefficients of fitted classifier (hyperplane) are given by:

$$\hat{\boldsymbol{w}} = \sum_{n=1}^{N} \hat{\alpha}_n \widetilde{y}_n \boldsymbol{x}_n \tag{48}$$

We observe that:

- When $\hat{\alpha}_n \neq 0$, then $\widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) - 1 + \xi_n = 0$
- Observations for which $\hat{\alpha}$ is nonzero are the **support vectors**
  - $\alpha_n < C$, $\xi_n = 0$ (observation on margin)
  - $\alpha_n = C$: $\xi_n \leq 1$: (obs. inside margin);

Support vector classifier solution form

Recall constraint (39) of the primal Lagrangian $\mathcal{L}_P$:

$$\boldsymbol{w} - \sum_{n=1}^{N} \alpha_n \widetilde{y}_n \boldsymbol{x}_n = 0 \tag{47}$$

This indicates that the coefficients of fitted classifier (hyperplane) are given by:

$$\hat{\boldsymbol{w}} = \sum_{n=1}^{N} \hat{\alpha}_n \widetilde{y}_n \boldsymbol{x}_n \tag{48}$$

We observe that:

- When $\hat{\alpha}_n \neq 0$, then $\widetilde{y}_n(w_0 + \boldsymbol{w}^\top \boldsymbol{x}_n) - 1 + \xi_n = 0$
- Observations for which $\hat{\alpha}$ is nonzero are the **support vectors**
  - $\alpha_n < C$, $\xi_n = 0$ (observation on margin)
  - $\alpha_n = C$: $\xi_n \leq 1$: (obs. inside margin);
    $\xi_n > 1$ (obs. lie on the wrong side of hyperplane)

Estimating the intercept

Estimating the intercept

The decision boundary (hyperplane) of the fitted SVC is given by:

Estimating the intercept

The decision boundary (hyperplane) of the fitted SVC is given by:

$$f(\boldsymbol{x}; \hat{w}_0, \hat{\boldsymbol{w}}) = \hat{w}_0 + \hat{\boldsymbol{w}}^\top \boldsymbol{x}$$

Estimating the intercept

The decision boundary (hyperplane) of the fitted SVC is given by:

$$f(\boldsymbol{x}; \hat{w}_0, \hat{\boldsymbol{w}}) = \hat{w}_0 + \hat{\boldsymbol{w}}^\top \boldsymbol{x} = \hat{w}_0 + \sum_{n=1}^{N} \hat{\alpha}_n \widetilde{y}_n \boldsymbol{x}_n^\top \boldsymbol{x} \tag{49}$$

Estimating the intercept

The decision boundary (hyperplane) of the fitted SVC is given by:

$$f(\boldsymbol{x}; \hat{w}_0, \hat{\boldsymbol{w}}) = \hat{w}_0 + \hat{\boldsymbol{w}}^\top \boldsymbol{x} = \hat{w}_0 + \sum_{n=1}^{N} \hat{\alpha}_n \widetilde{y}_n \boldsymbol{x}_n^\top \boldsymbol{x} \tag{49}$$

The intercept is estimated as:

Estimating the intercept

The decision boundary (hyperplane) of the fitted SVC is given by:

$$f(\boldsymbol{x}; \hat{w}_0, \hat{\boldsymbol{w}}) = \hat{w}_0 + \hat{\boldsymbol{w}}^\top \boldsymbol{x} = \hat{w}_0 + \sum_{n=1}^{N} \hat{\alpha}_n \widetilde{y}_n \boldsymbol{x}_n^\top \boldsymbol{x} \tag{49}$$

The intercept is estimated as:

$$\hat{w}_0 = \frac{1}{|\mathcal{M}|} \sum_{n \in \mathcal{M}} \left(\widetilde{y}_n - \sum_{m \in \mathcal{S}} \hat{\alpha}_m \widetilde{y}_m \boldsymbol{x}_m^\top \boldsymbol{x}_n \right) \tag{50}$$

Estimating the intercept

The decision boundary (hyperplane) of the fitted SVC is given by:

$$f(\boldsymbol{x}; \hat{w}_0, \hat{\boldsymbol{w}}) = \hat{w}_0 + \hat{\boldsymbol{w}}^\top \boldsymbol{x} = \hat{w}_0 + \sum_{n=1}^{N} \hat{\alpha}_n \widetilde{y}_n \boldsymbol{x}_n^\top \boldsymbol{x} \tag{49}$$

The intercept is estimated as:

$$\hat{w}_0 = \frac{1}{|\mathcal{M}|} \sum_{n \in \mathcal{M}} \left(\widetilde{y}_n - \sum_{m \in \mathcal{S}} \hat{\alpha}_m \widetilde{y}_m \boldsymbol{x}_m^\top \boldsymbol{x}_n \right) \tag{50}$$

where:

Estimating the intercept

The decision boundary (hyperplane) of the fitted SVC is given by:

$$f(\boldsymbol{x}; \hat{w}_0, \hat{\boldsymbol{w}}) = \hat{w}_0 + \hat{\boldsymbol{w}}^\top \boldsymbol{x} = \hat{w}_0 + \sum_{n=1}^{N} \hat{\alpha}_n \widetilde{y}_n \boldsymbol{x}_n^\top \boldsymbol{x} \tag{49}$$

The intercept is estimated as:

$$\hat{w}_0 = \frac{1}{|\mathcal{M}|} \sum_{n \in \mathcal{M}} \left(\widetilde{y}_n - \sum_{m \in \mathcal{S}} \hat{\alpha}_m \widetilde{y}_m \boldsymbol{x}_m^\top \boldsymbol{x}_n \right) \tag{50}$$

where:

