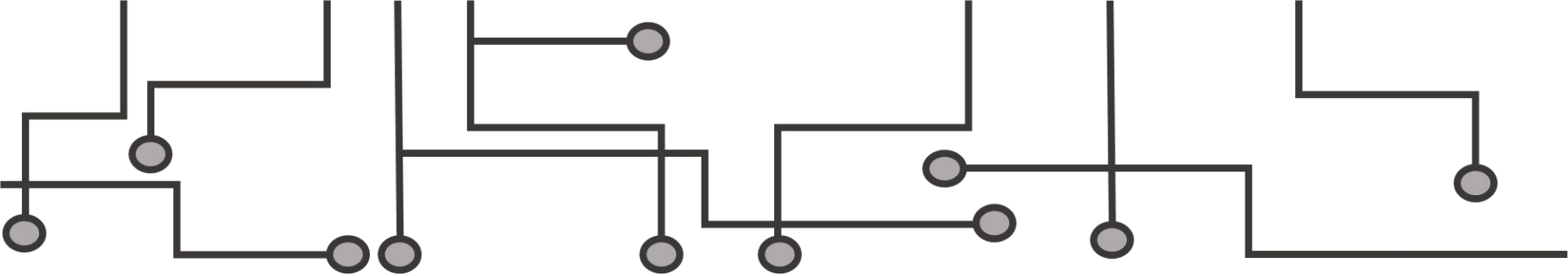


MEDICAL MICROBIOLOGY AND INFECTIOUS DISEASES CODING WORKSHOP

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 - idea
- Support vector machines - theory & practice (in R)
  - linearly separable / non-linearly separable data
  - 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: motivation

