Machine Learning for Official Statistics & SDGs

Support Vector Machine

UNITED NATIONS SIAP
Statistical Institute for Asia and the Pacific

[Understanding SVM]

Support Vector Machine (SVM) is a classifier

[Understanding SVM]

Support Vector Machine (SVM) is a classifier

- One of the best "out of the box" classifier

[Understanding SVM]

Support Vector Machine (SVM) is a classifier

- One of the best "out of the box" classifier
- Lots of refinements

[Understanding SVM]

Support Vector Machine (SVM) is a classifier

- One of the best "out of the box" classifier
- Lots of refinements
- Based on two very different ideas

[Understanding SVM]

Support Vector Machine (SVM) is a classifier

- One of the best "out of the box" classifier
- Lots of refinements
- Based on two very different ideas
  - Dimension augmentation

[Understanding SVM]

Support Vector Machine (SVM) is a classifier

- One of the best "out of the box" classifier
- Lots of refinements
- Based on two very different ideas
  - Dimension augmentation
  - Maximal margin classifier

[Understanding SVM]

Support Vector Machine (SVM) is a classifier

- One of the best "out of the box" classifier
- Lots of refinements
- Based on two very different ideas
  - Dimension augmentation
  - Maximal margin classifier
- Technical details require advanced mathematics

[Understanding SVM]

Support Vector Machine (SVM) is a classifier

- One of the best "out of the box" classifier
- Lots of refinements
- Based on two very different ideas
  - Dimension augmentation
  - Maximal margin classifier
- Technical details require advanced mathematics

$\hookrightarrow$ Focus on seminal ideas and intuitions

[The Vapnik–Chervonenkis dimension]

[The Vapnik–Chervonenkis dimension]

- The *Vapnik–Chervonenkis* (VC) dimension measures the complexity that can be learned by a classifier

[The Vapnik–Chervonenkis dimension]

- The *Vapnik–Chervonenkis* (VC) dimension measures the complexity that can be learned by a classifier
- The VC dimension is defined as the maximum number of points allowing a separation by a linear classifier

[The Vapnik–Chervonenkis dimension]

- The *Vapnik–Chervonenkis* (VC) dimension measures the complexity that can be learned by a classifier
- The VC dimension is defined as the maximum number of points allowing a separation by a linear classifier

$\hookrightarrow$ $VC(f)$ = # points always separable in the current space

[The Vapnik–Chervonenkis dimension]

- The *Vapnik–Chervonenkis* (VC) dimension measures the complexity that can be learned by a classifier
- The VC dimension is defined as the maximum number of points allowing a separation by a linear classifier

$\hookrightarrow$ $VC(f)$ = # points always separable in the current space

- Useful to measure the complexity of a particular model in a given space

[The Vapnik–Chervonenkis dimension]

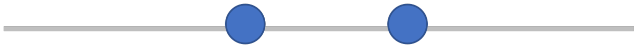
[The Vapnik–Chervonenkis dimension]

In 1-D, there are 4 situation with 2 points:

Both orange

[The Vapnik–Chervonenkis dimension]

In 1-D, there are 4 situation with 2 points:



Both blue

[The Vapnik–Chervonenkis dimension]

In 1-D, there are 4 situation with 2 points:

One orange, one blue

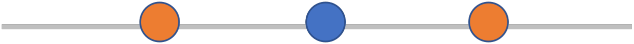
[The Vapnik–Chervonenkis dimension]

In 1-D, there are 4 situation with 2 points:

One blue, one orange

[The Vapnik–Chervonenkis dimension]

What happens with 3 points?

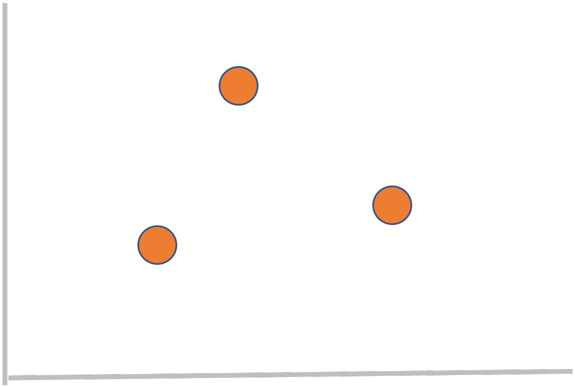


A situation where no linear classifier works!
$\hookrightarrow VC(f_{lin}) = 2$

[The Vapnik–Chervonenkis dimension]