- $\mathcal{M}$: set of points where $0 < \alpha_n < C$ (on margin)

Estimating the intercept

The decision boundary (hyperplane) of the fitted SVC is given by:

$$f(\boldsymbol{x}; \hat{w}_0, \hat{\boldsymbol{w}}) = \hat{w}_0 + \hat{\boldsymbol{w}}^\top \boldsymbol{x} = \hat{w}_0 + \sum_{n=1}^{N} \hat{\alpha}_n \widetilde{y}_n \boldsymbol{x}_n^\top \boldsymbol{x} \tag{49}$$

The intercept is estimated as:

$$\hat{w}_0 = \frac{1}{|\mathcal{M}|} \sum_{n \in \mathcal{M}} \left(\widetilde{y}_n - \sum_{m \in \mathcal{S}} \hat{\alpha}_m \widetilde{y}_m \boldsymbol{x}_m^\top \boldsymbol{x}_n \right) \tag{50}$$

where:

- $\mathcal{M}$: set of points where $0 < \alpha_n < C$ (on margin)
- $\mathcal{S}$: set of support vectors ($\alpha_m \neq 0$)

Support vector classifier as inner product

Support vector classifier as inner product

Recall the dual problem defining the support vector classifier:

Support vector classifier as inner product

Recall the dual problem defining the support vector classifier:

$\mathcal{L}_D =$

Support vector classifier as inner product

Recall the dual problem defining the support vector classifier:

$$\mathcal{L}_D = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{n'=1}^{N} \alpha_n \alpha_{n'} \widetilde{y}_n y_{n'} \boldsymbol{x}_n^\top \boldsymbol{x}_{n'}$$

Support vector classifier as inner product

Recall the dual problem defining the support vector classifier:

$$\mathcal{L}_D = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{n'=1}^{N} \alpha_n \alpha_{n'} \widetilde{y}_n y_{n'} \boldsymbol{x}_n^\top \boldsymbol{x}_{n'} =$$

Support vector classifier as inner product

Recall the dual problem defining the support vector classifier:

$$\mathcal{L}_D = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{n'=1}^{N} \alpha_n \alpha_{n'} \widetilde{y}_n y_{n'} \boldsymbol{x}_n^\top \boldsymbol{x}_{n'} = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{n'=1}^{N} \alpha_n \alpha_{n'} \widetilde{y}_n y_{n'} \langle \boldsymbol{x}_n, \boldsymbol{x}_{n'} \rangle \tag{51}$$

Support vector classifier as inner product

Recall the dual problem defining the support vector classifier:

$$\mathcal{L}_D = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{n'=1}^{N} \alpha_n \alpha_{n'} \widetilde{y}_n y_{n'} \boldsymbol{x}_n^\top \boldsymbol{x}_{n'} = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{n'=1}^{N} \alpha_n \alpha_{n'} \widetilde{y}_n y_{n'} \langle \boldsymbol{x}_n, \boldsymbol{x}_{n'} \rangle \tag{51}$$

We observed that the fitted coefficients of the classifier have the form:

Support vector classifier as inner product

Recall the dual problem defining the support vector classifier:

$$\mathcal{L}_D = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{n'=1}^{N} \alpha_n \alpha_{n'} \widetilde{y}_n y_{n'} \boldsymbol{x}_n^\top \boldsymbol{x}_{n'} = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{n'=1}^{N} \alpha_n \alpha_{n'} \widetilde{y}_n y_{n'} \langle \boldsymbol{x}_n, \boldsymbol{x}_{n'} \rangle \tag{51}$$

We observed that the fitted coefficients of the classifier have the form:

$$\hat{\boldsymbol{w}} = \sum_{n=1}^{N} \hat{\alpha}_n \widetilde{y}_n \boldsymbol{x}_n \tag{52}$$

Support vector classifier as inner product

Recall the dual problem defining the support vector classifier:

$$\mathcal{L}_D = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{n'=1}^{N} \alpha_n \alpha_{n'} \widetilde{y}_n y_{n'} \boldsymbol{x}_n^\top \boldsymbol{x}_{n'} = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{n'=1}^{N} \alpha_n \alpha_{n'} \widetilde{y}_n y_{n'} \langle \boldsymbol{x}_n, \boldsymbol{x}_{n'} \rangle \tag{51}$$

We observed that the fitted coefficients of the classifier have the form:

$$\hat{\boldsymbol{w}} = \sum_{n=1}^{N} \hat{\alpha}_n \widetilde{y}_n \boldsymbol{x}_n \tag{52}$$

Which means that the estimated classifier is given by:

Support vector classifier as inner product

Recall the dual problem defining the support vector classifier:

$$\mathcal{L}_D = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{n'=1}^{N} \alpha_n \alpha_{n'} \widetilde{y}_n y_{n'} \boldsymbol{x}_n^\top \boldsymbol{x}_{n'} = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{n'=1}^{N} \alpha_n \alpha_{n'} \widetilde{y}_n y_{n'} \langle \boldsymbol{x}_n, \boldsymbol{x}_{n'} \rangle \tag{51}$$

We observed that the fitted coefficients of the classifier have the form:

$$\hat{\boldsymbol{w}} = \sum_{n=1}^{N} \hat{\alpha}_n \widetilde{y}_n \boldsymbol{x}_n \tag{52}$$

Which means that the estimated classifier is given by:

$$\hat{f}(\boldsymbol{x}) = \hat{w}_0 + \hat{\boldsymbol{w}}^\top \boldsymbol{x} \tag{53}$$

