SUPPORT VECTOR MACHINES

E. Fersini

INTRODUCTION

- During this lab, we will illustrate the following:
    1. How to train a SVM
    2. Observing how the choice of cost can affect SVM
    3. Visualizing SVM fit
    4. Predicting the labels of a testing dataset based on the model trained by SVM
    5. Tuning SVM
- The two most well known and popular support vector machine tools are ***libsvm*** in the *e1071* package and ***SVMLite*** in the *klaR* package.
    - During this lab we will use libsvm
    - We will work on the iris dataset

TRAINING SVM

- Load the e1071 package:

```
> library(e1071)
```

- Load the dataset, create training and testing:

```
> ind = sample(2, nrow(iris), replace = TRUE, prob=c(0.7, 0.3))
> testset = iris[ind == 2,]
> trainset = iris[ind == 1,]
```

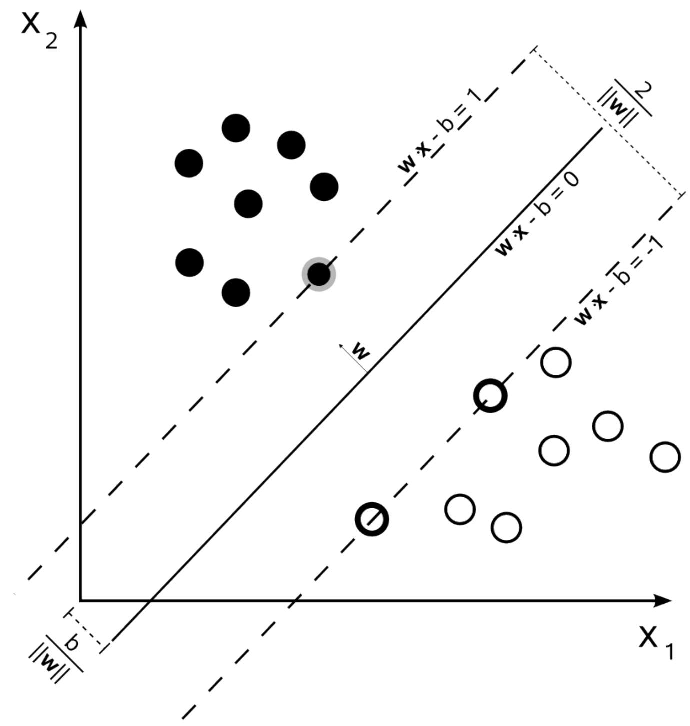
- Train the model:

```
> svm.model = svm(Species ~ ., data=trainset, kernel='linear', cost=1)
```

TRAINING SVM

- Our SVM constructs a hyperplane (or set of hyperplanes) that maximize the margin width between two classes.

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_i \xi_i$$



TRAINING SVM

- Our SVM constructs a hyperplane (or set of hyperplanes) that maximize the margin width between two classes.
- We can obtain overall information about the induced model with the print() function:

```
> print(svm.model)
```

- The SVM model has a cost parameter C that controls the trade-off between the slack variable penalty (misclassifications) and width of the margin:
  - a **small cost** creates a large margin (a soft margin) and allows **more** misclassifications;
  - a **large cost** creates a narrow margin (a hard margin) and allows **few** misclassifications.
  - C=infinity implies that all the constraints must be satisfied: no points within the margin

CHOOSE the COST of SVM

- With the following steps, we will understand how large and small cost will affect SVM.

- Performing the following steps to generate two different classification examples with different costs:
    1. Subset the iris dataset with columns named as Sepal.Length, Sepal.Width, Species, **only** with species in setosa and virginica

```
> iris.subset = subset(trainset, select=c("Sepal.Length", "Sepal.Width",
"Species"), Species %in% c("setosa","virginica"))
```

CHOOSE the COST of SVM

2. Then, you can generate a scatter plot with Sepal.Length as the x-axis and the Sepal.Width as the y-axis:

```
> plot(x=iris.subset$Sepal.Length,y=iris.subset$Sepal.Width,
col=iris.subset$Species, pch=19)
```

3. Next, we can train SVM based on iris.subset with the cost equal to 1:

```
> svm.model = svm(Species ~ ., data=iris.subset, kernel='linear', cost=1,
scale=FALSE)
```

CHOOSE the COST of SVM

4. Then, we can mark the support vectors with blue circles:

```
> points(iris.subset[svm.model$index,c(1,2)],col="blue",cex=2)
```