- 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: motivation

- 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

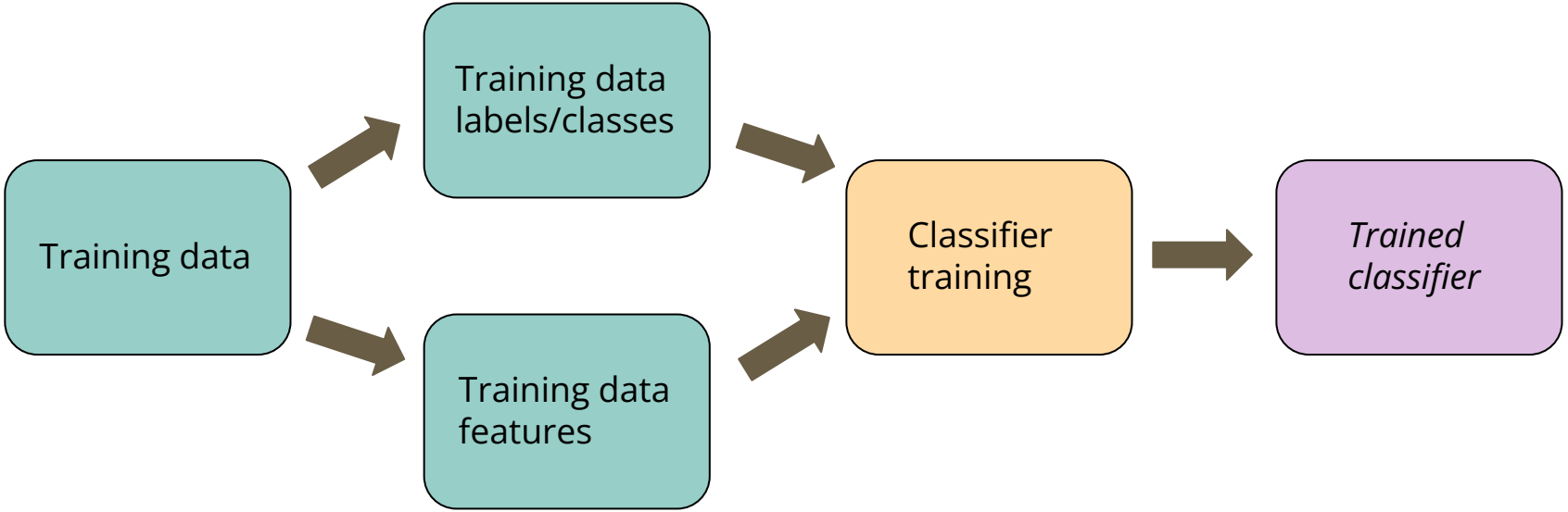
Machine learning: motivation

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

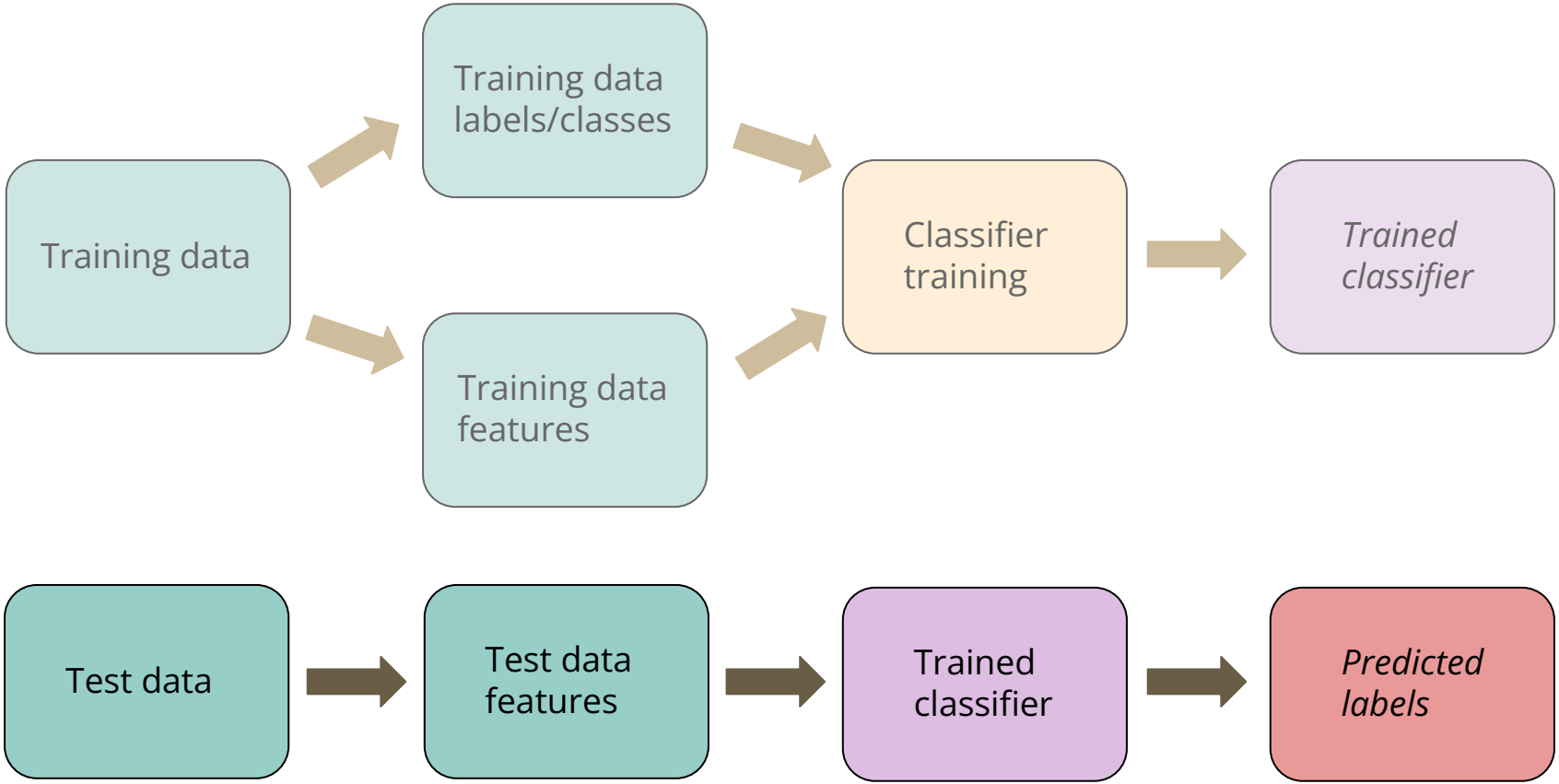
Machine learning: motivation

- 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

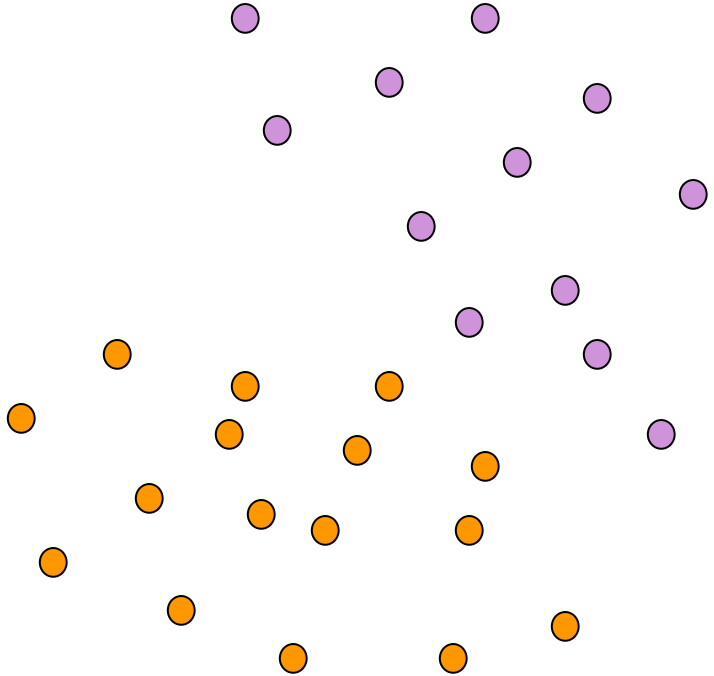
- List of different types:
  - dimensionality reduction algorithms
  - random forests
  - **support vector machines**
  - decision trees
  - neural networks
  - …

