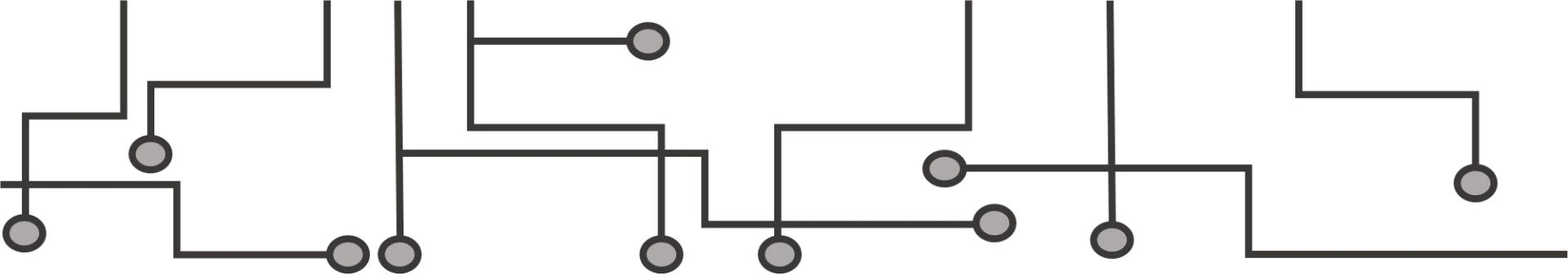


# MEDICAL MICROBIOLOGY AND INFECTIOUS DISEASES CODING WORKSHOP

***Presents***

***Introduction to machine learning in R***


**INSTRUCTED BY**

*Vasena Jayamanna (MSc student)*

# INFORMATION FOR PARTICIPANTS

**All workshops are being recorded and posted to the MMID Coding Workshop - YouTube**

***Question and Answer period will not be recorded***

# LEARNING OBJECTIVES

- What is machine learning?
  - Motivation for use
- Support vector machines - theory
- Support vector machines - in R
  - Linearly separable / non-linearly separable
  - Multi-class variables
  - Kernel trick
  - Overfitting

# Libraries

- *magrittr*
  - a forward-pipe operator, here we will use it to organize our steps in a feed-forward manner
- *tidyverse*
  - a set of packages for organized data analysis
- *e1071*
  - a statistical analysis package, which includes functions for support vector machine learning
- ggplot2
  - for some general plotting

# Machine learning

- Why do we do this?
  - Learning from data; in a big data era
    - More features/fields - high dimensional data sets
    - Larger amounts of data

# Machine learning

- Why do we do this?
  - Learning from data; in a big data era
    - More features/fields - high dimensional data sets
    - Larger amounts of data
  - Essentially:
    - Pattern recognition can be tricky to program
    - → Learning from examples

# Machine learning

- Unsupervised learning: finding patterns in the data without known labels
  - For dimensionality reduction, clustering, …

# Machine learning

- Unsupervised learning: finding patterns in the data without known labels
  - For dimensionality reduction, clustering, …
- Supervised learning (our focus today)
  - We can train a *classifier* on a set of data with known classes → the training set
  - For classification, regression

# Machine learning

- Classification
  - “Assign[ing] an unknown *pattern* to one out of a number of [known] classes” [1]
  - E.g. identifying an image

# Machine learning

- Classification
  - “Assign[ing] an unknown *pattern* to one out of a number of [known] classes” [1]
  - E.g. identifying an image
- Regression
  - A similar idea, where we have values to predict instead of a categorical value
  - We estimate a function *f* using the training data

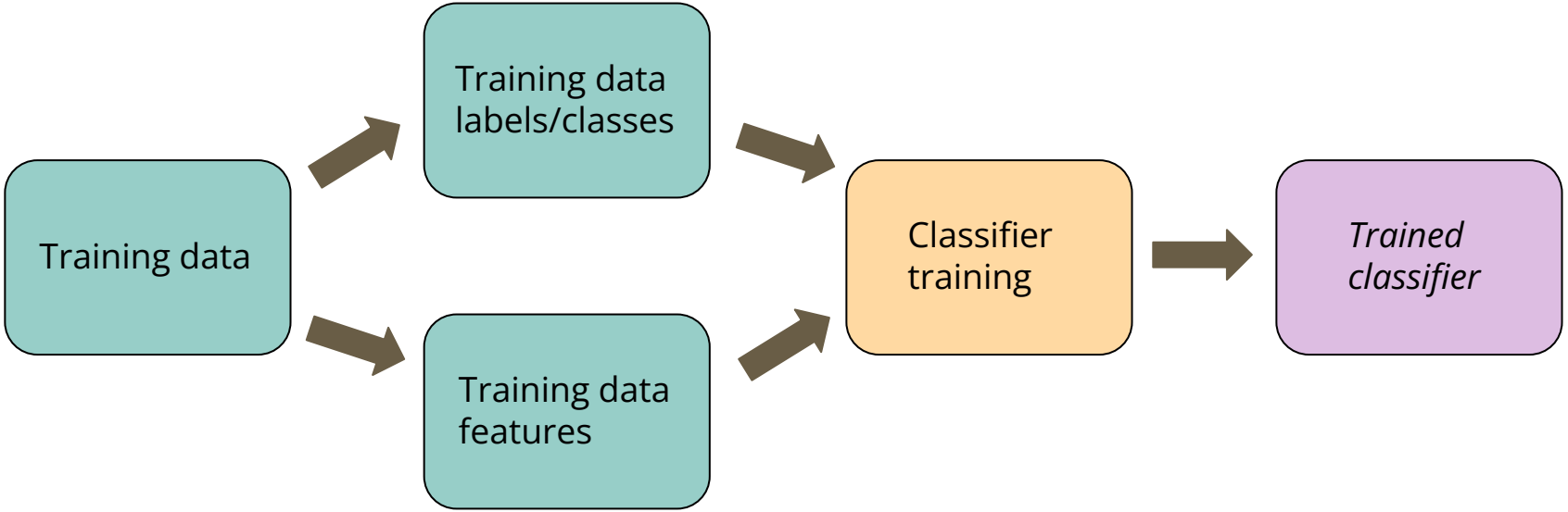
# Machine learning

- Why do we do this?
  - Machine learning builds on statistical techniques
  - A different approach to problems increasing in complexity