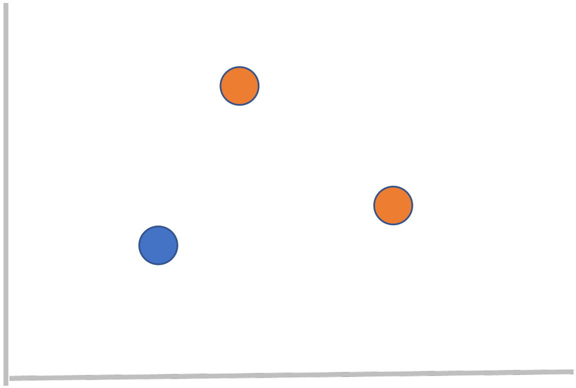
[The Vapnik–Chervonenkis dimension]

In 2-D, there are $2^3 = 8$ different situations with 3 points:



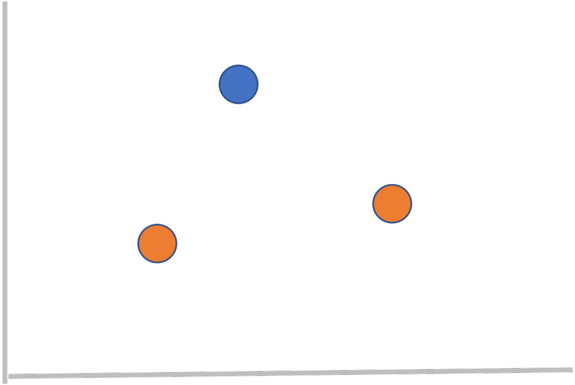
[The Vapnik–Chervonenkis dimension]

In 2-D, there are $2^3 = 8$ different situations with 3 points:



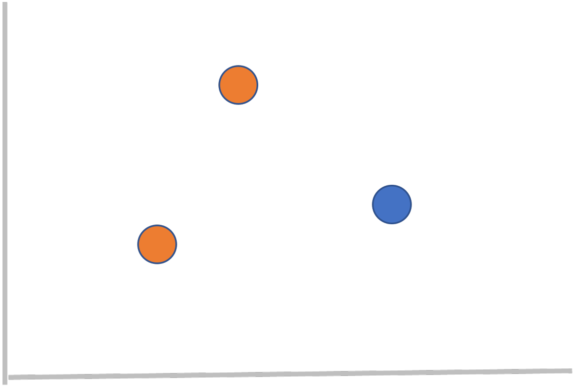
[The Vapnik–Chervonenkis dimension]

In 2-D, there are $2^3 = 8$ different situations with 3 points:



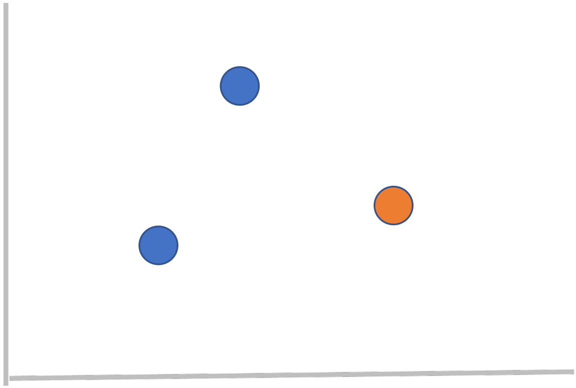
[The Vapnik–Chervonenkis dimension]

In 2-D, there are $2^3 = 8$ different situations with 3 points:



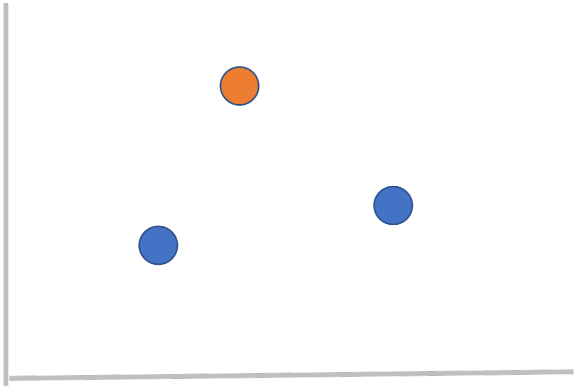
[The Vapnik–Chervonenkis dimension]

In 2-D, there are $2^3 = 8$ different situations with 3 points:



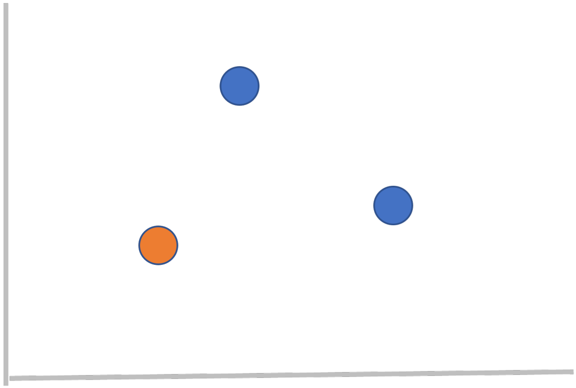
[The Vapnik–Chervonenkis dimension]

In 2-D, there are $2^3 = 8$ different situations with 3 points:



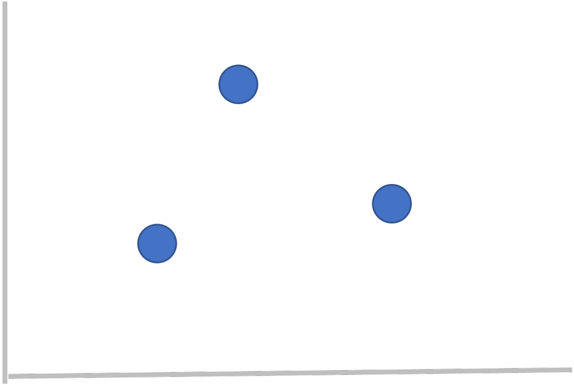
[The Vapnik–Chervonenkis dimension]

In 2-D, there are $2^3 = 8$ different situations with 3 points:



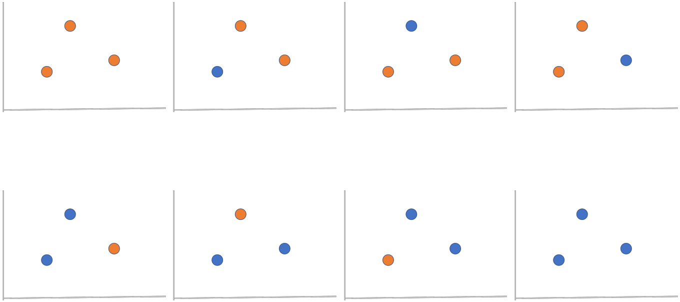
[THE VAPNIK–CHERVONENKIS DIMENSION]

In 2-D, there are $2^3 = 8$ different situations with 3 points:



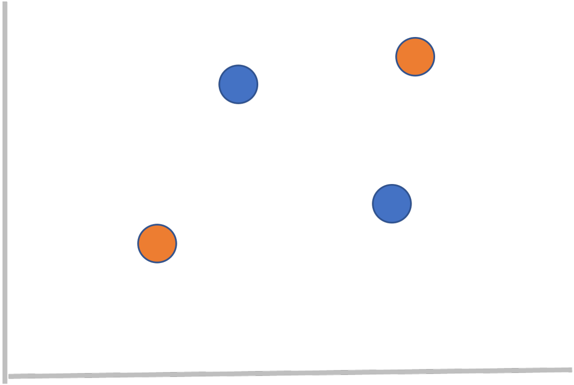
[The Vapnik–Chervonenkis dimension]

In 2-D, there are $2^3 = 8$ different situations with 3 points:



[The Vapnik–Chervonenkis dimension]

What happens with 4 points?



A situation where no linear classifier works!

$\hookrightarrow VC(f_{lin}) = 3$

[Interpretation of the VC dimension]

[Interpretation of the VC dimension]

- $VC(f)$ measures the complexity of the classifier f, linear or not

[Interpretation of the VC dimension]

- $VC(f)$ measures the complexity of the classifier f, linear or not
- A highly non-linear classifier will always be able to separate a high number of points

[Interpretation of the VC dimension]

- $VC(f)$ measures the complexity of the classifier f, linear or not
- A highly non-linear classifier will always be able to separate a high number of points

  It can be shown that:

[Interpretation of the VC dimension]

- $VC(f)$ measures the complexity of the classifier f, linear or not
- A highly non-linear classifier will always be able to separate a high number of points

  It can be shown that:

  - Classifiers with a **high** VC dimension will be better at learning and fitting the training data set

[Interpretation of the VC dimension]

- $VC(f)$ measures the complexity of the classifier f, linear or not
- A highly non-linear classifier will always be able to separate a high number of points