Support Vector Machines

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

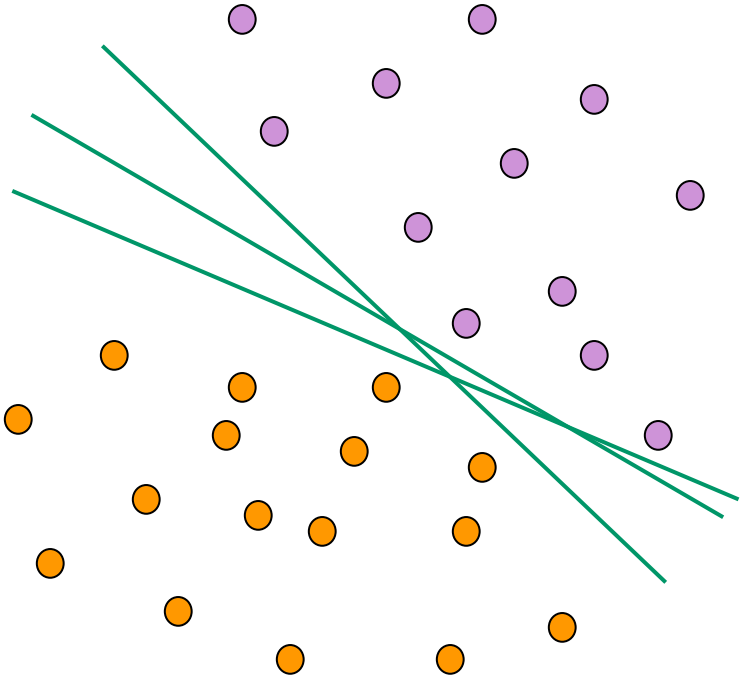
Linearly separable data and SVMs

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



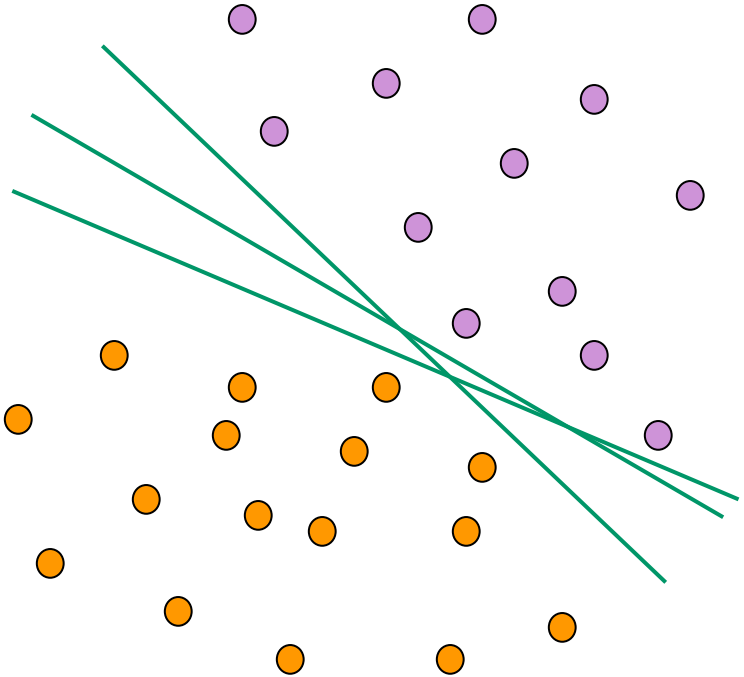
Linearly separable data and SVMs

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



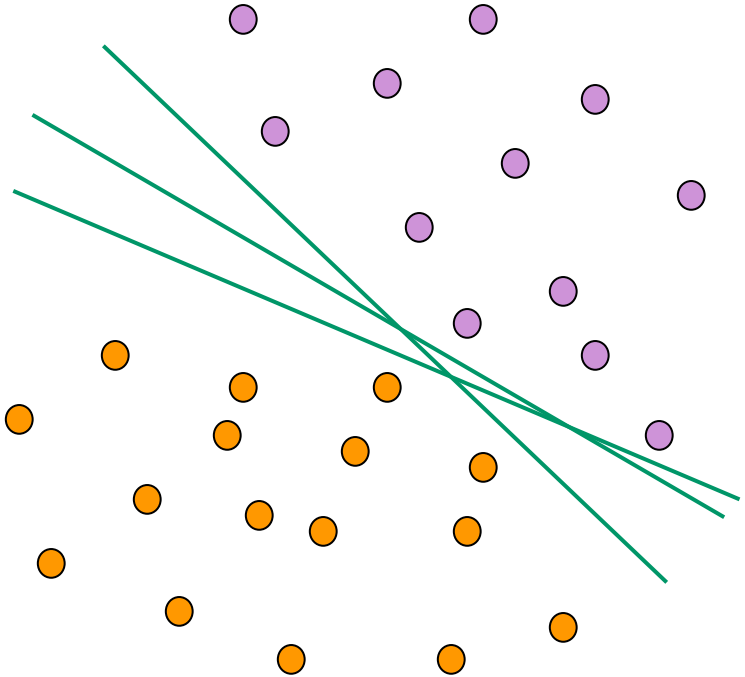
Linearly separable data and SVMs

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



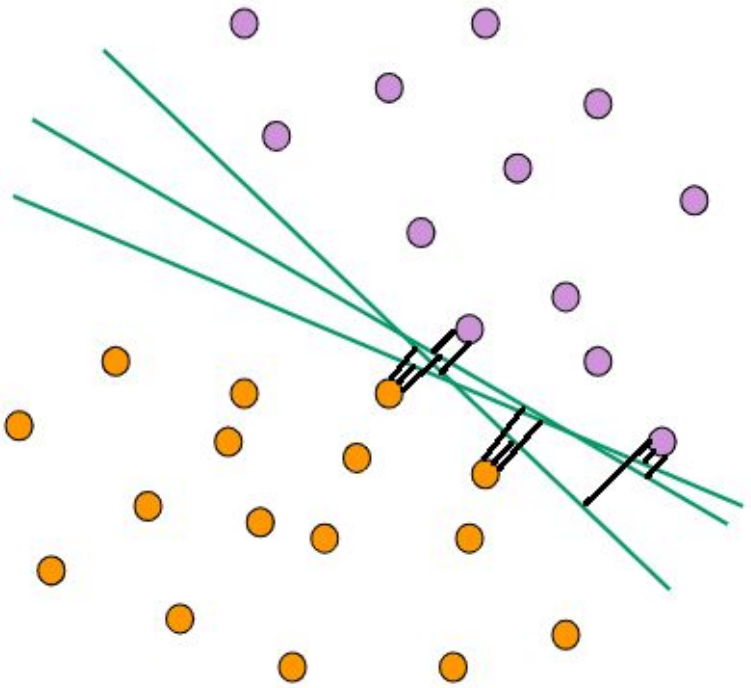
Linearly separable data and SVMs

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



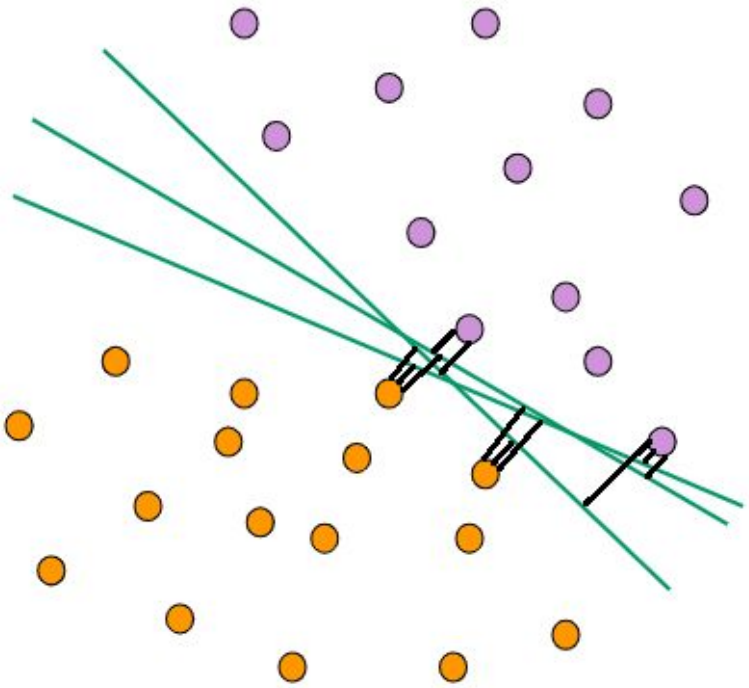
SVMs - boundaries

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

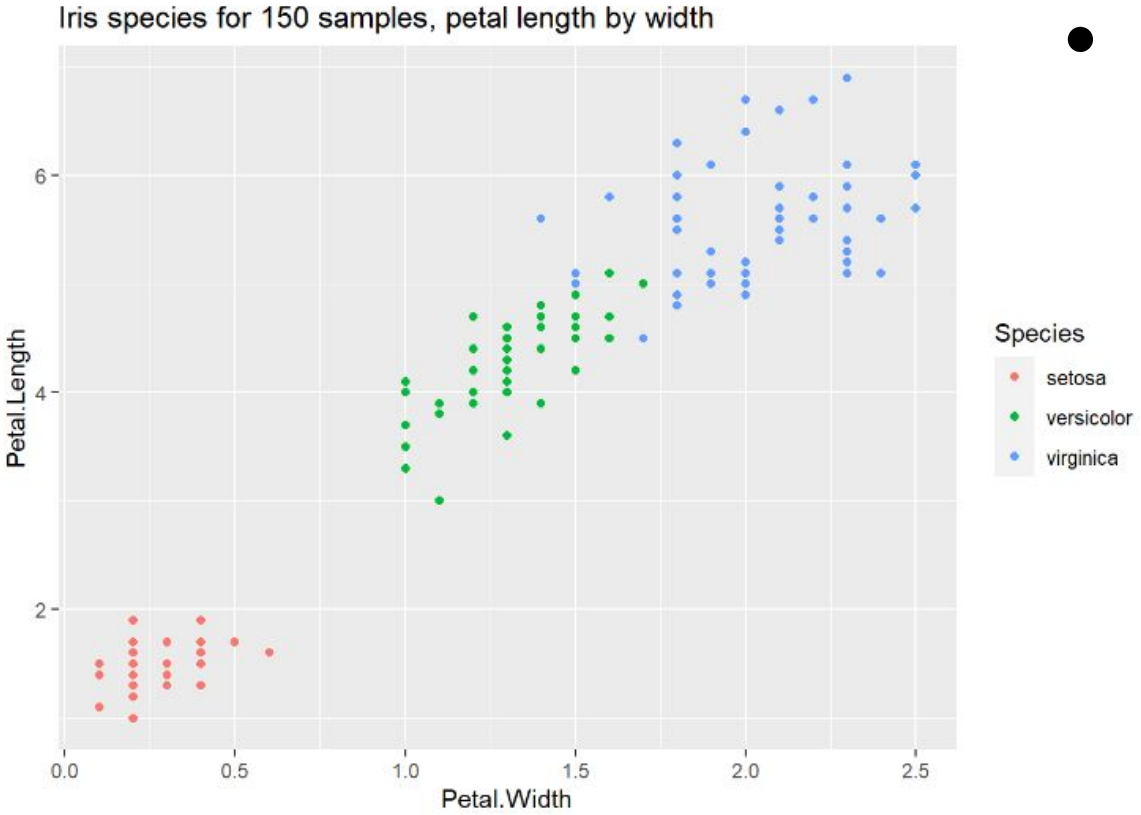


SVMs - boundaries

- 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*
    - *those nearest points* → *support vectors* → *define margin boundaries*