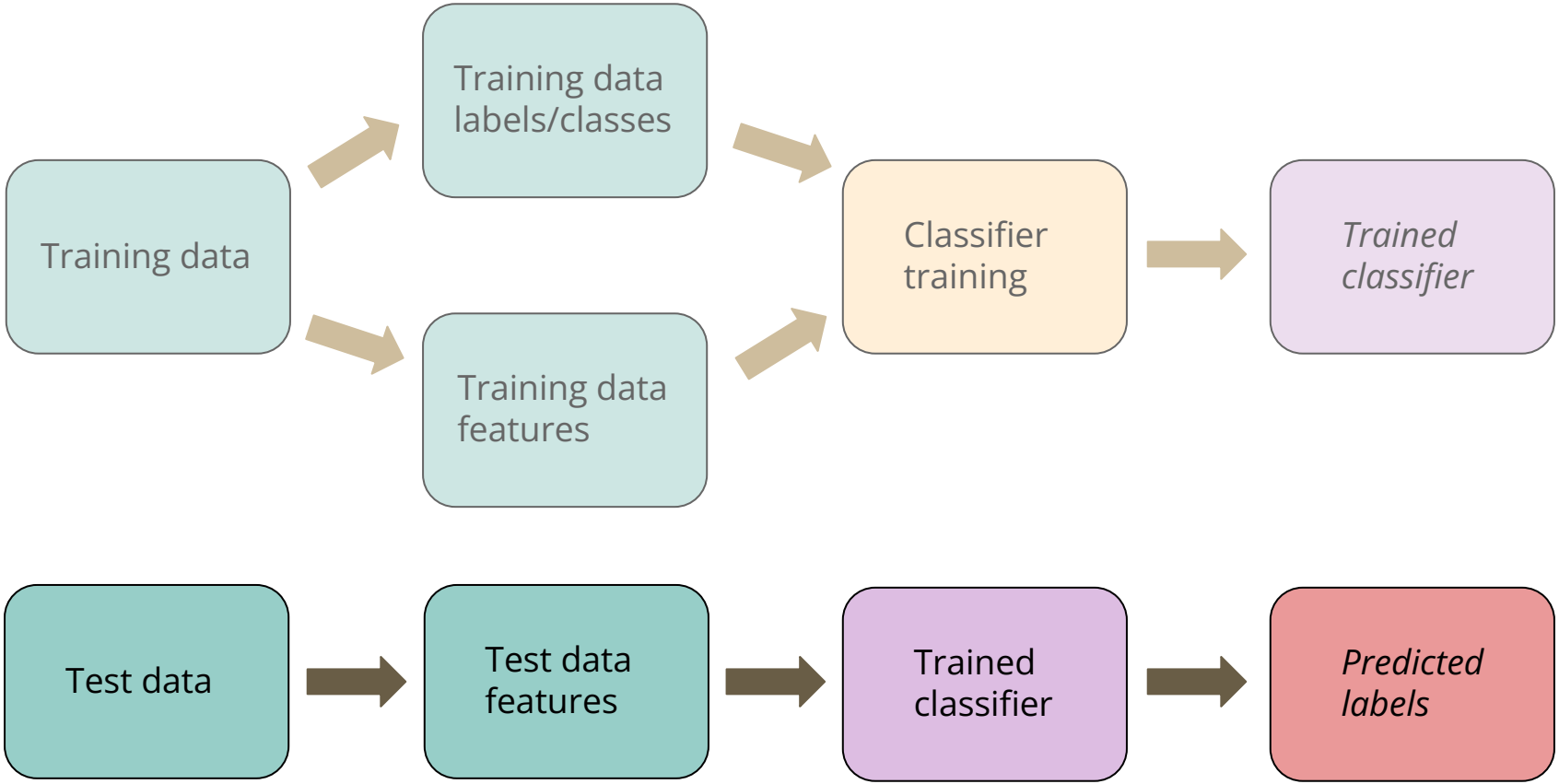
# Machine learning

- Why do we do this?
  - Machine learning builds on statistical techniques
  - A different approach to problems increasing in complexity
  - Uses:
    - Image recognition
    - Language processing
    - Computational biology
    - Robotics
    - etc.

# Supervised learning



# Supervised learning



# Machine learning algorithms

- List of different types:
  - Dimensionality reduction algorithms
  - Random forests
  - Support vector machines
  - Decision trees
  - Neural networks
  - …

# Machine learning algorithms

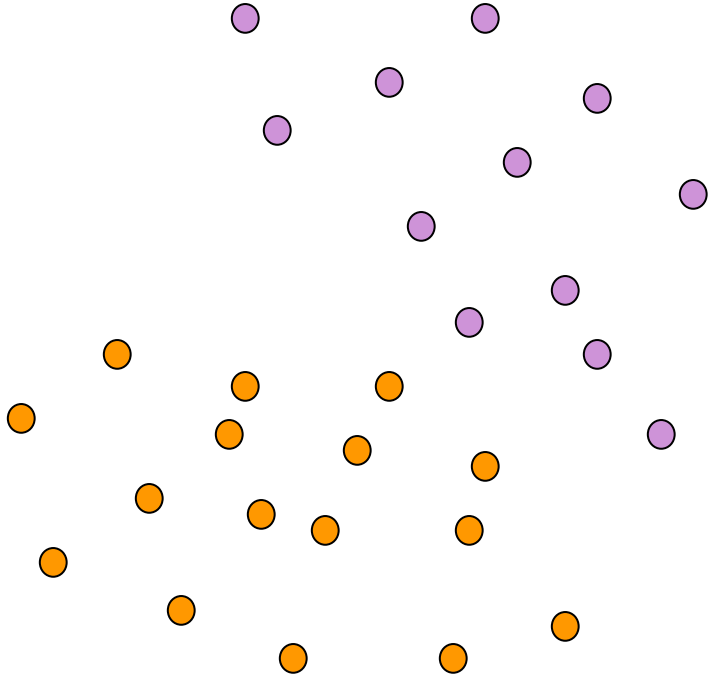
- List of different types:
  - Dimensionality reduction algorithms
  - Random forests
  - **Support vector machines**
  - Decision trees
  - Neural networks
  - …

# Support Vector Machines

- A common classifier
- Many (official *and* unofficial) implementations available in different programming languages
  - Including python and R

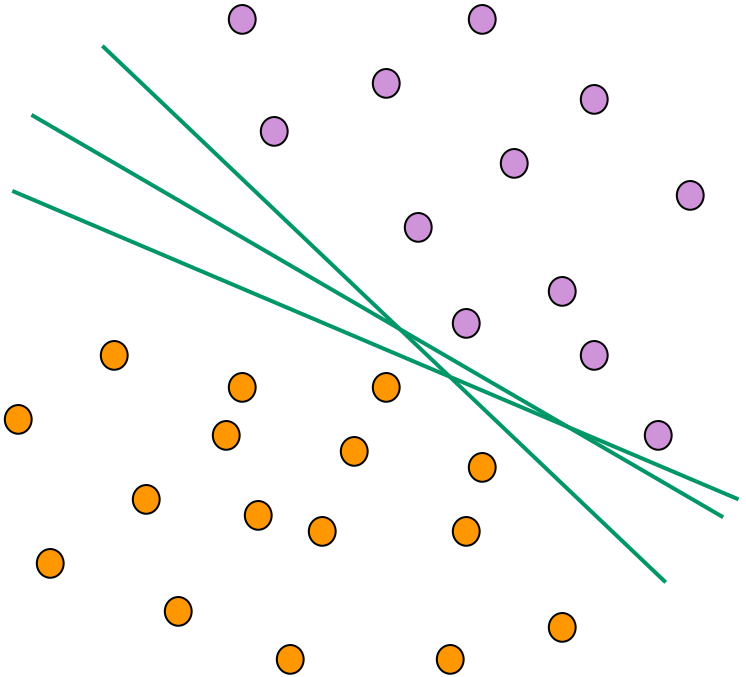
# Linearly separable data and SVMs

- Consider a data set with two classes that are linearly separable in a 2D plane



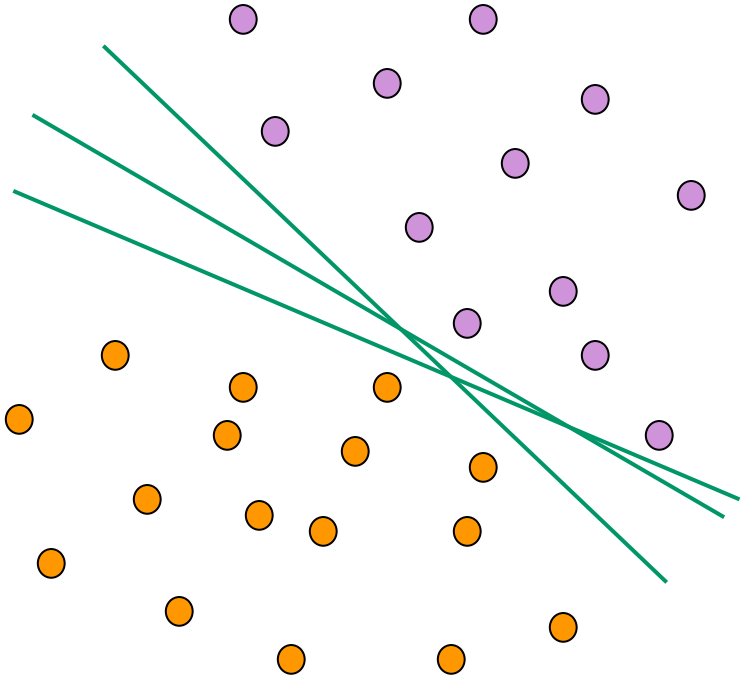
# Linearly separable data and SVMs

- Consider a data set with two classes that are linearly separable in a 2D plane
  - Which line is the best fit?