Support vector classifier as inner product

Recall the dual problem defining the support vector classifier:

$$\mathcal{L}_D = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{n'=1}^{N} \alpha_n \alpha_{n'} \widetilde{y}_n y_{n'} \boldsymbol{x}_n^\top \boldsymbol{x}_{n'} = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{n'=1}^{N} \alpha_n \alpha_{n'} \widetilde{y}_n y_{n'} \langle \boldsymbol{x}_n, \boldsymbol{x}_{n'} \rangle \tag{51}$$

We observed that the fitted coefficients of the classifier have the form:

$$\hat{\boldsymbol{w}} = \sum_{n=1}^{N} \hat{\alpha}_n \widetilde{y}_n \boldsymbol{x}_n \tag{52}$$

Which means that the estimated classifier is given by:

$$\hat{f}(\boldsymbol{x}) = \hat{w}_0 + \hat{\boldsymbol{w}}^\top \boldsymbol{x} \tag{53}$$

$$= \hat{w}_0 + \left(\sum_{n=1}^{N} \hat{\alpha}_n \widetilde{y}_n \boldsymbol{x}_n \right)^\top \boldsymbol{x} \tag{54}$$

Support vector classifier as inner product

Recall the dual problem defining the support vector classifier:

$$\mathcal{L}_D = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{n'=1}^{N} \alpha_n \alpha_{n'} \widetilde{y}_n y_{n'} \boldsymbol{x}_n^\top \boldsymbol{x}_{n'} = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{n'=1}^{N} \alpha_n \alpha_{n'} \widetilde{y}_n y_{n'} \langle \boldsymbol{x}_n, \boldsymbol{x}_{n'} \rangle \tag{51}$$

We observed that the fitted coefficients of the classifier have the form:

$$\hat{\boldsymbol{w}} = \sum_{n=1}^{N} \hat{\alpha}_n \widetilde{y}_n \boldsymbol{x}_n \tag{52}$$

Which means that the estimated classifier is given by:

$$\begin{aligned} \hat{f}(\boldsymbol{x}) &= \hat{w}_0 + \hat{\boldsymbol{w}}^\top \boldsymbol{x} && (53) \\ &= \hat{w}_0 + \left(\sum_{n=1}^{N} \hat{\alpha}_n \widetilde{y}_n \boldsymbol{x}_n \right)^\top \boldsymbol{x} && (54) \\ &= \hat{w}_0 + \sum_{n=1}^{N} \hat{\alpha}_n \widetilde{y}_n \boldsymbol{x}^\top \boldsymbol{x}_n \end{aligned}$$

Support vector classifier as inner product

Recall the dual problem defining the support vector classifier:

$$\mathcal{L}_D = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{n'=1}^{N} \alpha_n \alpha_{n'} \widetilde{y}_n y_{n'} \boldsymbol{x}_n^\top \boldsymbol{x}_{n'} = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{n'=1}^{N} \alpha_n \alpha_{n'} \widetilde{y}_n y_{n'} \langle \boldsymbol{x}_n, \boldsymbol{x}_{n'} \rangle \tag{51}$$

We observed that the fitted coefficients of the classifier have the form:

$$\hat{\boldsymbol{w}} = \sum_{n=1}^{N} \hat{\alpha}_n \widetilde{y}_n \boldsymbol{x}_n \tag{52}$$

Which means that the estimated classifier is given by:

$$\begin{aligned} \hat{f}(\boldsymbol{x}) &= \hat{w}_0 + \hat{\boldsymbol{w}}^\top \boldsymbol{x} && (53) \\ &= \hat{w}_0 + \left(\sum_{n=1}^{N} \hat{\alpha}_n \widetilde{y}_n \boldsymbol{x}_n \right)^\top \boldsymbol{x} && (54) \\ &= \hat{w}_0 + \sum_{n=1}^{N} \hat{\alpha}_n \widetilde{y}_n \boldsymbol{x}^\top \boldsymbol{x}_n = \hat{w}_0 + \sum_{n=1}^{N} \hat{\alpha}_n \widetilde{y}_n \langle \boldsymbol{x}, \boldsymbol{x}_n \rangle && (55) \end{aligned}$$

Nonlinear decision boundaries

Nonlinear decision boundaries

Suppose that even with a relaxed margin, a linear decision boundary performs poorly on a certain dataset?

Nonlinear decision boundaries

Suppose that even with a relaxed margin, a linear decision boundary performs poorly on a certain dataset?

Figure: A case where a linear decision boundary performs poorly

Nonlinear decision boundaries

Suppose that even with a relaxed margin, a linear decision boundary performs poorly on a certain dataset?

Figure: A case where a linear decision boundary performs poorly

Nonlinear decision boundaries

Suppose that even with a relaxed margin, a linear decision boundary performs poorly on a certain dataset?

Figure: A case where a linear decision boundary performs poorly

How can we address this using a support vector classifier?

Expanding the feature space for nonlinear boundaries

Expanding the feature space for nonlinear boundaries

We can address nonlinearity by projecting the feature space into a larger dimensional space in which the boundary becomes linear.

Expanding the feature space for nonlinear boundaries

We can address nonlinearity by projecting the feature space into a larger dimensional space in which the boundary becomes linear.

Consider the hyperplane function (where $D = 2$):

$$f(\boldsymbol{x}) = w_0 + \boldsymbol{w}^\top \boldsymbol{x} = w_0 + w_1 x_1 + w_2 x_2 \tag{56}$$

Expanding the feature space for nonlinear boundaries

We can address nonlinearity by projecting the feature space into a larger dimensional space in which the boundary becomes linear.