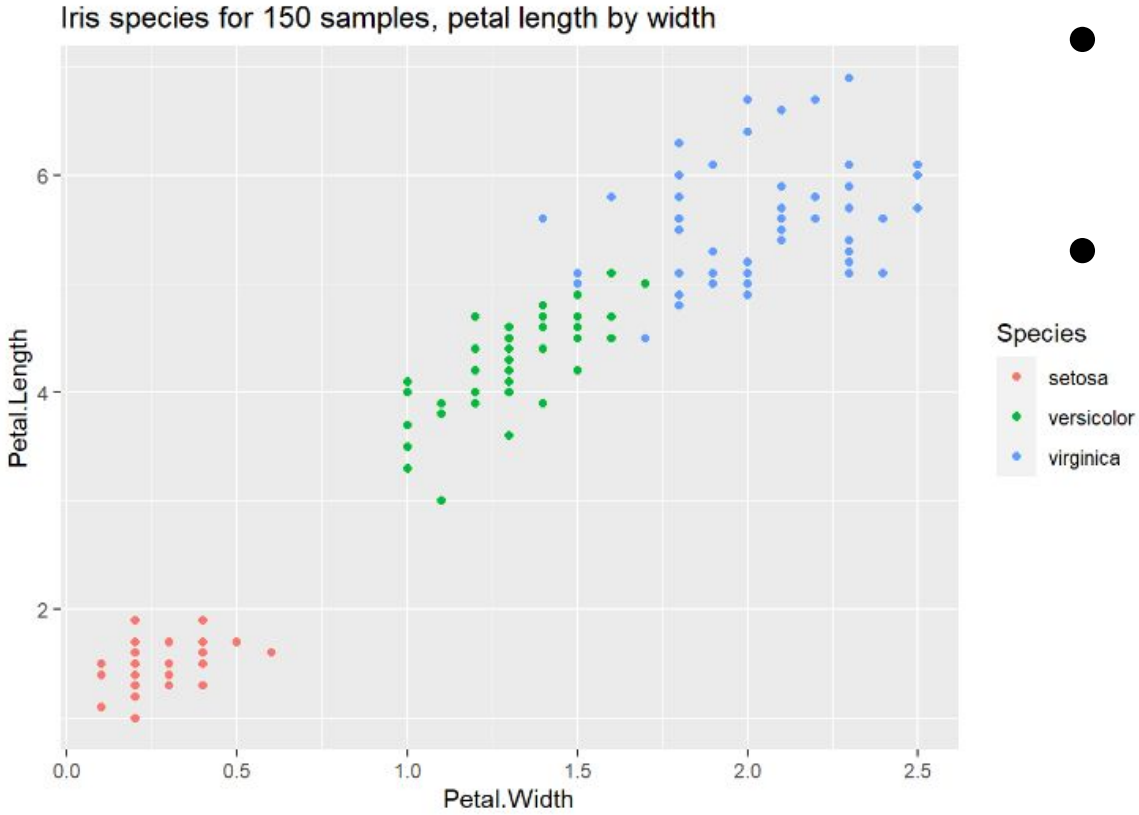


Linearly separable data and SVMs *in R*



- 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)
```
```

Training data

- 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)
```

Training data

- 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
```

Training data

- 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
```

```r
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
  - cost value: penalty for misclassified inputs or those that violate the margin boundary - larger cost → fewer allowed points in 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
  - cost value: penalty for misclassified inputs or those that violate the margin boundary - larger cost → fewer allowed points in 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
```

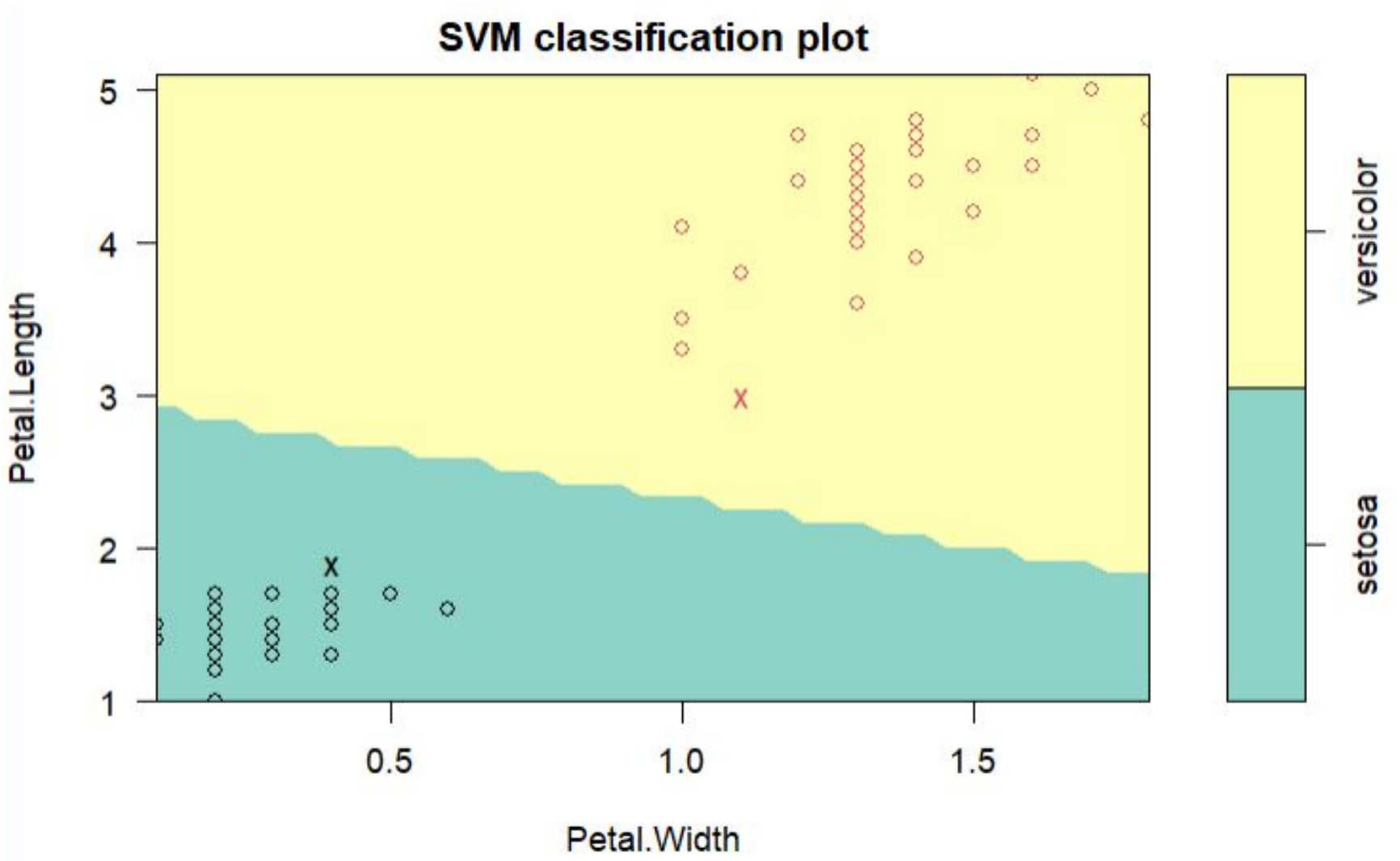
Linearly separable data and SVMs *in R*

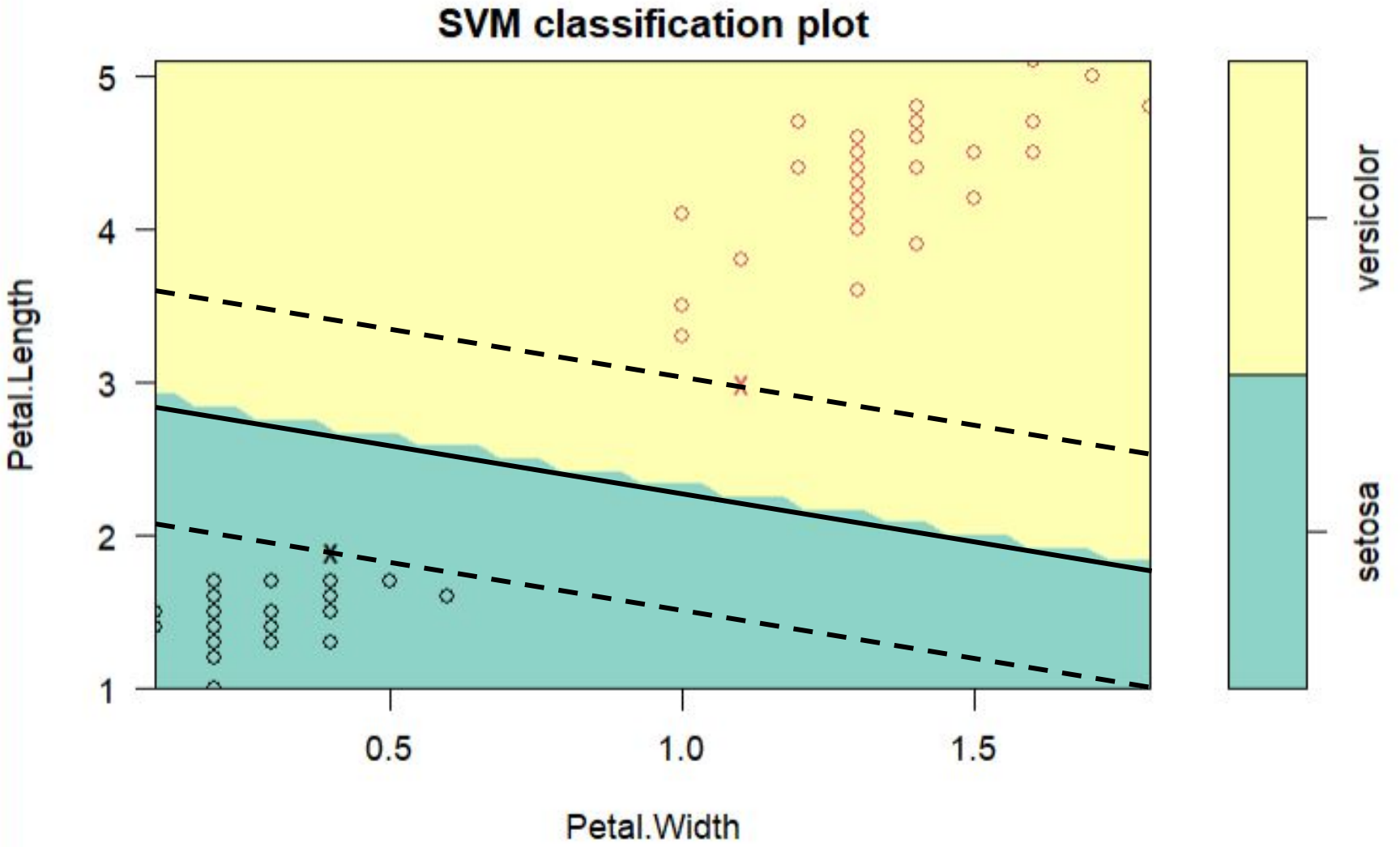
- We can plot the resulting model and see how it splits the data