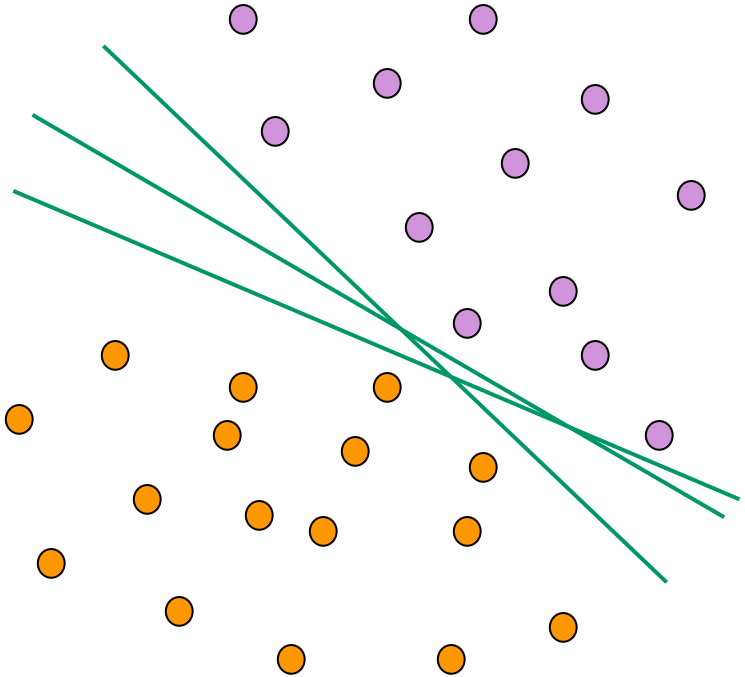
# Linearly separable data and SVMs

- Consider a data set with two classes that are linearly separable in a 2D plane
  - Which line is the best fit?
  - For a small example like this, we could use a linear/logistic regression model



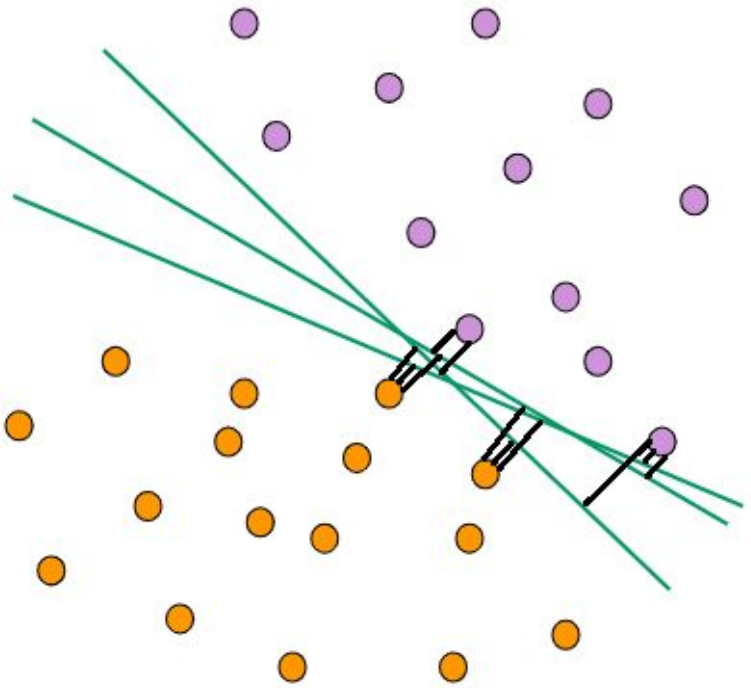
# Linearly separable data and SVMs

- Consider a data set with two classes that are linearly separable in a 2D plane
  - Which line is the best fit?
  - For a small example like this, we could use a linear/logistic regression model
  - But what happens when we get a lot more data, with more classes → high dimensional data?



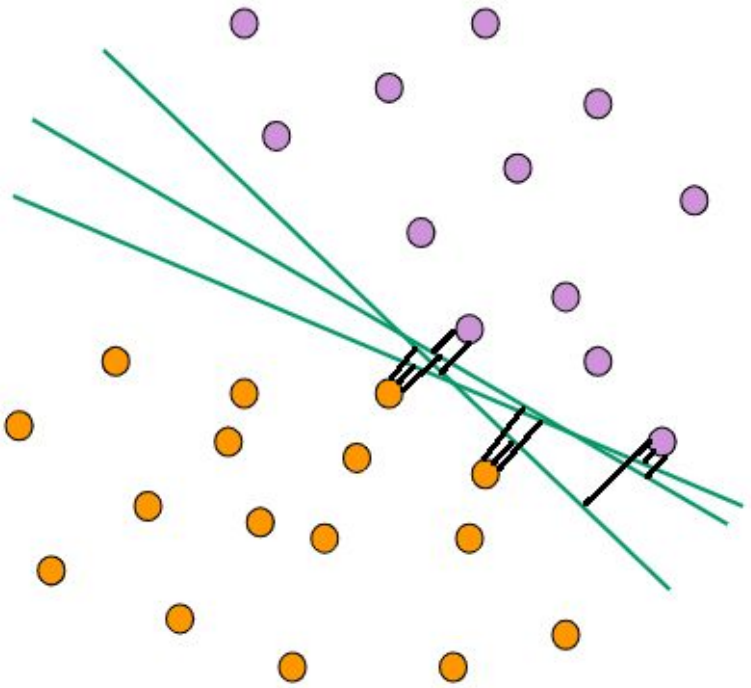
# Linearly separable data and SVMs

- Which line is the best fit?
  - Find the one with the largest margin between the line and the nearest data points (on both sides)