Consider the hyperplane function (where $D = 2$):

$$f(\boldsymbol{x}) = w_0 + \boldsymbol{w}^\top \boldsymbol{x} = w_0 + w_1 x_1 + w_2 x_2 \tag{56}$$

We could perform a quadratic transformation:

Expanding the feature space for nonlinear boundaries

We can address nonlinearity by projecting the feature space into a larger dimensional space in which the boundary becomes linear.

Consider the hyperplane function (where $D = 2$):

$$f(\boldsymbol{x}) = w_0 + \boldsymbol{w}^\top \boldsymbol{x} = w_0 + w_1 x_1 + w_2 x_2 \tag{56}$$

We could perform a quadratic transformation:

$$f^*(\boldsymbol{x}) = w_0 + \boldsymbol{w}^{*\top} \phi(\boldsymbol{x}) \tag{57}$$

Expanding the feature space for nonlinear boundaries

We can address nonlinearity by projecting the feature space into a larger dimensional space in which the boundary becomes linear.

Consider the hyperplane function (where $D = 2$):

$$f(\boldsymbol{x}) = w_0 + \boldsymbol{w}^\top \boldsymbol{x} = w_0 + w_1 x_1 + w_2 x_2 \tag{56}$$

We could perform a quadratic transformation:

$$f^*(\boldsymbol{x}) = w_0 + \boldsymbol{w}^{*\top} \phi(\boldsymbol{x}) \tag{57}$$

where $\phi(\boldsymbol{x}) = \begin{pmatrix} x_1 \\ x_2 \\ x_1^2 \\ x_2^2 \end{pmatrix}$

Expanding the feature space for nonlinear boundaries

We can address nonlinearity by projecting the feature space into a larger dimensional space in which the boundary becomes linear.

Consider the hyperplane function (where $D = 2$):

$$f(\boldsymbol{x}) = w_0 + \boldsymbol{w}^\top \boldsymbol{x} = w_0 + w_1 x_1 + w_2 x_2 \tag{56}$$

We could perform a quadratic transformation:

$$f^*(\boldsymbol{x}) = w_0 + \boldsymbol{w}^{*\top} \phi(\boldsymbol{x}) \tag{57}$$

where $\phi(\boldsymbol{x}) = \begin{pmatrix} x_1 \\ x_2 \\ x_1^2 \\ x_2^2 \end{pmatrix}$ (feature extractor) and $\boldsymbol{w}'$ is a 4×1 vector.

Generalizing the support vector classifier

Generalizing the support vector classifier

- We can generalize the inner product in $\hat{f}(\boldsymbol{x})$ as a basis expansion in $\boldsymbol{x}$ in order to obtain nonlinear decision boundaries.

Generalizing the support vector classifier

- We can generalize the inner product in $\hat{f}(\boldsymbol{x})$ as a basis expansion in $\boldsymbol{x}$ in order to obtain nonlinear decision boundaries.

Generalizing the support vector classifier

- We can generalize the inner product in $\hat{f}(\boldsymbol{x})$ as a basis expansion in $\boldsymbol{x}$ in order to obtain nonlinear decision boundaries.
- Let $\phi(\boldsymbol{x})$ be a vector of functions/transformations of x.

Generalizing the support vector classifier

- We can generalize the inner product in $\hat{f}(\boldsymbol{x})$ as a basis expansion in $\boldsymbol{x}$ in order to obtain nonlinear decision boundaries.
- Let $\phi(\boldsymbol{x})$ be a vector of functions/transformations of x.
- Then we can substitute this into the Lagrangian:

Generalizing the support vector classifier

- We can generalize the inner product in $\hat{f}(\boldsymbol{x})$ as a basis expansion in $\boldsymbol{x}$ in order to obtain nonlinear decision boundaries.
- Let $\phi(\boldsymbol{x})$ be a vector of functions/transformations of x.
- Then we can substitute this into the Lagrangian:

Generalizing the support vector classifier

- We can generalize the inner product in $\hat{f}(\boldsymbol{x})$ as a basis expansion in $\boldsymbol{x}$ in order to obtain nonlinear decision boundaries.
- Let $\phi(\boldsymbol{x})$ be a vector of functions/transformations of x.
- Then we can substitute this into the Lagrangian:

$$\mathcal{L}_D =$$

Generalizing the support vector classifier

- We can generalize the inner product in $\hat{f}(\boldsymbol{x})$ as a basis expansion in $\boldsymbol{x}$ in order to obtain nonlinear decision boundaries.
- Let $\phi(\boldsymbol{x})$ be a vector of functions/transformations of x.
- Then we can substitute this into the Lagrangian:

$$\mathcal{L}_D = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{n'=1}^{N} \alpha_n \alpha_{n'} \widetilde{y}_n y_{n'} \langle \phi(\boldsymbol{x}_n), \phi(\boldsymbol{x}_{n'}) \rangle \tag{58}$$

- The resulting classifier then has the form:

$$\hat{f}(x) = \hat{w}_0 + \sum_{n=1}^{N} \hat{\alpha}_n \widetilde{y}_n \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}_n) \rangle \tag{59}$$

Generalizing the support vector classifier

- We can generalize the inner product in $\hat{f}(\boldsymbol{x})$ as a basis expansion in $\boldsymbol{x}$ in order to obtain nonlinear decision boundaries.
- Let $\phi(\boldsymbol{x})$ be a vector of functions/transformations of x.
- Then we can substitute this into the Lagrangian:

$$\mathcal{L}_D = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{n'=1}^{N} \alpha_n \alpha_{n'} \widetilde{y}_n y_{n'} \langle \phi(\boldsymbol{x}_n), \phi(\boldsymbol{x}_{n'}) \rangle \tag{58}$$

- The resulting classifier then has the form:

$$\hat{f}(x) = \hat{w}_0 + \sum_{n=1}^{N} \hat{\alpha}_n \widetilde{y}_n \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}_n) \rangle \tag{59}$$

- This extension of the support vector classifier to allow for nonlinearity produces the **support vector machine (SVM)**.