```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


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

Hyperparameter tuning

- The tune() function in the e1071 package is for hyperparameter 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
```

Hyperparameter tuning

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

```
```{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
```

Hyperparameter tuning

- The tune() function in the e1071 package is for hyperparameter 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
```

Hyperparameter tuning

- 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 margin boundary)

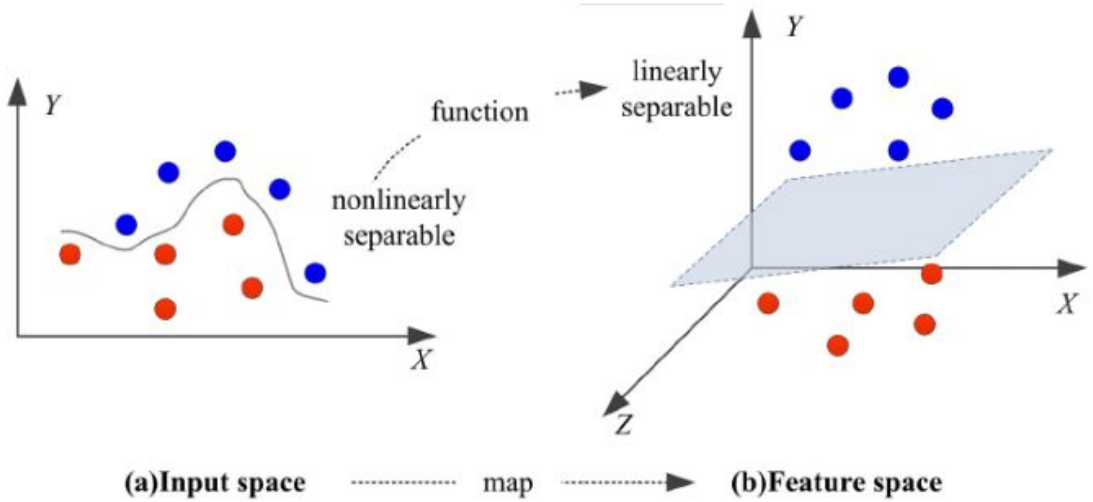
```r
tune_svm = tune(method = svm, Species ~ .,
                data = training_linsep,
                kernel = "linear",
                ranges = list(cost = 10^(-1:2)))
summary(tune_svm$best.model)
```

```
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*

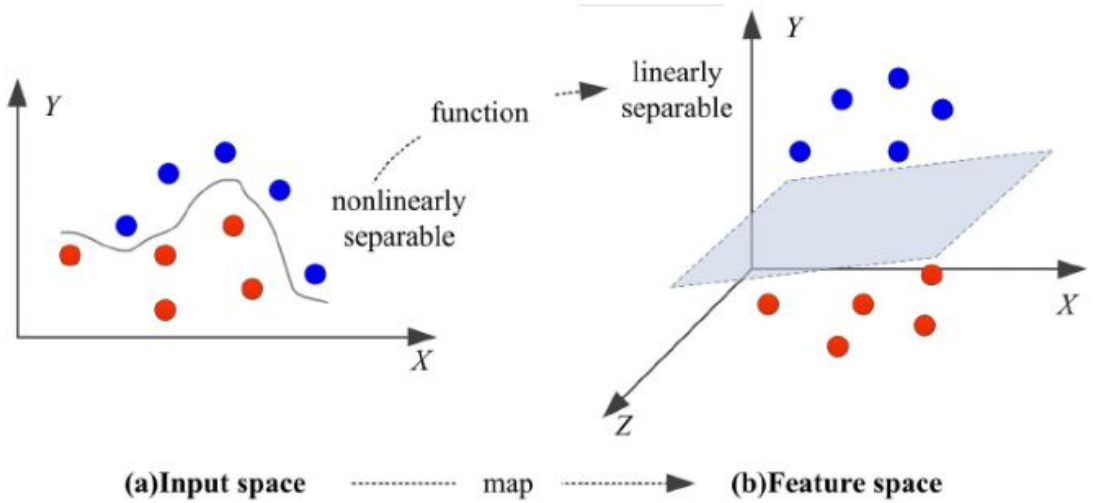
- What about non-linearly separable data?