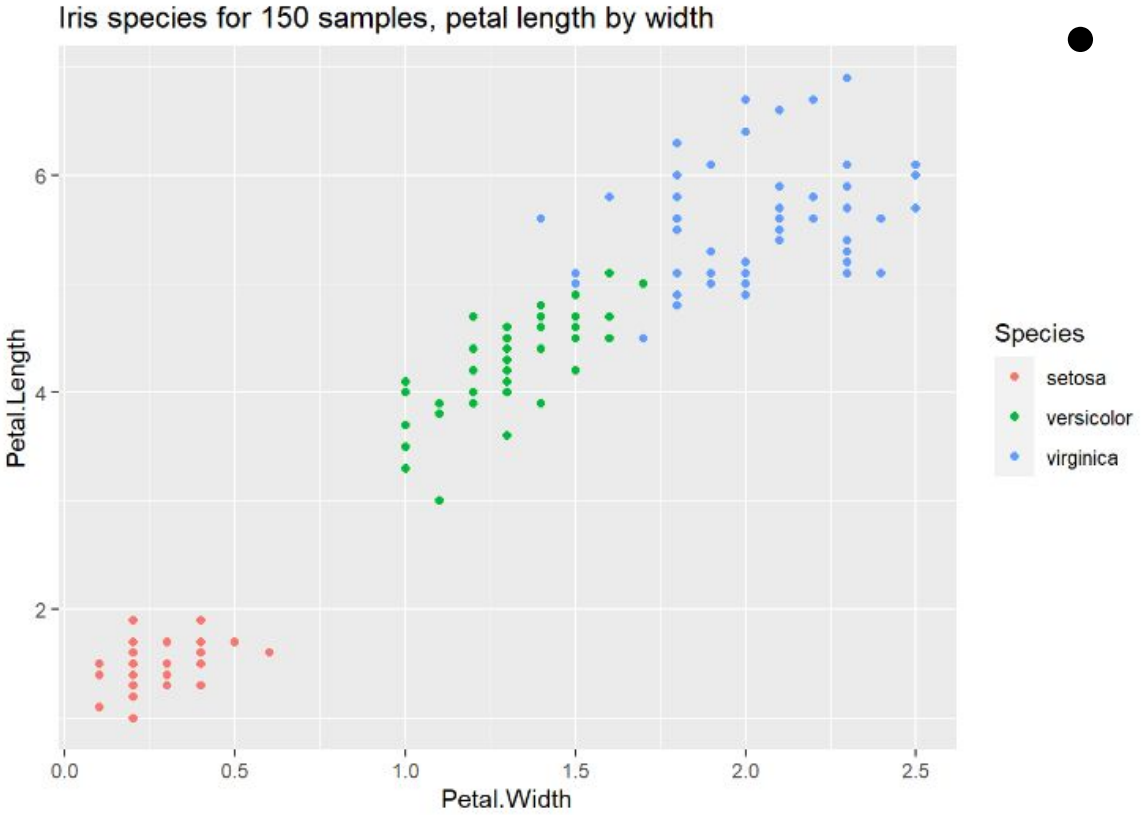


# Linearly separable data and SVMs

- Which line is the best fit?
  - Find the one with the largest margin between the line and the nearest data points (on both sides)
    - *decision boundary*

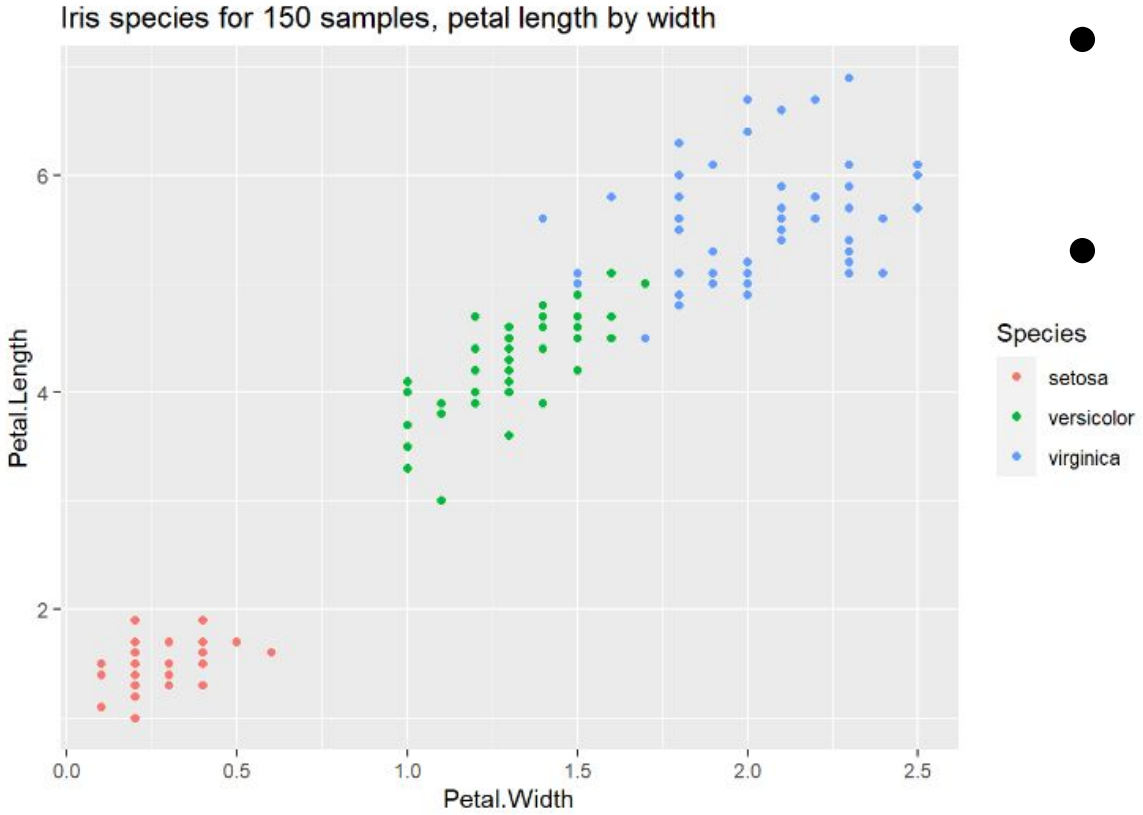


# Linearly separable data and SVMs *in R*




- Consider the *iris* dataset
  - an R built-in set

# Linearly separable data and SVMs *in R*




- Consider the *iris* dataset
  - an R built-in set
- There are four features and a single label class (*Species)*
  - We'll consider the petal widths and petal lengths
  - And start with a small subset of the data

# Linearly separable data and SVMs *in R*

- We'll start with the *setosa, versicolor* species
  - Since they are linearly separable in this set

```r
classes <- c("setosa", "versicolor")
lin_sep <- iris %>%
  filter(Species %in% classes) %>%
  select(Petal.Length, Petal.Width, Species)

lin_sep$Species <- factor(lin_sep$Species, levels = classes)
```

# Linearly separable data and SVMs *in R*

- We'll start with the *setosa, versicolor* species
  - Since they are linearly separable in this set
  - Make sure the *Species* variable is a factor
    - categorical data with preset values
    - → classification SVM (instead of a regression model)

```r
```{r}
classes <- c("setosa", "versicolor")
lin_sep <- iris %>%
  filter(Species %in% classes) %>%
  select(Petal.Length, Petal.Width, Species)

lin_sep$Species <- factor(lin_sep$Species, levels = classes)
```
```

# Linearly separable data and SVMs *in R*

- Here we set aside around 20% of the data to use to test our SVM
- We do this by generating a set of binary indices so we can subset the data into a training set and a test set

```r
sample_indices <- sample(2, nrow(lin_sep), replace = TRUE, prob = c(0.8,0.2))

table(sample_indices)
```

# Linearly separable data and SVMs *in R*

- Here we set aside around 20% of the data to use to test our SVM
- We do this by generating a set of binary indices so we can subset the data into a training set and a test set

```r
sample_indices <- sample(2, nrow(lin_sep), replace = TRUE, prob = c(0.8,0.2))

table(sample_indices)
```

```
sample_indices
 1  2
82 18
```

# Linearly separable data and SVMs *in R*

- Here we set aside around 20% of the data to use to test our SVM
- We do this by generating a set of binary indices so we can subset the data into a training set and a test set

```{r}
sample_indices <- sample(2, nrow(lin_sep), replace = TRUE, prob = c(0.8,0.2))

table(sample_indices)
```

```
sample_
 1  2
82 18
```

```
training_linsep <- lin_sep[sample_indices == 1,]
test_linsep <- lin_sep[sample_indices == 2,]
```

# Linearly separable data and SVMs *in R*

# Linearly separable data and SVMs *in R*

- The function call itself is pretty straightforward
  - We'll start with a linear kernel and a cost value of 10
  - The cost value is for penalizing misclassified inputs or those that violate the margin boundary - larger cost → smaller margin

```r
svm_model <- svm(Species ~ ., data = training_linsep, kernel = "linear", cost = 10, scale = FALSE)
print(svm_model)
```