The kernel trick

The kernel trick

More generally, we specify the basis expansion of the inner product of the inputs via Mercer **kernel** functions:

The kernel trick

More generally, we specify the basis expansion of the inner product of the inputs via Mercer **kernel** functions:

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle \tag{60}$$

The kernel trick

More generally, we specify the basis expansion of the inner product of the inputs via Mercer **kernel** functions:

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle \tag{60}$$

The linear kernel is simply:

The kernel trick

More generally, we specify the basis expansion of the inner product of the inputs via Mercer **kernel** functions:

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle \tag{60}$$

The linear kernel is simply:

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x}, \boldsymbol{x}' \rangle \tag{61}$$

The kernel trick

More generally, we specify the basis expansion of the inner product of the inputs via Mercer **kernel** functions:

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle \tag{60}$$

The linear kernel is simply:

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x}, \boldsymbol{x}' \rangle \tag{61}$$

which produces a linear boundary (suport vector classifier).

The kernel trick

More generally, we specify the basis expansion of the inner product of the inputs via Mercer **kernel** functions:

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle \tag{60}$$

The linear kernel is simply:

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x}, \boldsymbol{x}' \rangle \tag{61}$$

which produces a linear boundary (suport vector classifier).

Other common inner-product kernel choices are:

Polynomial of degree d:

The kernel trick

More generally, we specify the basis expansion of the inner product of the inputs via Mercer **kernel** functions:

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle \tag{60}$$

The linear kernel is simply:

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x}, \boldsymbol{x}' \rangle \tag{61}$$

which produces a linear boundary (suport vector classifier).

Other common inner-product kernel choices are:

$$\textbf{Polynomial of degree } \boldsymbol{d}\textbf{:} \quad \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') \quad = \quad (1 + \langle \boldsymbol{x}, \boldsymbol{x}' \rangle)^d \tag{62}$$

The kernel trick

More generally, we specify the basis expansion of the inner product of the inputs via Mercer **kernel** functions:

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle \tag{60}$$

The linear kernel is simply:

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x}, \boldsymbol{x}' \rangle \tag{61}$$

which produces a linear boundary (suport vector classifier).

Other common inner-product kernel choices are:

$$\textbf{Polynomial of degree } \boldsymbol{d}\textbf{:} \quad \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = (1 + \langle \boldsymbol{x}, \boldsymbol{x}' \rangle)^d \tag{62}$$

Radial basis function (RBF):

The kernel trick

More generally, we specify the basis expansion of the inner product of the inputs via Mercer **kernel** functions:

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle \tag{60}$$

The linear kernel is simply:

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x}, \boldsymbol{x}' \rangle \tag{61}$$

which produces a linear boundary (suport vector classifier).

Other common inner-product kernel choices are:

$$\textbf{Polynomial of degree } \boldsymbol{d}\textbf{:} \quad \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = (1 + \langle \boldsymbol{x}, \boldsymbol{x}' \rangle)^d \tag{62}$$

$$\textbf{Radial basis function (RBF):} \quad \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\gamma ||\boldsymbol{x} - \boldsymbol{x}'||^2) \tag{63}$$

The kernel trick

More generally, we specify the basis expansion of the inner product of the inputs via Mercer **kernel** functions:

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle \tag{60}$$

The linear kernel is simply:

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x}, \boldsymbol{x}' \rangle \tag{61}$$

which produces a linear boundary (suport vector classifier).

Other common inner-product kernel choices are:

$$\textbf{Polynomial of degree } \boldsymbol{d}\textbf{:} \quad \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = (1 + \langle \boldsymbol{x}, \boldsymbol{x}' \rangle)^d \tag{62}$$

$$\textbf{Radial basis function (RBF):} \quad \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\gamma ||\boldsymbol{x} - \boldsymbol{x}'||^2) \tag{63}$$

Neural network (sigmoid):

The kernel trick

More generally, we specify the basis expansion of the inner product of the inputs via Mercer **kernel** functions:

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle \tag{60}$$

The linear kernel is simply:

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x}, \boldsymbol{x}' \rangle \tag{61}$$

which produces a linear boundary (suport vector classifier).

Other common inner-product kernel choices are:

$$\textbf{Polynomial of degree } \boldsymbol{d}\textbf{:} \quad \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = (1 + \langle \boldsymbol{x}, \boldsymbol{x}' \rangle)^d \tag{62}$$

$$\textbf{Radial basis function (RBF):} \quad \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\gamma ||\boldsymbol{x} - \boldsymbol{x}'||^2) \tag{63}$$

$$\textbf{Neural network (sigmoid):} \quad \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \tanh(1 + \kappa_1 \langle \boldsymbol{x}, \boldsymbol{x}' \rangle + \kappa_2) \tag{64}$$

SVM general solution form

SVM general solution form

The support vector machine is also the solution to the Lagrangian dual problem:

SVM general solution form

The support vector machine is also the solution to the Lagrangian dual problem:

$$\max \mathcal{L}_D = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{n'=1}^{N} \alpha_n \alpha_{n'} \widetilde{y}_n y_{n'} \mathcal{K}(\boldsymbol{x}_n, \boldsymbol{x}_{n'}) \tag{65}$$

SVM general solution form

The support vector machine is also the solution to the Lagrangian dual problem:

$$\max \mathcal{L}_D = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{n'=1}^{N} \alpha_n \alpha_{n'} \widetilde{y}_n y_{n'} \mathcal{K}(\boldsymbol{x}_n, \boldsymbol{x}_{n'}) \tag{65}$$

where $\mathcal{K}(\boldsymbol{x}_n, \boldsymbol{x}_{n'})$ is a **Mercer kernel** specifying the inner product expansion in the feature space.