  It can be shown that:
  - Classifiers with a **high** VC dimension will be better at learning and fitting the training data set

  $\hookrightarrow$ **High** risk of over-fitting

[Interpretation of the VC dimension]

- $VC(f)$ measures the complexity of the classifier f, linear or not
- A highly non-linear classifier will always be able to separate a high number of points

  It can be shown that:
  - Classifiers with a **high** VC dimension will be better at learning and fitting the training data set

  $\hookrightarrow$ **High** risk of over-fitting
  - Classifiers with a **low** VC dimension make more errors on the training data set, but may be better in prediction

[Interpretation of the VC dimension]

- $VC(f)$ measures the complexity of the classifier f, linear or not
- A highly non-linear classifier will always be able to separate a high number of points

  It can be shown that:
  - Classifiers with a **high** VC dimension will be better at learning and fitting the training data set

  $\hookrightarrow$ **High** risk of over-fitting
  - Classifiers with a **low** VC dimension make more errors on the training data set, but may be better in prediction

  $\hookrightarrow$ **Low** risk of over-fitting

[Interpretation of the VC dimension]

- $VC(f)$ measures the complexity of the classifier f, linear or not
- A highly non-linear classifier will always be able to separate a high number of points

  It can be shown that:
  - Classifiers with a **high** VC dimension will be better at learning and fitting the training data set

  $\hookrightarrow$ **High** risk of over-fitting
  - Classifiers with a **low** VC dimension make more errors on the training data set, but may be better in prediction

  $\hookrightarrow$ **Low** risk of over-fitting
- New version of the **Bias-Variance** trade off!

[Implications of the VC dimension]

[Implications of the VC dimension]

- The Vapnik–Chervonenkis inequality (*simplified*):

$$\epsilon_{\mathit{validation}}(f) \leq \epsilon_{\mathit{train}}(f) + G(VC(f), m)$$

[Implications of the VC dimension]

- The Vapnik–Chervonenkis inequality (*simplified*):

$$\epsilon_{validation}(f) \leq \epsilon_{train}(f) + G(VC(f), m)$$

Where:

[Implications of the VC dimension]

- The Vapnik–Chervonenkis inequality (*simplified*):

$$\epsilon_{\,validation}(f) \leq \epsilon_{\,train}(f) + G(VC(f), m)$$

Where:

- $\epsilon(\cdot)$ is the error rate,

[Implications of the VC dimension]

- The Vapnik–Chervonenkis inequality (*simplified*):

$$\epsilon_{\,validation}(f) \leq \epsilon_{\,train}(f) + G(VC(f), m)$$

Where:

- $\epsilon(\cdot)$ is the error rate,
- m is the nb of training sets and

[Implications of the VC dimension]

- The Vapnik–Chervonenkis inequality (*simplified*):

$$\epsilon_{validation}(f) \leq \epsilon_{train}(f) + G(VC(f), m)$$

Where:

- $\epsilon(\cdot)$ is the error rate,
- m is the nb of training sets and
- $G(\cdot)$ a function **decreasing** with m (obviously)

[Implications of the VC dimension]

- The Vapnik–Chervonenkis inequality (*simplified*):

$$\epsilon_{\,validation}(f) \leq \epsilon_{\,train}(f) + G(VC(f), m)$$

Where:

- $\epsilon(\cdot)$ is the error rate,
- m is the nb of training sets and
- $G(\cdot)$ a function **decreasing** with m (obviously)
  increasing with VC !!

[Implications of the VC dimension]

- The Vapnik–Chervonenkis inequality (*simplified*):

$$\epsilon_{\,validation}(f) \leq \epsilon_{\,train}(f) + G(VC(f), m)$$

Where:

- $\epsilon(\cdot)$ is the error rate,
- m is the nb of training sets and
- $G(\cdot)$ a function **decreasing** with m (obviously)
  increasing with VC !!

$\hookrightarrow$ The higher $VC(f)$ the greater the difference between performance on the *training* and *validation* sample

[Implications of the VC dimension]

- The Vapnik–Chervonenkis inequality (*simplified*):

$$\epsilon_{\,validation}(f) \leq \epsilon_{\,train}(f) + G(VC(f), m)$$

Where:

- $\epsilon(\cdot)$ is the error rate,
- m is the nb of training sets and
- $G(\cdot)$ a function **decreasing** with m (obviously)
  increasing with VC !!