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*

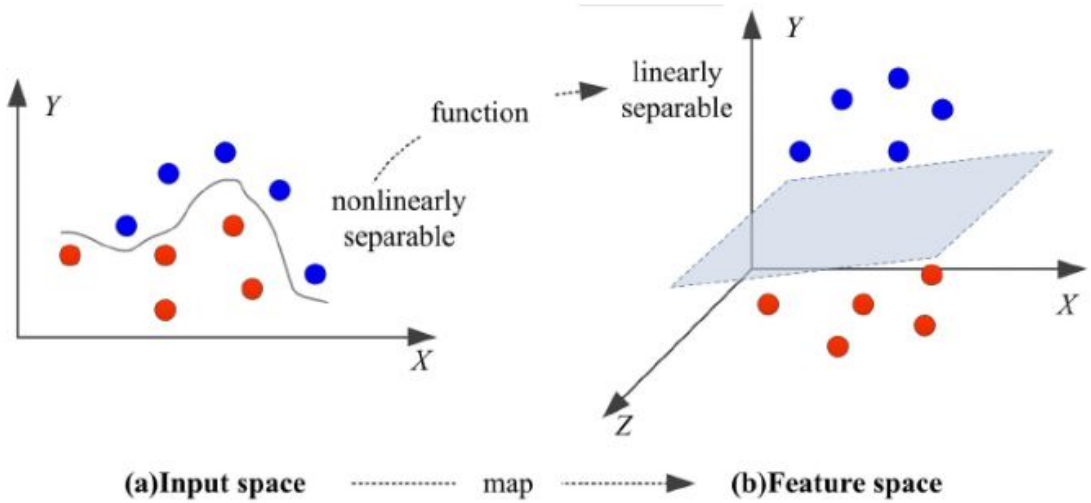
- What about non-linearly separable data?
  - you can add another feature to move to a higher dimension



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*

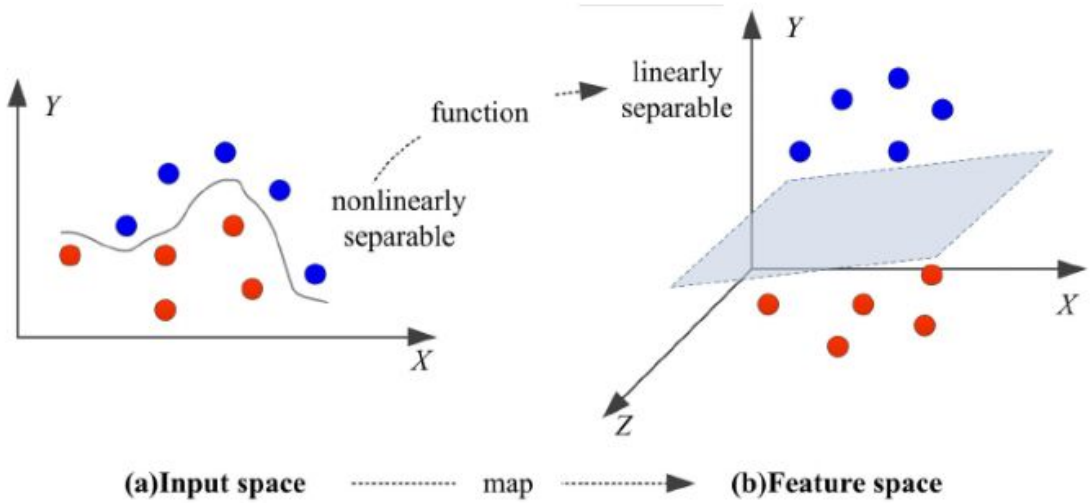
- What about non-linearly separable data?
  - you can add another feature to move to a higher dimension
    - to map some combination of other features



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
  - *train the classifier in this feature space*

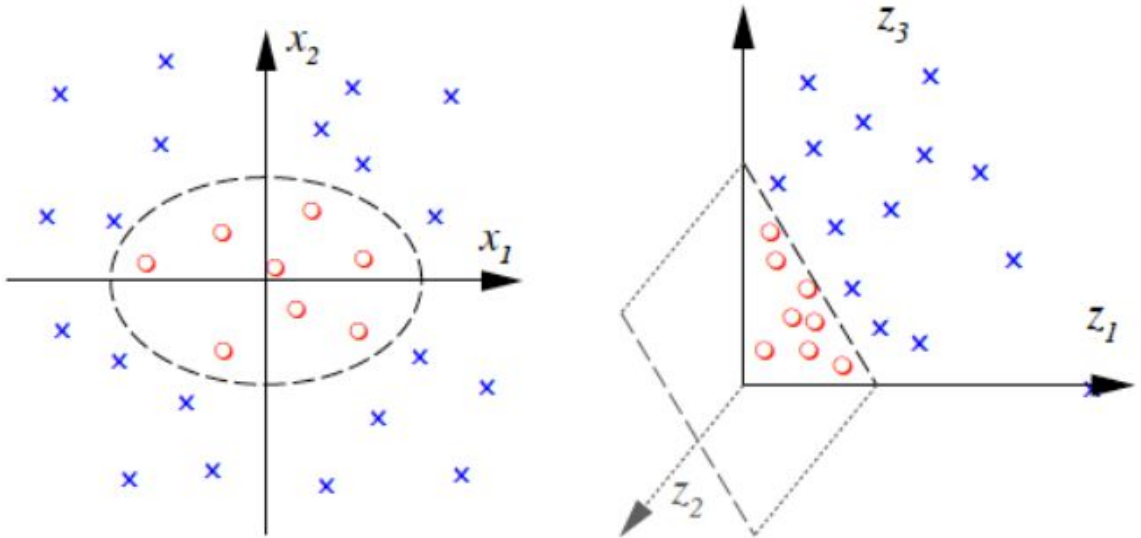


https://www.researchgate.net/publication/282920922_Heuristic_Methods_for_Reservoir_Monthly_Inflow_Forecasting_A_Case_Study_of_Xinfengjiang_Reservoir_in_Pearl_River_China

Example

$$\Phi : R^2 \to R^3$$

$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{(2)}x_1x_2, x_2^2)$$



https://math.stackexchange.com/q/474552

Feature mapping

- Consider regression model $y_i = w^T x_i + b$
  - w^T gives us a vector of weights

Feature mapping

- Consider regression model $y_i = w^T x_i + b$
  - w^T gives us a vector of weights
- The dot product is key to finding the classifier

Feature mapping

- Consider regression model $y_i = w^T x_i + b$
  - w^T gives us a vector of weights
- The dot product is key to finding the classifier
- So by using a higher dimension to separate the data:
  - x input vectors
    → map to higher dimension
    → dot product of mapped values
    → map back to original data dimension
    → run classifier/model on new data

Feature mapping

- Feature space often has a (much) higher dimension
- So this can be computationally expensive