SVM general solution form

The support vector machine is also the solution to the Lagrangian dual problem:

$$\max \mathcal{L}_D = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{n'=1}^{N} \alpha_n \alpha_{n'} \widetilde{y}_n y_{n'} \mathcal{K}(\boldsymbol{x}_n, \boldsymbol{x}_{n'}) \tag{65}$$

where $\mathcal{K}(\boldsymbol{x}_n, \boldsymbol{x}_{n'})$ is a **Mercer kernel** specifying the inner product expansion in the feature space.

Therefore the SVM solution has the form:

$$\hat{f}(\boldsymbol{x}) = \sum_{n=1}^{N} \hat{\alpha}_n \widetilde{y}_n \mathcal{K}(\boldsymbol{x}_n, \boldsymbol{x}) + \hat{w}_0 \tag{66}$$

SVM general solution form

The support vector machine is also the solution to the Lagrangian dual problem:

$$\max \mathcal{L}_D = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{n'=1}^{N} \alpha_n \alpha_{n'} \widetilde{y}_n y_{n'} \mathcal{K}(\boldsymbol{x}_n, \boldsymbol{x}_{n'}) \tag{65}$$

where $\mathcal{K}(\boldsymbol{x}_n, \boldsymbol{x}_{n'})$ is a **Mercer kernel** specifying the inner product expansion in the feature space.

Therefore the SVM solution has the form:

$$\hat{f}(\boldsymbol{x}) = \sum_{n=1}^{N} \hat{\alpha}_n \widetilde{y}_n \mathcal{K}(\boldsymbol{x}_n, \boldsymbol{x}) + \hat{w}_0 \tag{66}$$

The beauty of the kernel transformation is that the inner products in $\mathcal{K}(\boldsymbol{x}_n, \boldsymbol{x})$ can be cheaply computed without explicitly specifying high-dimensional basis function expansions $\phi(\boldsymbol{x})$.

More on the radial kernel

More on the radial kernel

The radial kernel transformation has a localizing effect on the distance measurement.

More on the radial kernel

The radial kernel transformation has a localizing effect on the distance measurement.

- As defined in Eq. (63), it is a Gaussian function

More on the radial kernel

The radial kernel transformation has a localizing effect on the distance measurement.

- As defined in Eq. (63), it is a Gaussian function
- The localization effect increases with γ (leading to a more flexible decision boundary)

More on the radial kernel

The radial kernel transformation has a localizing effect on the distance measurement.

- As defined in Eq. (63), it is a Gaussian function
- The localization effect increases with γ (leading to a more flexible decision boundary)

Example: polynomial kernel of degree 2

Example: polynomial kernel of degree 2

Consider a case with two inputs x_1 and x_2.

Example: polynomial kernel of degree 2

Consider a case with two inputs x_1 and x_2. We can write the degree-2 polynomial kernel as follows:

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = (1 + \langle \boldsymbol{x}, \boldsymbol{x}' \rangle)^2$$

Example: polynomial kernel of degree 2

Consider a case with two inputs x_1 and x_2. We can write the degree-2 polynomial kernel as follows:

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = (1 + \langle \boldsymbol{x}, \boldsymbol{x}' \rangle)^2 = (1 + x_1 x_1' + x_2 x_2')^2$$

Example: polynomial kernel of degree 2

Consider a case with two inputs x_1 and x_2. We can write the degree-2 polynomial kernel as follows:

$$\begin{aligned}\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') &= (1 + \langle \boldsymbol{x}, \boldsymbol{x}' \rangle)^2 = (1 + x_1 x_1' + x_2 x_2')^2 \\ &= 1 + 2x_1 x_1' + 2x_2 x_2' + (x_1 x_1')^2 + (x_2 x_2')^2 + 2x_1 x_1' x_2 x_2'\end{aligned} \tag{67}$$

Example: polynomial kernel of degree 2

Consider a case with two inputs x_1 and x_2. We can write the degree-2 polynomial kernel as follows:

$$\begin{aligned}\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') &= (1 + \langle \boldsymbol{x}, \boldsymbol{x}' \rangle)^2 = (1 + x_1 x_1' + x_2 x_2')^2 \\ &= 1 + 2x_1 x_1' + 2x_2 x_2' + (x_1 x_1')^2 + (x_2 x_2')^2 + 2x_1 x_1' x_2 x_2'\end{aligned} \tag{67}$$

Explicitly, this is an inner product of basis functions $\phi(\boldsymbol{x})$:

Example: polynomial kernel of degree 2

Consider a case with two inputs x_1 and x_2. We can write the degree-2 polynomial kernel as follows:

$$\begin{aligned}\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') &= (1 + \langle \boldsymbol{x}, \boldsymbol{x}' \rangle)^2 = (1 + x_1 x_1' + x_2 x_2')^2 \\ &= 1 + 2x_1 x_1' + 2x_2 x_2' + (x_1 x_1')^2 + (x_2 x_2')^2 + 2x_1 x_1' x_2 x_2'\end{aligned} \tag{67}$$