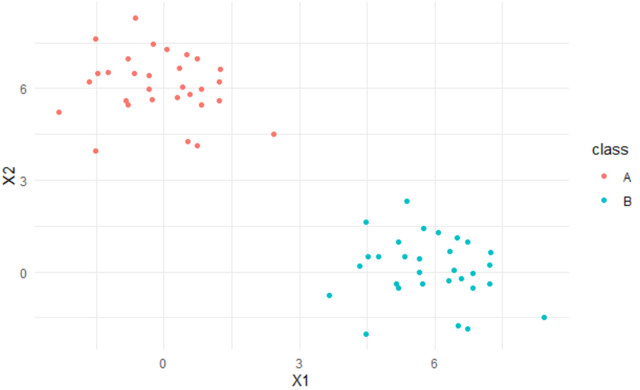
↪ The higher $VC(f)$ the greater the difference between performance on the *training* and *validation* sample

↪ Better to use simple models (low VC)

[Linear Boundaries in a simple example]

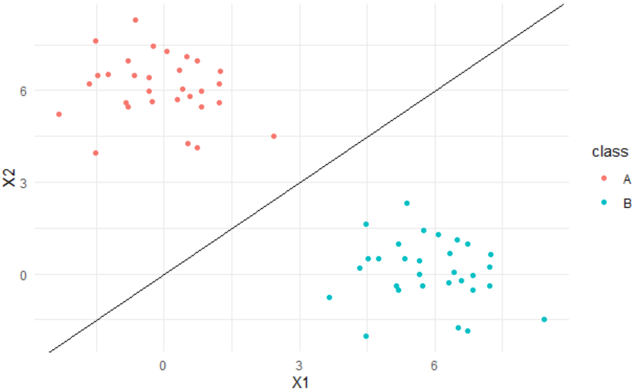
[Linear boundaries in a simple example]

Take the following hypothetical situation



[Linear Boundaries in a simple example]

Take the following hypothetical situation

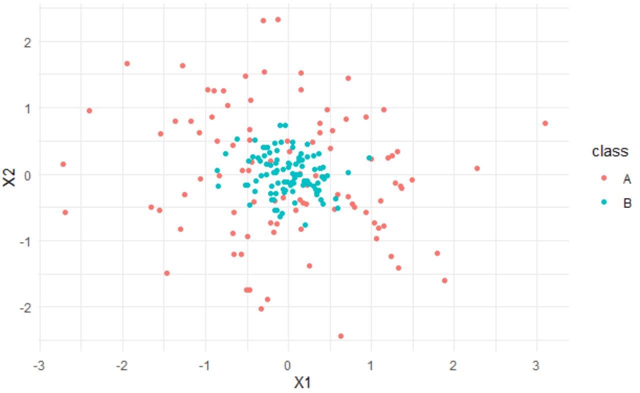


↪ It is easy to *separate* the two classes with a linear classifier

[Linear Boundaries in a simple example]

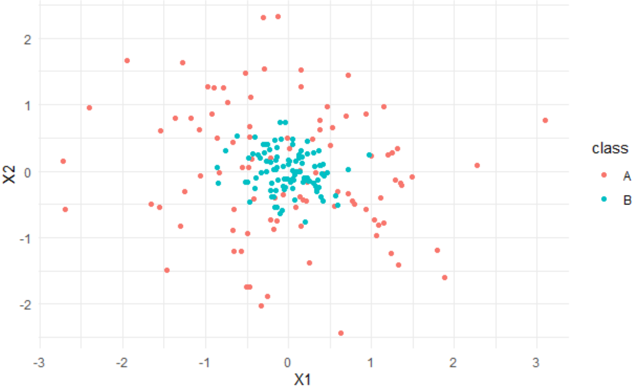
[Linear Boundaries in a simple example]

But in this situation?



[Linear Boundaries in a simple example]

But in this situation?



↪ There is no way to *separate* the two classes with a linear classifier!!

[Linear Boundaries in a simple example]

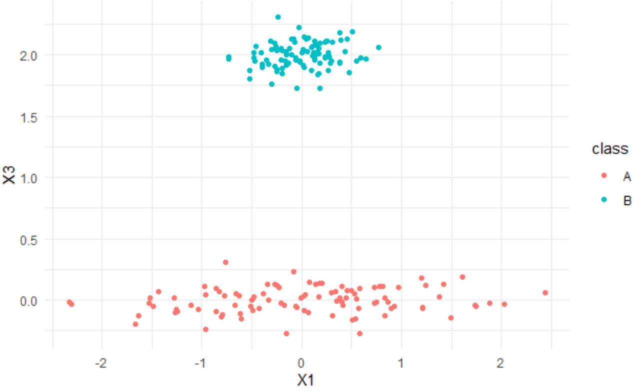
[Linear Boundaries in a simple example]

Unless...