Feature mapping

- Feature space often has a (much) higher dimension
- So this can be computationally expensive

➢ Kernels! $K(x_i, x_j) = \boldsymbol{\phi}(x_i)^T \boldsymbol{\phi}(x_j) \rightarrow$ K is a kernel function

- Intuitively, think of the kernel function as a dot product in the feature mapping space

Kernels

- Linear kernel

$$K(u, v) = u^T v$$

- Polynomial kernel (degree d)

$$K(u, v) = (\gamma u^T v + coef0)^d$$

- Radial basis kernel

$$K(u, v) = e^{(-\gamma ||u-v||^2)}$$

- Sigmoid

$$K(u, v) = \tanh(\gamma uv + coef0)$$

https://cran.r-project.org/web//packages/e1071/e1071.pdf

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*

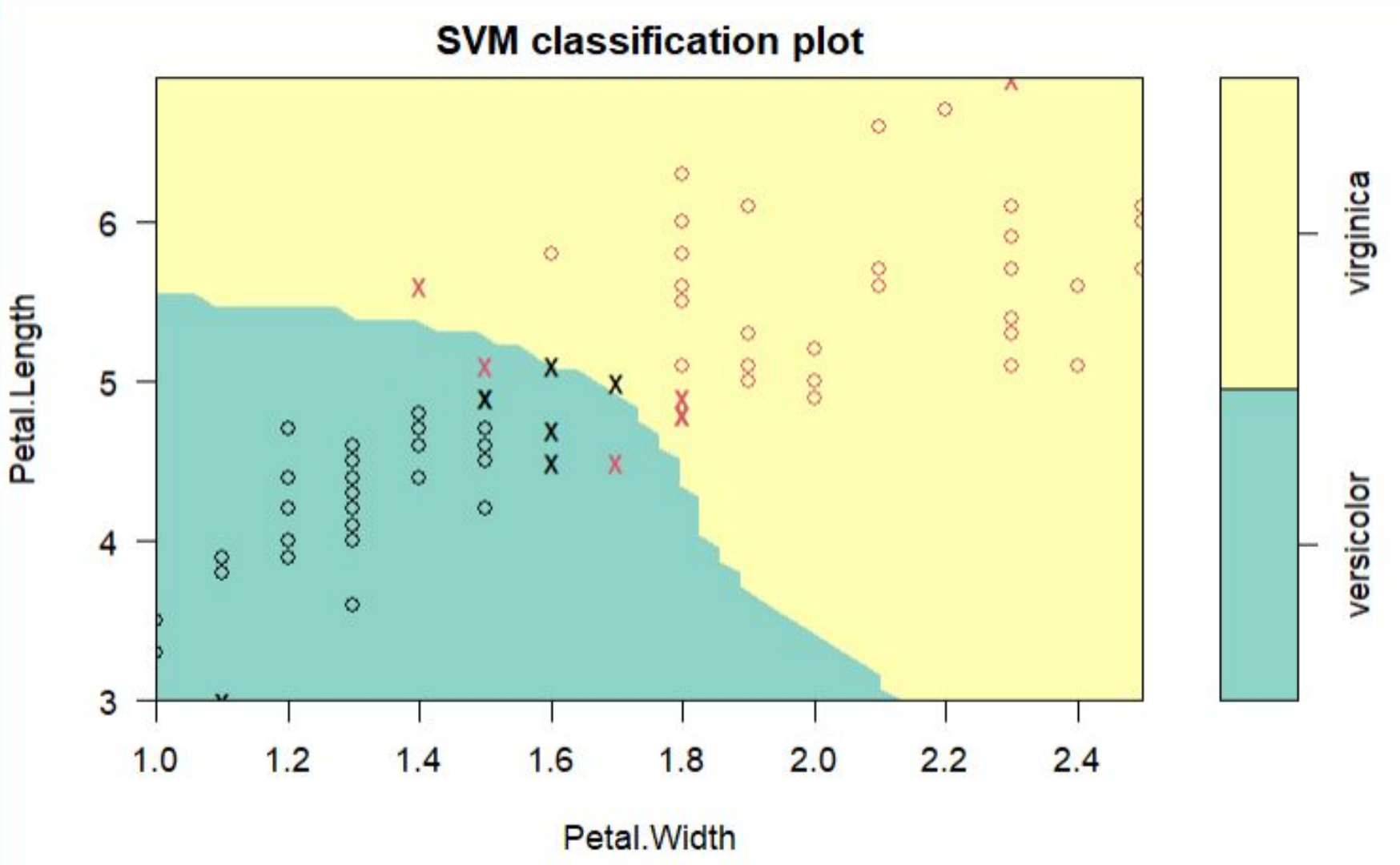
- We'll consider the *virginica* and *versicolor* labels
  - using a "radial" kernel
    - radial basis function kernel
      - think of it as a high dimensional feature mapping to a space where the decision boundary can be well-defined

```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
      - 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 (γ) = 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, 10, 100),
                              gamma = seq(0.5, 5, 0.5)))
summary(tune_svm$best.model)
```

Non-linearly separable data and SVMs *in R*

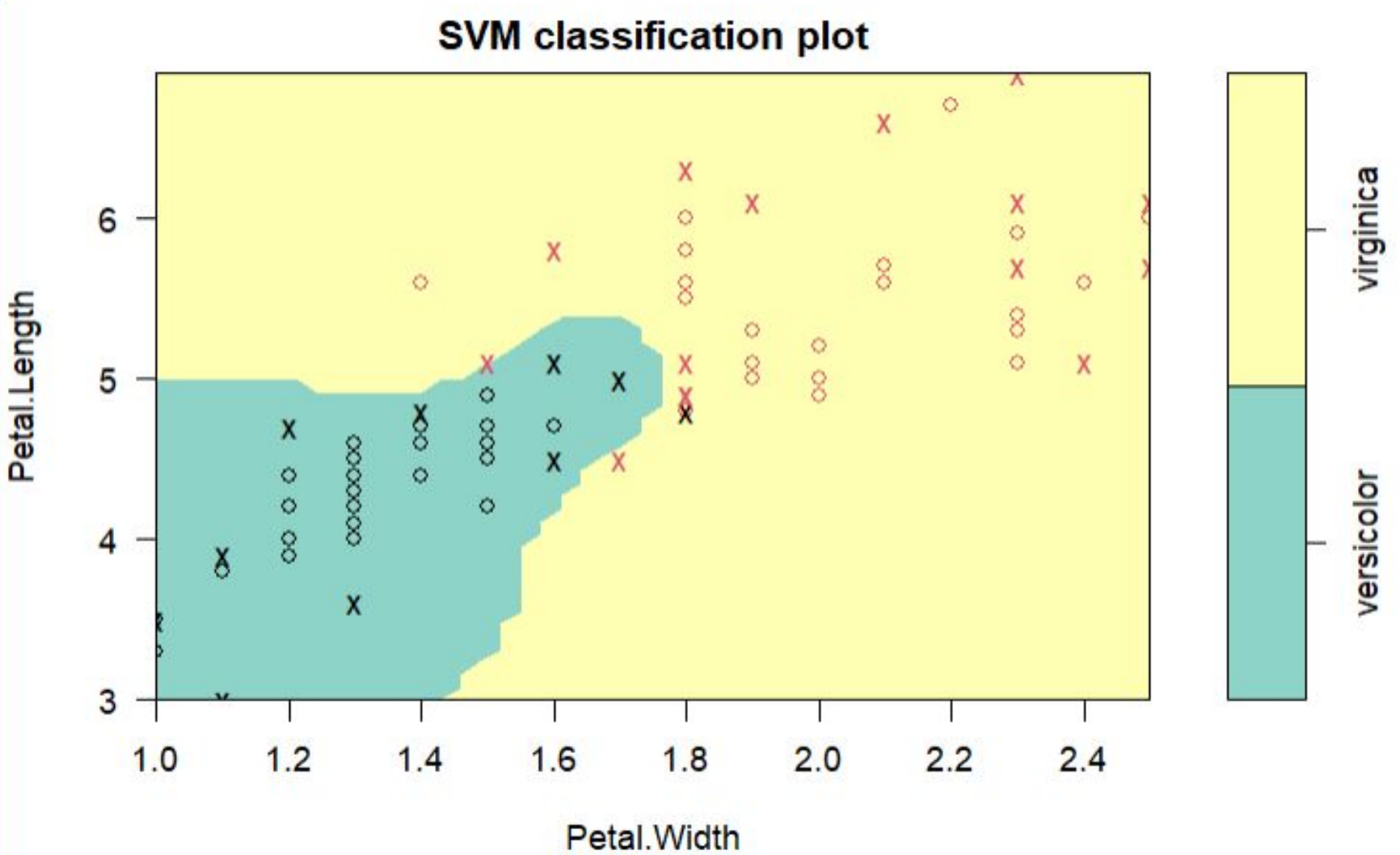
- The X's in this plot indicate which points were used to define the support vectors
- Again, we try hyper-parameter tuning
  - RBF kernel → need to supply cost and γ values

```
tune_svm = tune(method = svm, Species ~ .,
                data = training_nonlin, kernel = "radial",
                ranges = list(cost = c(0.001, 0.01, 0.1, 1, 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 γ values
    - γ indicates how much influence we want features to have on the decision boundary - high γ→ 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, 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 γ 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 (predict() is from *stats*)