Explicitly, this is an inner product of basis functions $\phi(\boldsymbol{x})$:

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') =$$

Example: polynomial kernel of degree 2

Consider a case with two inputs x_1 and x_2. We can write the degree-2 polynomial kernel as follows:

$$\begin{aligned}\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') &= (1 + \langle \boldsymbol{x}, \boldsymbol{x}' \rangle)^2 = (1 + x_1 x_1' + x_2 x_2')^2 \\ &= 1 + 2x_1 x_1' + 2x_2 x_2' + (x_1 x_1')^2 + (x_2 x_2')^2 + 2x_1 x_1' x_2 x_2'\end{aligned} \tag{67}$$

Explicitly, this is an inner product of basis functions $\phi(\boldsymbol{x})$:

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \begin{pmatrix} 1 & x_1\sqrt{2} & x_2\sqrt{2} & x_1^2 & x_2^2 & x_1 x_2 \sqrt{2} \end{pmatrix}$$

Example: polynomial kernel of degree 2

Consider a case with two inputs x_1 and x_2. We can write the degree-2 polynomial kernel as follows:

$$\begin{aligned}\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') &= (1 + \langle \boldsymbol{x}, \boldsymbol{x}' \rangle)^2 = (1 + x_1 x_1' + x_2 x_2')^2 \\ &= 1 + 2x_1 x_1' + 2x_2 x_2' + (x_1 x_1')^2 + (x_2 x_2')^2 + 2x_1 x_1' x_2 x_2'\end{aligned} \tag{67}$$

Explicitly, this is an inner product of basis functions $\phi(\boldsymbol{x})$:

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \begin{pmatrix} 1 & x_1\sqrt{2} & x_2\sqrt{2} & x_1^2 & x_2^2 & x_1 x_2 \sqrt{2} \end{pmatrix} \begin{pmatrix} 1 \\ x_1'\sqrt{2} \\ x_2'\sqrt{2} \\ x_1'^2 \\ x_2'^2 \\ x_1' x_1' \sqrt{2} \end{pmatrix}$$

Example: polynomial kernel of degree 2

Consider a case with two inputs x_1 and x_2. We can write the degree-2 polynomial kernel as follows:

$$\begin{aligned}\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') &= (1 + \langle \boldsymbol{x}, \boldsymbol{x}' \rangle)^2 = (1 + x_1 x_1' + x_2 x_2')^2 \\ &= 1 + 2x_1 x_1' + 2x_2 x_2' + (x_1 x_1')^2 + (x_2 x_2')^2 + 2x_1 x_1' x_2 x_2'\end{aligned} \tag{67}$$

Explicitly, this is an inner product of basis functions $\phi(\boldsymbol{x})$:

$$\begin{aligned}\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') &= \begin{pmatrix} 1 & x_1\sqrt{2} & x_2\sqrt{2} & x_1^2 & x_2^2 & x_1 x_2 \sqrt{2} \end{pmatrix} \begin{pmatrix} 1 \\ x_1'\sqrt{2} \\ x_2'\sqrt{2} \\ x_1'^2 \\ x_2'^2 \\ x_1' x_1' \sqrt{2} \end{pmatrix} \\ &= \phi(\boldsymbol{x})^\top \phi(\boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle\end{aligned} \tag{68}$$

Example: polynomial kernel of degree 2

Consider a case with two inputs x_1 and x_2. We can write the degree-2 polynomial kernel as follows:

$$\begin{aligned}\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') &= (1 + \langle \boldsymbol{x}, \boldsymbol{x}' \rangle)^2 = (1 + x_1 x_1' + x_2 x_2')^2 \\ &= 1 + 2x_1 x_1' + 2x_2 x_2' + (x_1 x_1')^2 + (x_2 x_2')^2 + 2x_1 x_1' x_2 x_2'\end{aligned} \tag{67}$$

Explicitly, this is an inner product of basis functions $\phi(\boldsymbol{x})$:

$$\begin{aligned}\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') &= \begin{pmatrix} 1 & x_1\sqrt{2} & x_2\sqrt{2} & x_1^2 & x_2^2 & x_1 x_2 \sqrt{2} \end{pmatrix} \begin{pmatrix} 1 \\ x_1'\sqrt{2} \\ x_2'\sqrt{2} \\ x_1'^2 \\ x_2'^2 \\ x_1' x_1' \sqrt{2} \end{pmatrix} \\ &= \phi(\boldsymbol{x})^\top \phi(\boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle\end{aligned} \tag{68}$$

Example: SVM degree-2 polynomial (quadratic) kernel

Example: SVM degree-2 polynomial (quadratic) kernel

Figure: A quadratic decision boundary produced on the earlier dataset using an SVM with a quadratic kernel

Example: SVM degree-2 polynomial (quadratic) kernel

Figure: A quadratic decision boundary produced on the earlier dataset using an SVM with a quadratic kernel

Extensions of SVM

Extensions of SVM

Multiple classes:

Extensions of SVM

Multiple classes:
SVM is essentially a binary classification tool, but it can handle multi-class assignments via the following approaches:

- One-versus-one classification

Extensions of SVM

Multiple classes:
SVM is essentially a binary classification tool, but it can handle multi-class assignments via the following approaches:

- One-versus-one classification
- One-versus-all classification

Extensions of SVM

Multiple classes:
SVM is essentially a binary classification tool, but it can handle multi-class assignments via the following approaches:

- One-versus-one classification
- One-versus-all classification

Extensions of SVM

Multiple classes:
SVM is essentially a binary classification tool, but it can handle multi-class assignments via the following approaches:

- One-versus-one classification
- One-versus-all classification

Regression:

Extensions of SVM

Multiple classes:
SVM is essentially a binary classification tool, but it can handle multi-class assignments via the following approaches:

- One-versus-one classification
- One-versus-all classification

Regression:
SVM can also be used for regression (**SVM regression**):

Extensions of SVM

Multiple classes:
SVM is essentially a binary classification tool, but it can handle multi-class assignments via the following approaches:

- One-versus-one classification
- One-versus-all classification

Regression:
SVM can also be used for regression (**SVM regression**):

- Here, we estimate a hyperplane in which the values of the fitted function $\hat{f}(\boldsymbol{x})$ minimize residuals only for the support vectors beyond the margin

Sparse vector machines:

Extensions of SVM

Multiple classes:
SVM is essentially a binary classification tool, but it can handle multi-class assignments via the following approaches:

- One-versus-one classification
- One-versus-all classification

Regression:
SVM can also be used for regression (**SVM regression**):

- Here, we estimate a hyperplane in which the values of the fitted function $\hat{f}(\boldsymbol{x})$ minimize residuals only for the support vectors beyond the margin

Sparse vector machines:

Extensions of SVM

Multiple classes:
SVM is essentially a binary classification tool, but it can handle multi-class assignments via the following approaches:

- One-versus-one classification
- One-versus-all classification

Regression:
SVM can also be used for regression (**SVM regression**):

- Here, we estimate a hyperplane in which the values of the fitted function $\hat{f}(\boldsymbol{x})$ minimize residuals only for the support vectors beyond the margin

Sparse vector machines:

- Employ regularization for better performance: L1VM and L2VM

Summary

- Separating hyperplane: perfectly partitions a set of observations into two

Summary

- Separating hyperplane: perfectly partitions a set of observations into two
  - Has $D - 1$ dimensions in D-dimensional space

Summary

- Separating hyperplane: perfectly partitions a set of observations into two
  - Has $D - 1$ dimensions in D-dimensional space
- Maximal margin classifier:

Summary

- Separating hyperplane: perfectly partitions a set of observations into two
  - Has $D-1$ dimensions in D-dimensional space
- Maximal margin classifier:
  - margin M: perpendicular distance from support vector(s) to hyperplane

Summary

- Separating hyperplane: perfectly partitions a set of observations into two
  - Has $D-1$ dimensions in D-dimensional space
- Maximal margin classifier:
  - margin M: perpendicular distance from support vector(s) to hyperplane
  - partitions dataset by optimizing M

Summary

- Separating hyperplane: perfectly partitions a set of observations into two
  - Has $D-1$ dimensions in D-dimensional space
- Maximal margin classifier:
  - margin M: perpendicular distance from support vector(s) to hyperplane
  - partitions dataset by optimizing M
- Support vector (soft-margin) classifier:

Summary

- Separating hyperplane: perfectly partitions a set of observations into two
  - Has $D-1$ dimensions in D-dimensional space
- Maximal margin classifier:
  - margin M: perpendicular distance from support vector(s) to hyperplane
  - partitions dataset by optimizing M
- Support vector (soft-margin) classifier:
  - Introduces slack variables ξ_n to allow for overlap

Summary

- Separating hyperplane: perfectly partitions a set of observations into two
  - Has $D-1$ dimensions in D-dimensional space
- Maximal margin classifier:
  - margin M: perpendicular distance from support vector(s) to hyperplane
  - partitions dataset by optimizing M
- Support vector (soft-margin) classifier:
  - Introduces slack variables ξ_n to allow for overlap
  - Cost parameter C moderates amount of overlap (tolerable bias)

Summary

- Separating hyperplane: perfectly partitions a set of observations into two
  - Has $D-1$ dimensions in D-dimensional space
- Maximal margin classifier:
  - margin M: perpendicular distance from support vector(s) to hyperplane
  - partitions dataset by optimizing M
- Support vector (soft-margin) classifier:
  - Introduces slack variables ξ_n to allow for overlap
  - Cost parameter C moderates amount of overlap (tolerable bias)
- SVC problem in its dual form gives the hyperplane coefficients as an inner product in feature space

Summary

- Separating hyperplane: perfectly partitions a set of observations into two
  - Has $D-1$ dimensions in D-dimensional space
- Maximal margin classifier:
  - margin M: perpendicular distance from support vector(s) to hyperplane
  - partitions dataset by optimizing M
- Support vector (soft-margin) classifier:
  - Introduces slack variables ξ_n to allow for overlap
  - Cost parameter C moderates amount of overlap (tolerable bias)
- SVC problem in its dual form gives the hyperplane coefficients as an inner product in feature space
- **Kernelizing** the inner product in the SVC solution results in the **support vector machine**

Reading and resources

Reading and resources

- Reading: **PMLI** 17.3; **ESL** 12.2-3; **ISLR** 9.1-4;

Reading and resources

- Reading: **PMLI** 17.3; **ESL** 12.2-3; **ISLR** 9.1-4;
- Scikit-Learn documentation and tutorial: **here**

Reading and resources

- Reading: **PMLI** 17.3; **ESL** 12.2-3; **ISLR** 9.1-4;
- Scikit-Learn documentation and tutorial: **here**
- Plotting nice decision boundaries in R: **here**

Reading and resources

- Reading: **PMLI** 17.3; **ESL** 12.2-3; **ISLR** 9.1-4;
- Scikit-Learn documentation and tutorial: **here**
- Plotting nice decision boundaries in R: **here**
- Jupyter notebook on Moodle.