[Dimension augmentation]

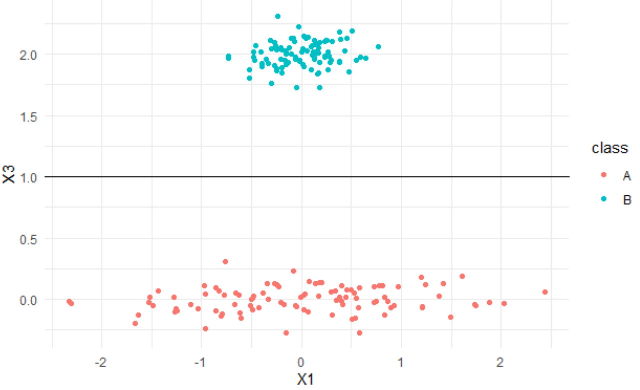
[Dimension augmentation]

Unless one adds a third dimension!



[DIMENSION AUGMENTATION]

Unless one adds a third dimension!

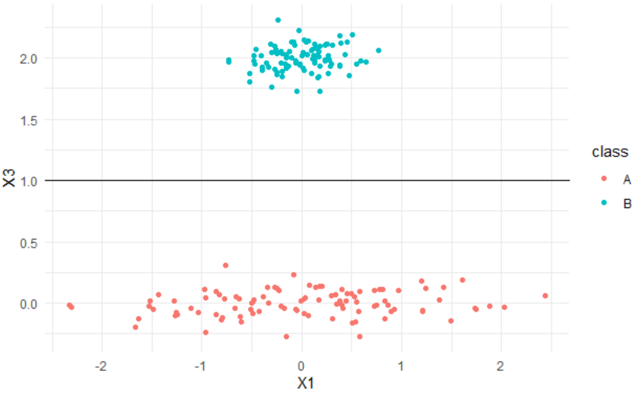


$\hookrightarrow$ A linear classifier can *separate* the two classes in an *augmented* space (third dimension)

[Maximal Margin Classifier]

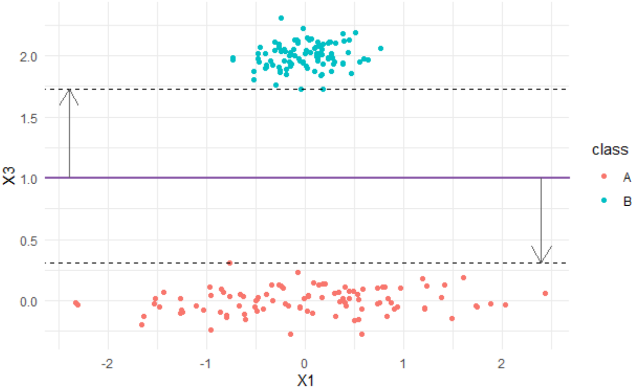
[Maximal Margin Classifier]

The maximum margin classifier is defined as:



[Maximal Margin Classifier]

The maximum margin classifier is defined as:



$\hookrightarrow$ M is the margin.

[Maximal Margin Classifier]

[Maximal Margin Classifier]

The optimization problem to find this margin is :
with $(\beta_0 + x'\beta)$ *defining the boundary (line or hyperplane)*

$$\begin{aligned} Max_{\beta_0,\beta} \quad & M \\ \textit{subject to}: \ y_i(\beta_0 + x'\beta) \quad & \geq \quad M \end{aligned}$$

and $y_i = 1 \ or -1$ depending on the class of observation i

[Maximal Margin Classifier]

The optimization problem to find this margin is :
↪ This can be relaxed

$$\begin{aligned} Max_{\beta_0,\beta} \quad & M \\ subject\ to:\ y_i(\beta_0 + x'\beta) \quad & \geq \quad M - \xi \end{aligned}$$

where ξ is the *slack* parameter

[Soft margin]

Introducing *soft margins* is one idea implemented in SVM

[Soft margin]

Introducing *soft margins* is one idea implemented in SVM

The slack parameter will play an important role

[Soft margin]

Introducing *soft margins* is one idea implemented in SVM

The slack parameter will play an important role

- A small *slack* allows few exceptions

[Soft margin]