# Linearly separable data and SVMs *in R*

- The function call itself is pretty straightforward
  - We'll start with a linear kernel and a cost value of 10
  - The cost value is for penalizing misclassified inputs or those that violate the margin boundary - larger cost → smaller margin

```r
svm_model <- svm(Species ~ ., data = training_linsep, kernel = "linear", cost = 10, scale = FALSE)
print(svm_model)
```

```
Call:
svm(formula = Species ~ ., data = training_linsep, kernel = "linear", cost = 10, scale = FALSE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  10

Number of Support Vectors:  2
```

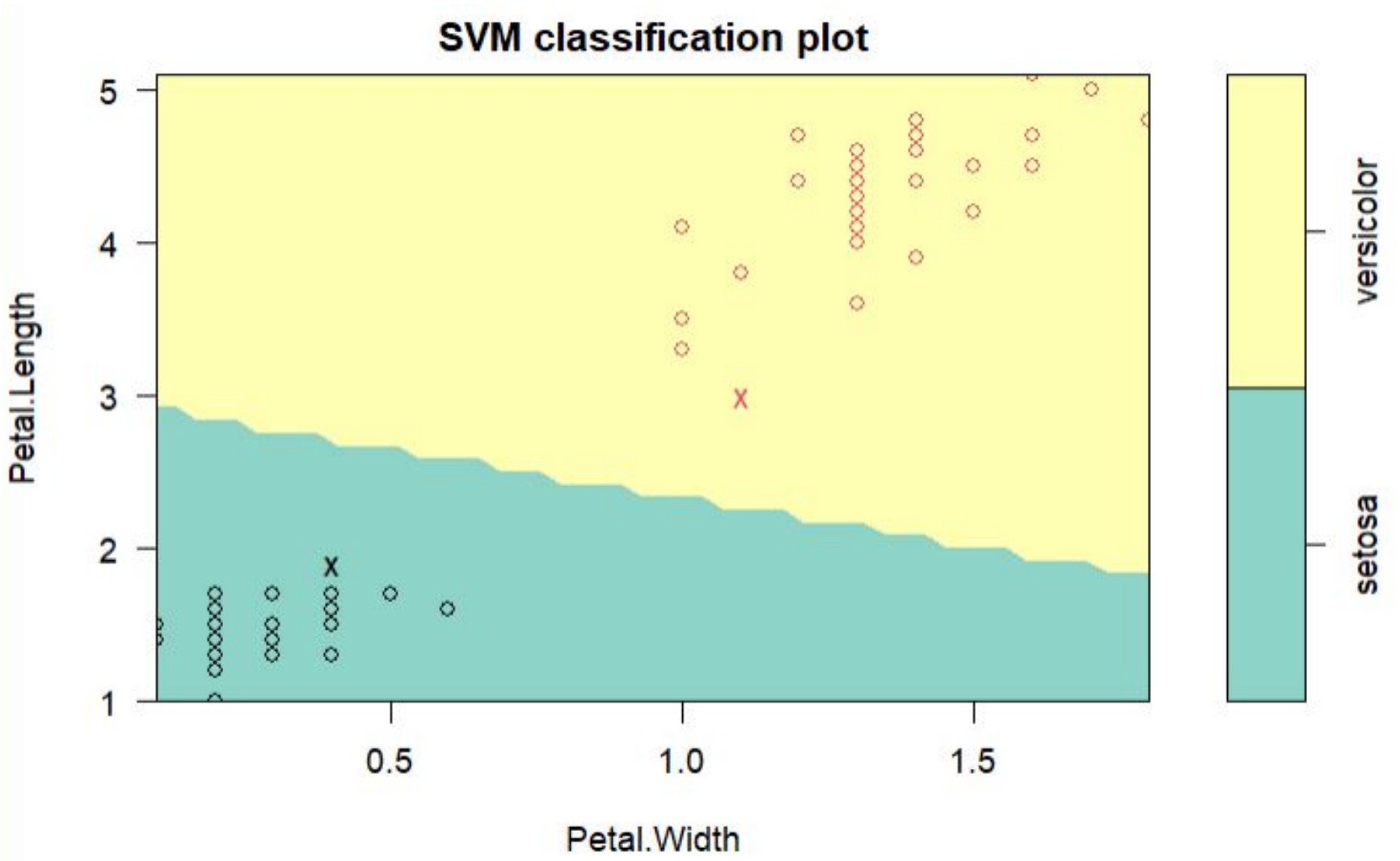
# Linearly separable data and SVMs *in R*

- We can plot the resulting model and see how it splits the data

```r
svm_model <- svm(Species ~ ., data = training_linsep, kernel = "linear", cost = 10, scale = FALSE)
print(svm_model)
```

```
Call:
svm(formula = Species ~ ., data = training_linsep, kernel = "linear", cost = 10, scale = FALSE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  10

Number of Support Vectors:  2
```

```r
plot(svm_model, training_linsep, dataSymbol = 1, col = c("#8DD3C7","#FFFFB3"))
```


SVM classification plot
Petal.Length
Petal.Width
5
4
3
2
1
0.5
1.0
1.5
versicolor
setosa

# Linearly separable data and SVMs *in R*

- The tune() function in the e1071 package is for hyper-parameter tuning

```{r}
tune_svm = tune(method = svm, Species ~ .,
                data = training_linsep,
                kernel = "linear",
                ranges = list(cost = 10^(-1:2)))
summary(tune_svm$best.model)
```

```
Call:
best.tune(method = svm, train.x = Speci
    kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.1

Number of Support Vectors:  12

 ( 6 6 )


Number of Classes:  2

Levels:
 setosa versicolor
```

# Linearly separable data and SVMs *in R*

- The tune() function in the e1071 package is for hyper-parameter tuning
- You provide a kernel: *linear*, *polynomial*, *sigmoid*, or *radial*

```
Call:
best.tune(method = svm, train.x = Speci
    kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.1

Number of Support Vectors:  12

 ( 6 6 )


Number of Classes:  2

Levels:
 setosa versicolor
```

```
```{r}
tune_svm = tune(method = svm, Species ~ .,
                data = training_linsep,
                kernel = "linear",
                ranges = list(cost = 10^(-1:2)))
summary(tune_svm$best.model)
```
```

# Linearly separable data and SVMs *in R*

- The tune() function in the e1071 package is for hyper-parameter tuning
- You provide a kernel: *linear*, *polynomial*, *sigmoid*, or *radial*
- With a list of values for each parameter you want to tune

```r
tune_svm = tune(method = svm, Species ~ .,
                data = training_linsep,
                kernel = "linear",
                ranges = list(cost = 10^(-1:2)))
summary(tune_svm$best.model)
```

```
Call:
best.tune(method = svm, train.x = Speci
    kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.1

Number of Support Vectors:  12

 ( 6 6 )


Number of Classes:  2

Levels:
 setosa versicolor
```

# Linearly separable data and SVMs *in R*

- In this example, the “best” model has a cost value of 0.1 (where the cost is the penalty for violating the constraints that define the decision boundary)