```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?

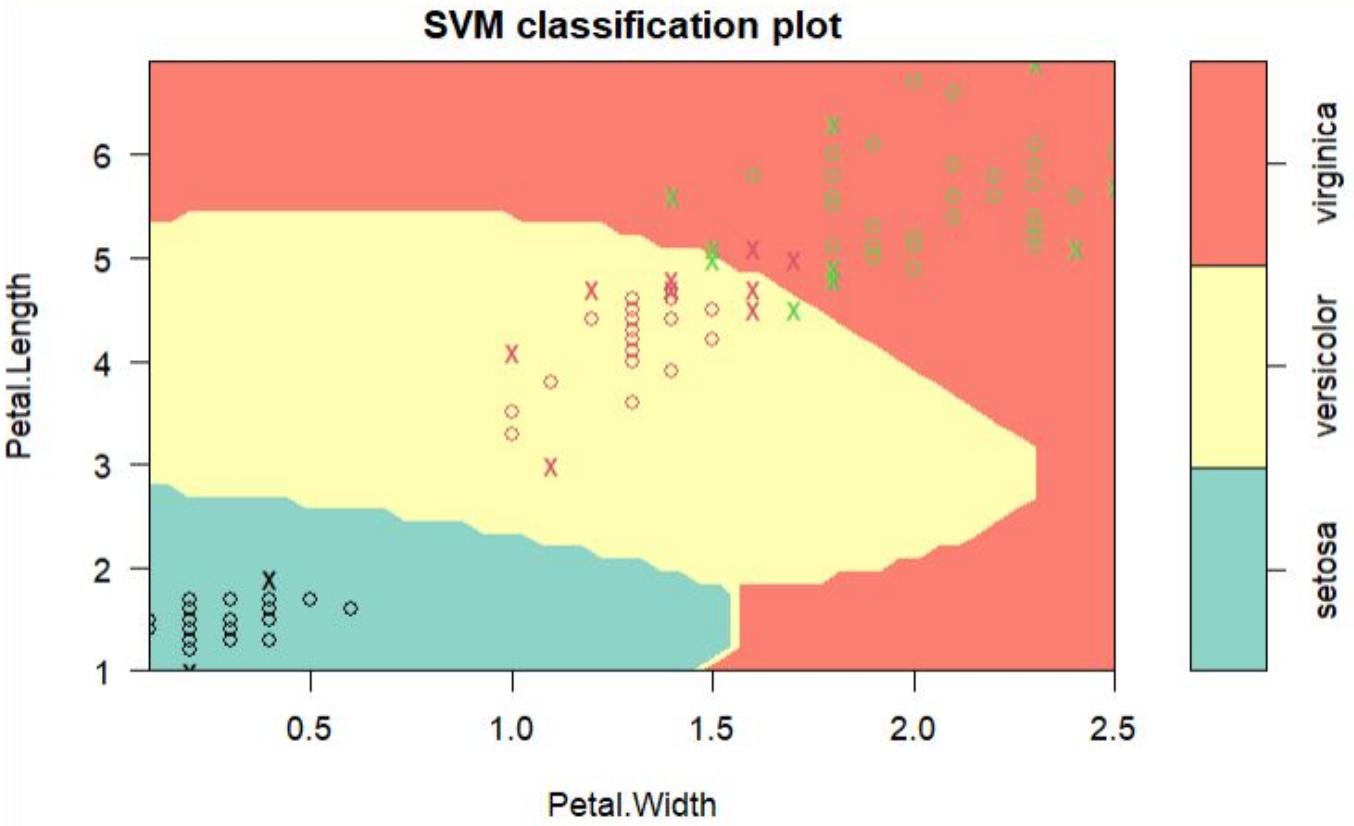
- “One against one”
  - learn a classifier for each pair of classes
  - how e1071’s svm() works with multiple classes

- “One against all”
  - learn each of the 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"))
```



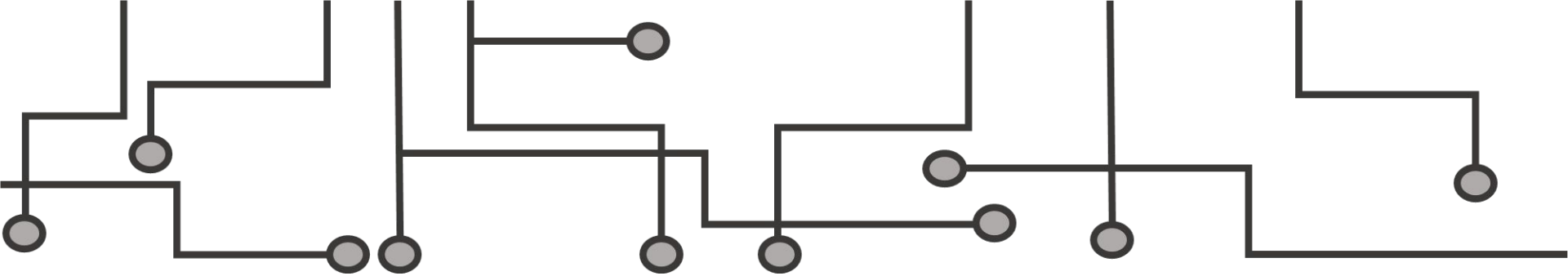
HELPFUL RESOURCES

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

[2]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