Introducing *soft margins* is one idea implemented in SVM

The slack parameter will play an important role

- A small *slack* allows few exceptions

↪ Boundary (classifier) sensitive to outliers

[Soft margin]

Introducing *soft margins* is one idea implemented in SVM

The slack parameter will play an important role

- A small *slack* allows few exceptions

$\hookrightarrow$ Boundary (classifier) sensitive to outliers

- A large *slack* allows lots of errors and misclassifications

[Soft margin]

Introducing *soft margins* is one idea implemented in SVM

The slack parameter will play an important role

- A small *slack* allows few exceptions

$\hookrightarrow$ Boundary (classifier) sensitive to outliers

- A large *slack* allows lots of errors and misclassifications

The optimization problem is equivalent to:

$$\begin{aligned} Min_{\beta_0,\beta} \quad & \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i \\ subject\ to: \quad & y_i(\beta_0 + x'\beta) \geq M - \xi_i \end{aligned} \tag{1}$$

[Soft margin]

Introducing *soft margins* is one idea implemented in SVM

The slack parameter will play an important role

- A small *slack* allows few exceptions

↪ Boundary (classifier) sensitive to outliers

- A large *slack* allows lots of errors and misclassifications

The optimization problem is equivalent to:

$$Min_{\beta_0,\beta} \quad \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i \tag{1}$$

$$subject\ to: \quad y_i(\beta_0 + x'\beta) \geq M - \xi_i$$

↪ SVM is a trade-off between maximal M and cost C of errors due to ξ

[The *kernel trick*]

The previous optimization method applies to an (*augmented*) space with separable classes

[The *kernel trick*]

The previous optimization method applies to an (*augmented*) space with separable classes

- How do we determine this *augmented* space?

[THE *kernel trick*]

The previous optimization method applies to an (*augmented*) space with separable classes

- How do we determine this *augmented* space?

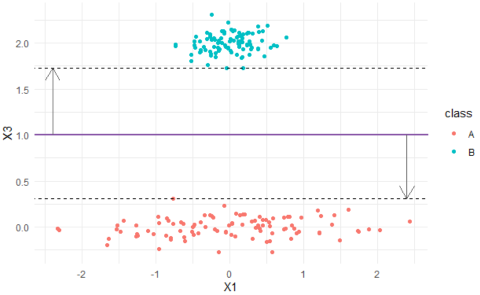
  How to find $\phi(\cdot)$ so that $X_3 = \phi(X_1, X_2)$ separates classes?

[THE *kernel trick*]

The previous optimization method applies to an (*augmented*) space with separable classes

- How do we determine this *augmented* space?

  How to find $\phi(\cdot)$ so that $X_3 = \phi(X_1, X_2)$ separates classes?

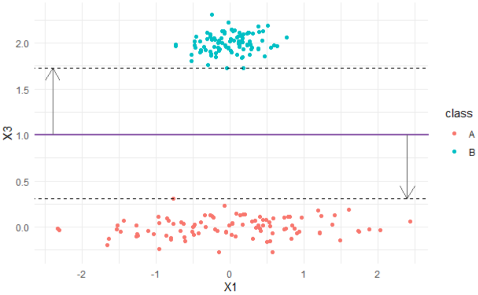


[The *kernel trick*]

The previous optimization method applies to an (*augmented*) space with separable classes

- How do we determine this *augmented* space?

  How to find $\phi(\cdot)$ so that $X_3 = \phi(X_1, X_2)$ separates classes?



$\hookrightarrow$ The *kernel trick!*

[The *kernel trick*]

Finding $\phi(\cdot)$ is difficult and sometimes infeasible!

[The *kernel trick*]

Finding $\phi(\cdot)$ is difficult and sometimes infeasible!

- One can rewrite the optimization problem

[The *kernel trick*]

Finding $\phi(\cdot)$ is difficult and sometimes infeasible!

- One can rewrite the optimization problem

$\hookrightarrow$ Transform the *Xs* through kernels $K(\cdot)$

[THE *kernel trick*]

Finding $\phi(\cdot)$ is difficult and sometimes infeasible!

- One can rewrite the optimization problem

$\hookrightarrow$ Transform the *X*s through kernels $K(\cdot)$

- A *kernel* quantifies the similarity of two observations (*inner product*)

[The *kernel trick*]

Finding $\phi(\cdot)$ is difficult and sometimes infeasible!