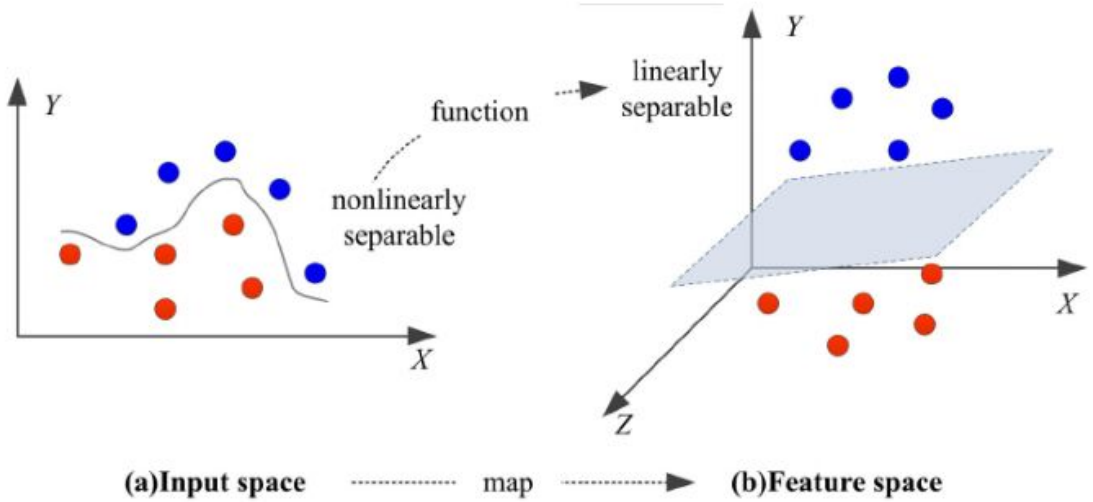
```r
tune_svm = tune(method = svm, Species ~ .,
                data = training_linsep,
                kernel = "linear",
                ranges = list(cost = 10^(-1:2)))
summary(tune_svm$best.model)
```

```
Call:
best.tune(method = svm, train.x = Speci
    kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.1

Number of Support Vectors:  12

 ( 6 6 )


Number of Classes:  2

Levels:
 setosa versicolor
```


What about non-linearly separable data?

# Non-linearly separable data and SVMs *in R*

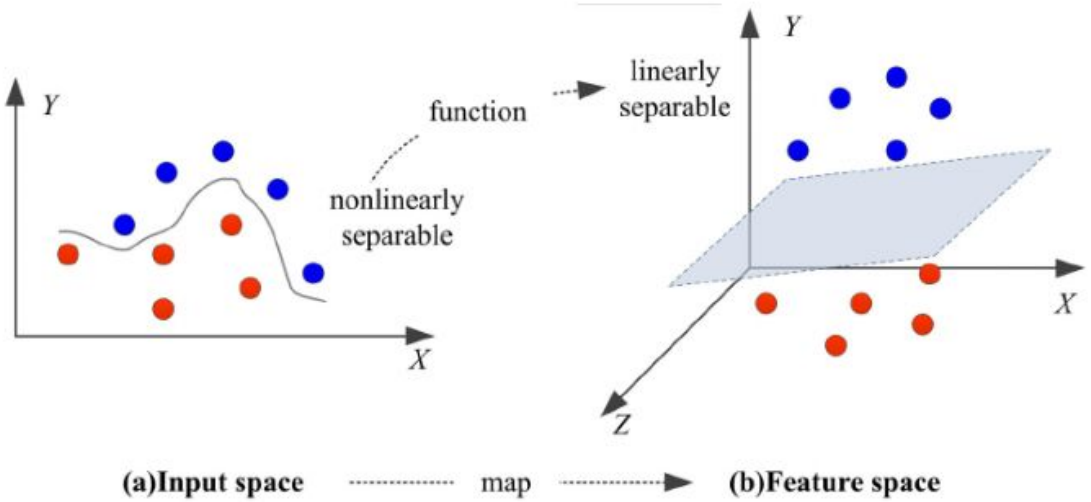
- What about non-linearly separable data?




[2]https://www.researchgate.net/publication/282920922_Heuristic_Methods_for_Reservoir_Monthly_Inflow_Forecasting_A_Case_Study_of_Xinfengjiang_Reservoir_in_Pearl_River_China

# Non-linearly separable data and SVMs *in R*

- What about non-linearly separable data?
  - You can add another feature to move to a higher dimension
    - To map some combination of other features




[2]https://www.researchgate.net/publication/282920922_Heuristic_Methods_for_Reservoir_Monthly_Inflow_Forecasting_A_Case_Study_of_Xinfengjiang_Reservoir_in_Pearl_River_China

# Non-linearly separable data and SVMs *in R*

- We'll consider the *virginica* and *versicolor* labels
  - Using a "radial" kernel

```r
svm_model <- svm(Species ~ ., data = training_nonlin, kernel = "radial", cost = 5, gamma = 1, scale = FALSE)
plot(svm_model, training_nonlin, dataSymbol = 1, col = c("#8DD3C7","#FFFFB3"))
```

# Non-linearly separable data and SVMs *in R*

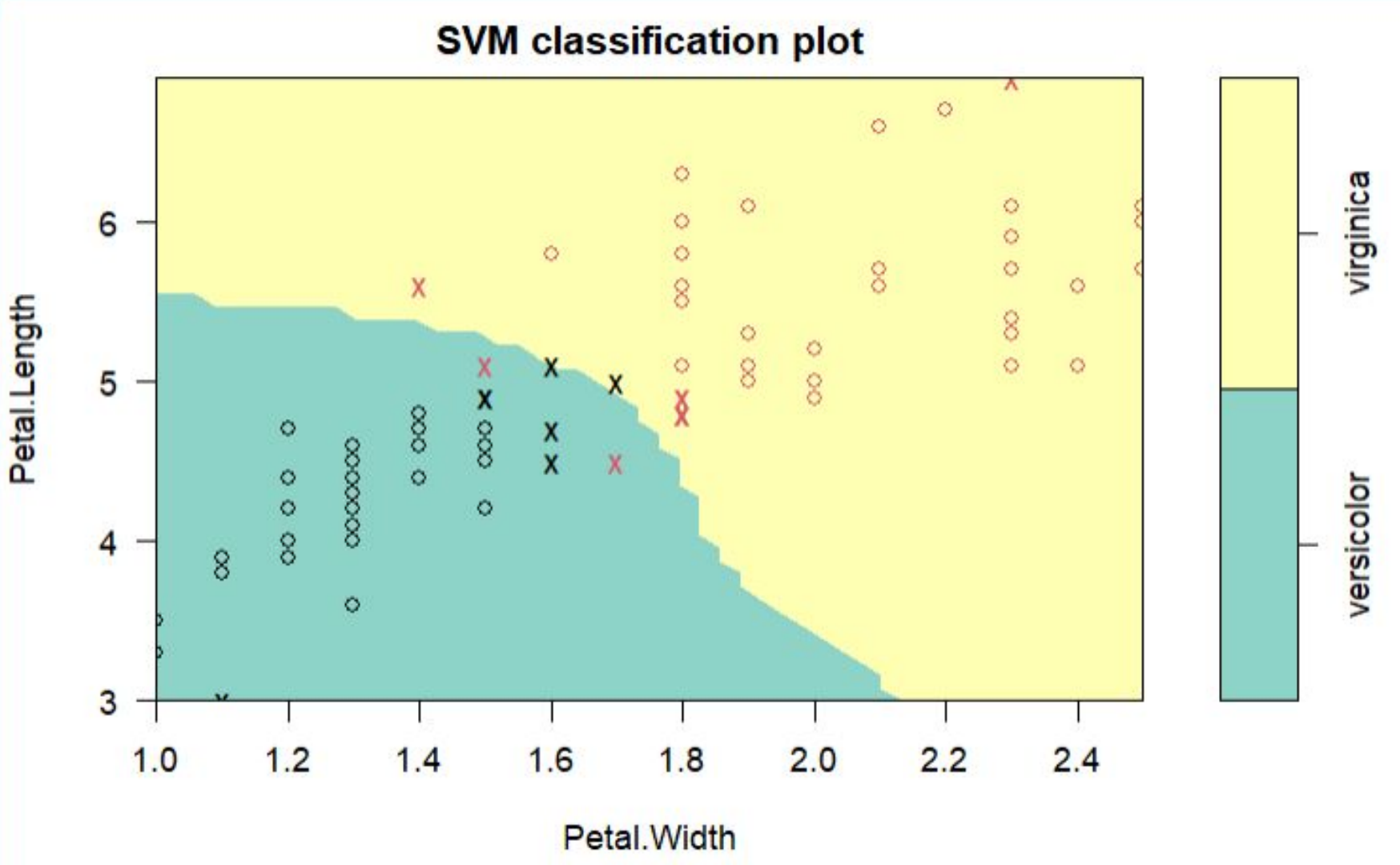
- We'll consider the *virginica* and *versicolor* labels
  - Using a "radial" kernel
    - Radial basis function kernel
      - Intuitively, think of it as a high dimensional feature mapping to a space where the decision boundary can be well-defined

```r
svm_model <- svm(Species ~ ., data = training_nonlin, kernel = "radial", cost = 5, gamma = 1, scale = FALSE)
plot(svm_model, training_nonlin, dataSymbol = 1, col = c("#8DD3C7","#FFFFB3"))
```