- One can rewrite the optimization problem

$\hookrightarrow$ Transform the Xs through kernels $K(\cdot)$

- A *kernel* quantifies the similarity of two observations (*inner product*)
- The optimization can be written **only** in terms of $K(X_i, X_j)$!

[The *kernel trick*]

Finding $\phi(\cdot)$ is difficult and sometimes infeasible!

- One can rewrite the optimization problem

$\hookrightarrow$ Transform the *Xs* through kernels $K(\cdot)$

- A *kernel* quantifies the similarity of two observations (*inner product*)
- The optimization can be written **only** in terms of $K(X_i, X_j)$!

$\hookrightarrow$ The solution is non linear in the original space

[HYPERPARAMETERS IN SVM]

Several hyperparameters have to be selected:

[Hyperparameters in SVM]

Several hyperparameters have to be selected:

- The slack parameter

[Hyperparameters in SVM]

Several hyperparameters have to be selected:

- The slack parameter
- The type of kernel (*linear*, *radial*, *sigmoid*, ..)

[Hyperparameters in SVM]

Several hyperparameters have to be selected:

- The slack parameter
- The type of kernel (*linear*, *radial*, *sigmoid*, ..)
- In an ML framework, one trains the SVM on several samples

[Hyperparameters in SVM]

Several hyperparameters have to be selected:

- The slack parameter
- The type of kernel (*linear*, *radial*, *sigmoid*, ..)
- In an ML framework, one trains the SVM on several samples
    - Number of CV sample should be reduced (100)

[Hyperparameters in SVM]

Several hyperparameters have to be selected:

- The slack parameter
- The type of kernel (*linear*, *radial*, *sigmoid*, ..)
- In an ML framework, one trains the SVM on several samples
  - Number of CV sample should be reduced (100)
  - Linear kernel are less computationally intensive

[Hyperparameters in SVM]

Several hyperparameters have to be selected:

- The slack parameter
- The type of kernel (*linear*, *radial*, *sigmoid*, ..)
- In an ML framework, one trains the SVM on several samples
  - Number of CV sample should be reduced (100)
  - Linear kernel are less computationally intensive
- Proceed with parsimony and increase complexity

[Quiz time]

[Quiz time]

[Takeaways]

SVM is a complex but popular technique

- Based on several clever ideas:

[Takeaways]

SVM is a complex but popular technique

- Based on several clever ideas:
  - Dimension augmentation

[Takeaways]

SVM is a complex but popular technique

- Based on several clever ideas:
  - Dimension augmentation
  - Maximal margin classifier

[TAKEAWAYS]

SVM is a complex but popular technique

- Based on several clever ideas:
  - Dimension augmentation
  - Maximal margin classifier

  $\hookrightarrow$ Soft margins with a *slack* parameter

[TAKEAWAYS]

SVM is a complex but popular technique

- Based on several clever ideas:
  - Dimension augmentation
  - Maximal margin classifier
  - $\hookrightarrow$ Soft margins with a *slack* parameter
  - A trick to avoid costly features transformations

[Takeaways]

SVM is a complex but popular technique

- Based on several clever ideas:
  - Dimension augmentation
  - Maximal margin classifier

  $\hookrightarrow$ Soft margins with a *slack* parameter
  - A trick to avoid costly features transformations

  $\hookrightarrow$ The *Kernel Trick*

[TAKEAWAYS]

SVM is a complex but popular technique

- Based on several clever ideas:
  - Dimension augmentation
  - Maximal margin classifier
  - $\hookrightarrow$ Soft margins with a *slack* parameter
  - A trick to avoid costly features transformations
  - $\hookrightarrow$ The *Kernel Trick*
- Due to the VC theory, SVM use simple models

[TAKEAWAYS]

SVM is a complex but popular technique

- Based on several clever ideas:
  - Dimension augmentation
  - Maximal margin classifier
  - $\hookrightarrow$ Soft margins with a *slack* parameter
  - A trick to avoid costly features transformations
  - $\hookrightarrow$ The *Kernel Trick*
- Due to the VC theory, SVM use simple models
- SVM requires optimization and CPU

[TAKEAWAYS]

SVM is a complex but popular technique

- Based on several clever ideas:
  - Dimension augmentation
  - Maximal margin classifier

  ↪ Soft margins with a *slack* parameter
  - A trick to avoid costly features transformations

  ↪ The *Kernel Trick*
- Due to the VC theory, SVM use simple models
- SVM requires optimization and CPU
- Implemented in many software!