# Non-linearly separable data and SVMs *in R*

- We'll consider the *virginica* and *versicolor* labels
  - Using a "radial" kernel
    - Radial basis function kernel
      - Intuitively, think of it as a high dimensional feature mapping to a space where the decision boundary can be well-defined
  - With cost = 5 and gamma ($\gamma$) = 1, to start with

```r
svm_model <- svm(Species ~ ., data = training_nonlin, kernel = "radial", cost = 5, gamma = 1, scale = FALSE)
plot(svm_model, training_nonlin, dataSymbol = 1, col = c("#8DD3C7","#FFFFB3"))
```


SVM classification plot
Petal.Length
Petal.Width
3
4
5
6
1.0
1.2
1.4
1.6
1.8
2.0
2.2
2.4
virginica
versicolor

# Non-linearly separable data and SVMs *in R*

- The X's in this plot indicate which points were used to define the support vectors

```
tune_svm = tune(method = svm, Species ~ .,
                data = training_nonlin, kernel = "radial",
                ranges = list(cost = c(0.001, 0.01, 0.1, 1,5,10,100),
                              gamma = seq(0.5, 5, 0.5)))
summary(tune_svm$best.model)
```

# Non-linearly separable data and SVMs *in R*

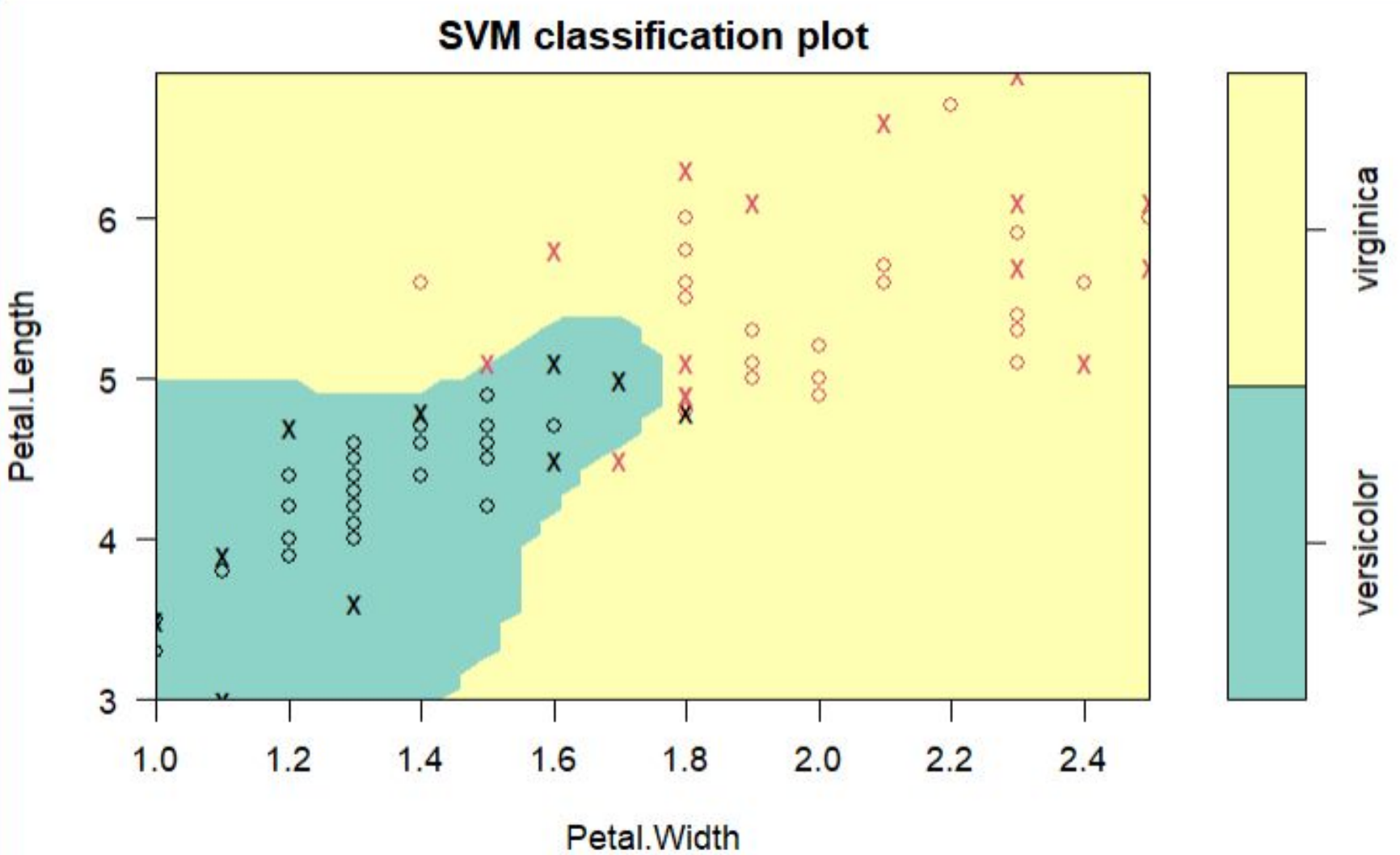
- The X's in this plot indicate which points were used to define the support vectors
- Again, we try hyper-parameter tuning
  - RBF kernel → need to supply cost and $\gamma$ values

```
tune_svm = tune(method = svm, Species ~ .,
                data = training_nonlin, kernel = "radial",
                ranges = list(cost = c(0.001, 0.01, 0.1, 1,5,10,100),
                              gamma = seq(0.5, 5, 0.5)))
summary(tune_svm$best.model)
```

# Non-linearly separable data and SVMs *in R*

- The X's in this plot indicate which points were used to define the support vectors
- Again, we try hyper-parameter tuning
  - RBF kernel → need to supply cost and $\gamma$ values
    - $\gamma$ indicates how much influence we want features to have on the decision boundary - high $\gamma$→ fits training data more (be careful to avoid overfitting)

```
tune_svm = tune(method = svm, Species ~ .,
                data = training_nonlin, kernel = "radial",
                ranges = list(cost = c(0.001, 0.01, 0.1, 1,5,10,100),
                              gamma = seq(0.5, 5, 0.5)))
summary(tune_svm$best.model)
```


SVM classification plot
Petal.Length
Petal.Width
3
4
5
6
1.0
1.2
1.4
1.6
1.8
2.0
2.2
2.4
virginica
versicolor

# Overfitting

- Kernels and high dimensional data bring us to really large feature spaces

# Overfitting

- Kernels and high dimensional data bring us to really large feature spaces
- What about overfitting to the training data?

# Overfitting

- Kernels and high dimensional data bring us to really large feature spaces
- What about overfitting to the training data?
  - You can handle this by adjusting:
    - Cost - if you reduce this, you can allow a little error (soft-margin SVM)

# Overfitting

- Kernels and high dimensional data bring us to really large feature spaces
- What about overfitting to the training data?
  - You can handle this by adjusting:
    - Cost - if you reduce this, you can allow a little error (soft-margin SVM)
    - Kernel selection (e.g. picking a polynomial kernel instead of a linear one, using data-specific knowledge)

# Overfitting

- Kernels and high dimensional data bring us to really large feature spaces
- What about overfitting to the training data?
  - You can handle this by adjusting:
    - Cost - if you reduce this, you can allow a little error (soft-margin SVM)
    - Kernel selection (e.g. picking a polynomial kernel instead of a linear one, using data-specific knowledge)
    - Hyper-parameter tuning - kernel dependent
      - Like adjusting the $\gamma$ value, which tells you how strictly to the data points you want your model to fit

# Non-linearly separable data and SVMs *in R*

- In this case, we're fine
  - we can see that the model was mostly correct with predicting the test data classes

```r
table(predicted = predict(tune_svm$best.model, newdata = test_nonlin),
      actual = test_nonlin$Species)
```

```
            actual
predicted    versicolor virginica
  versicolor          6         1
  virginica           0        11
```

# What about multiple classes?

- “One against one”
  - Learn a classifier for each pair of classes
  - How e1071’s svm() works with multiple classes

# What about multiple classes?

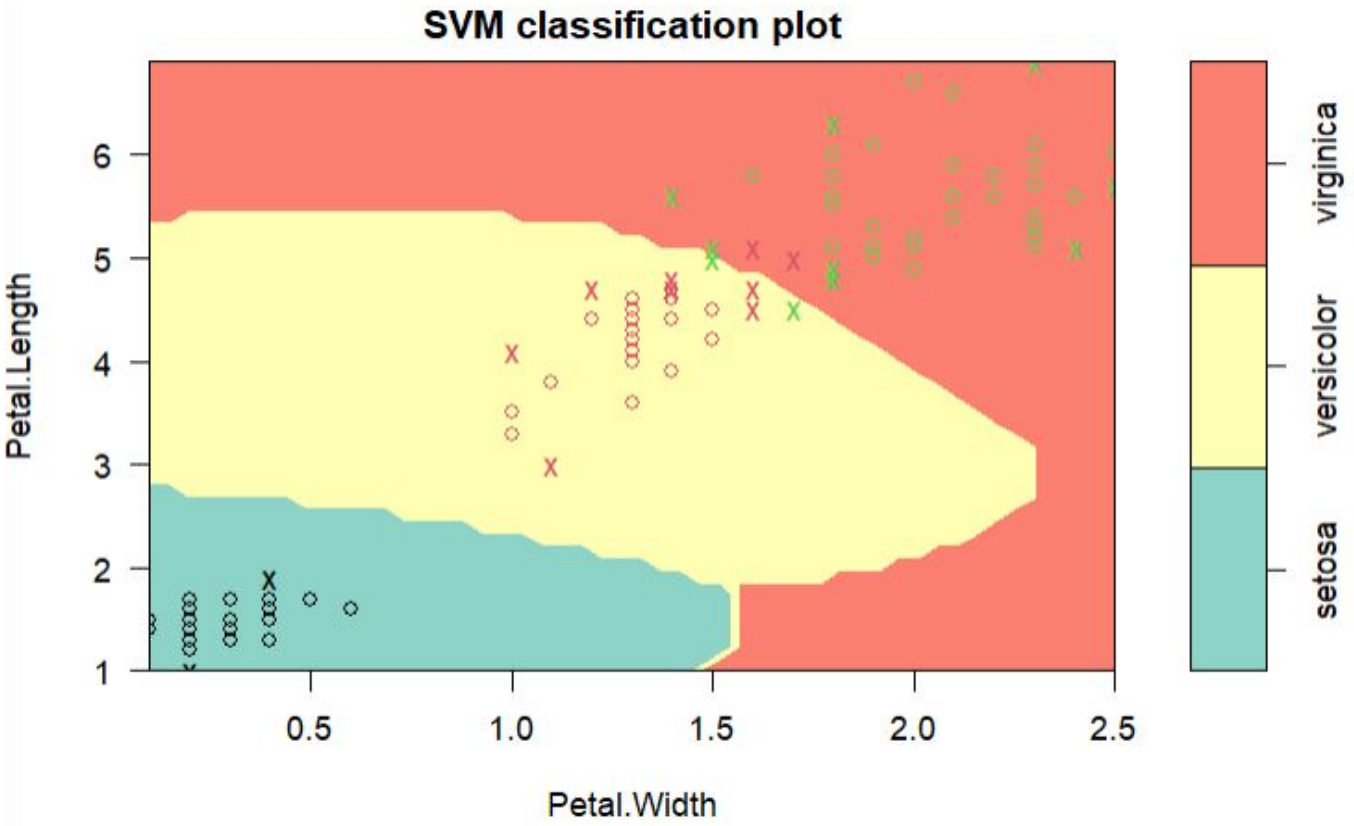
- “One against one”
  - Learn a classifier for each pair of classes
  - How e1071’s svm() works with multiple classes

- “One against all”
  - Learn each of the m classifiers separately (class k versus the rest)
  - Commonly used method

# Multi-class data


- We're back to the full dataset
  - Still focusing on petal widths and lengths
- In practice, same as with two features

```r
svm_model <- svm(Species ~ ., data = training_multi, kernel = "radial", cost = 5, gamma = 1, scale = FALSE)
plot(svm_model, training_multi, dataSymbol = 1, col = c("#8DD3C7","#FFFFB3", "#FB8072"))
```



# REFERENCES

*[1]* Theodoridis, S. (2020). *Machine Learning: A Bayesian and Optimization Perspective* (2nd ed.). Academic Press.

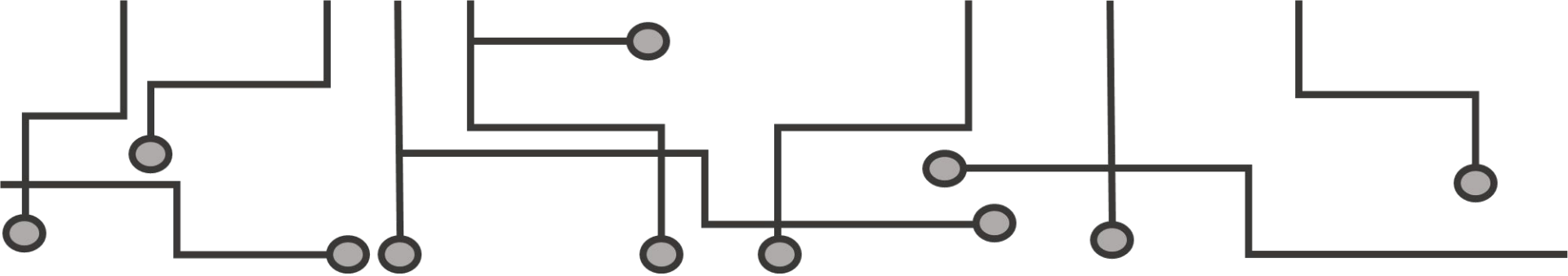
*[2] Cheng, Chun-Tian & Feng, zhong-kai & Niu, Wen-Jing & Liao, Shengli. (2015). Heuristic Methods for Reservoir Monthly Inflow Forecasting: A Case Study of Xinfengjiang Reservoir in Pearl River, China. Water (Switzerland). 7. 4477-4495. 10.3390/w7084477.*

# HELPFUL RESOURCES


*[1]https://ocw.mit.edu/courses/sloan-school-of-management/15-097-prediction-machine-learning-and-statistics-spring-2012/lecture-notes/MIT15_097S12_lec02.pdf*

*[2]https://www.mit.edu/~amidi/teaching/modeling/study-guide/machine-learning-with-r/#supervised-models*

*[3]https://bradleyboehmke.github.io/HOML/svm.html*



# THANK YOU FOR ATTENDING!

## *The Q&A Session will now begin.*

***Please make sure to fill out the Exit Survey***
***We value your feedback!***

***More questions? Please email us at***
***mmid.coding.workshop@gmail.com or post them to the workshop slack channel***