5. We can add a separation line on the plot:

```
> w = t(svm.model$coefs) %*% svm.model$SV
> c = -svm.model$rho
> abline (a=-c/w[1,2],b=-w[1,1]/w[1,2], col="red", lty=5)
```

intercept

slope

dotted line

CHOOSE the COST of SVM

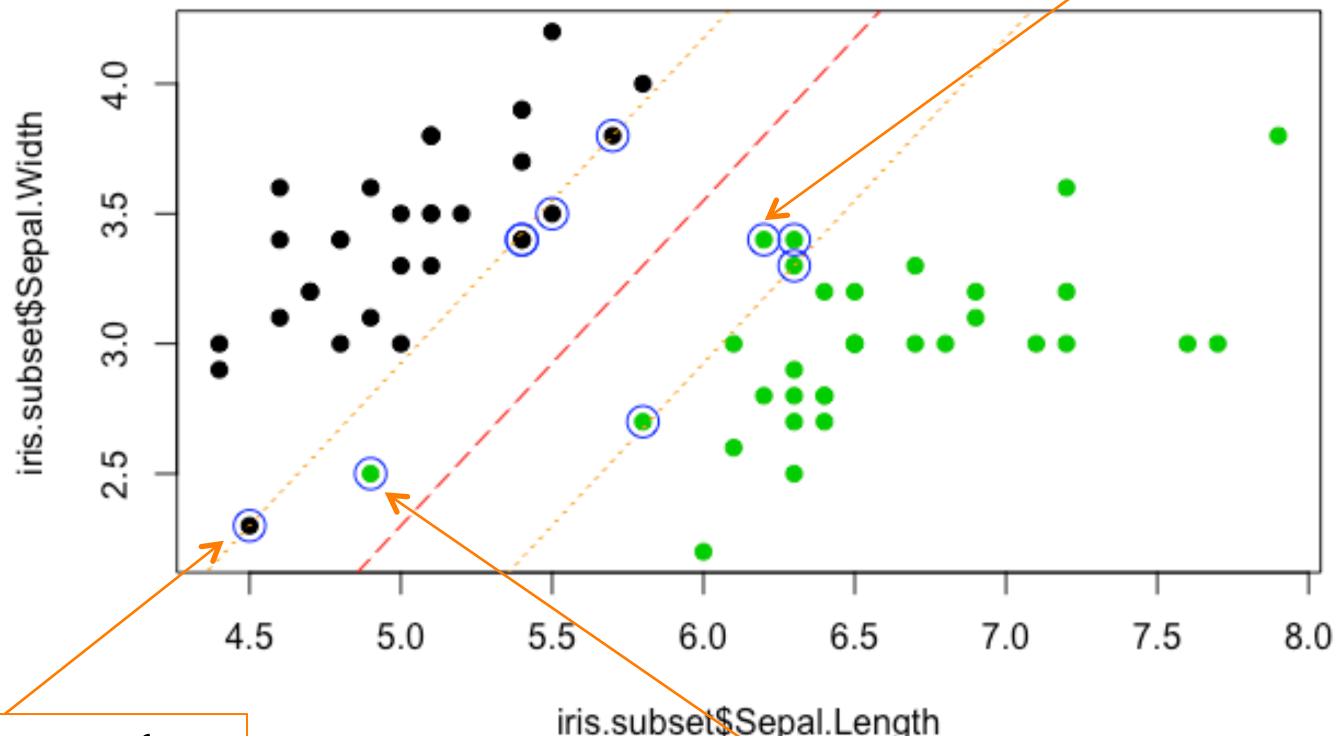
5. Lastly, we can add the margin on the plot:

```
> abline(a=(-c-1)/w[1,2], b=-w[1,1]/w[1,2], col="orange", lty=3)
> abline(a=(-c+1)/w[1,2], b=-w[1,1]/w[1,2], col="orange", lty=3)
```

CHOOSE the COST of SVM

- Our SVM looks as follows:

Slack variable $0 < \xi < 1$



Slack variable $\xi = 0$

Slack variable $\xi > 1$

CHOOSE the COST of SVM

- We can now work with a different cost parameter c=10,000
  1. Reset the plot:

```
> plot(x=iris.subset$Sepal.Length,y=iris.subset$Sepal.Width,
col=iris.subset$Species, pch=19)
```

  2. Train SVM with c=10000:

```
> svm.model = svm(Species ~ ., data=iris.subset, kernel='linear',
cost=10000, scale=FALSE)
```

CHOOSE the COST of SVM

3. Mark the support vectors with blue circles:

```
> points(iris.subset[svm.model$index,c(1,2)],col="blue",cex=2)
```

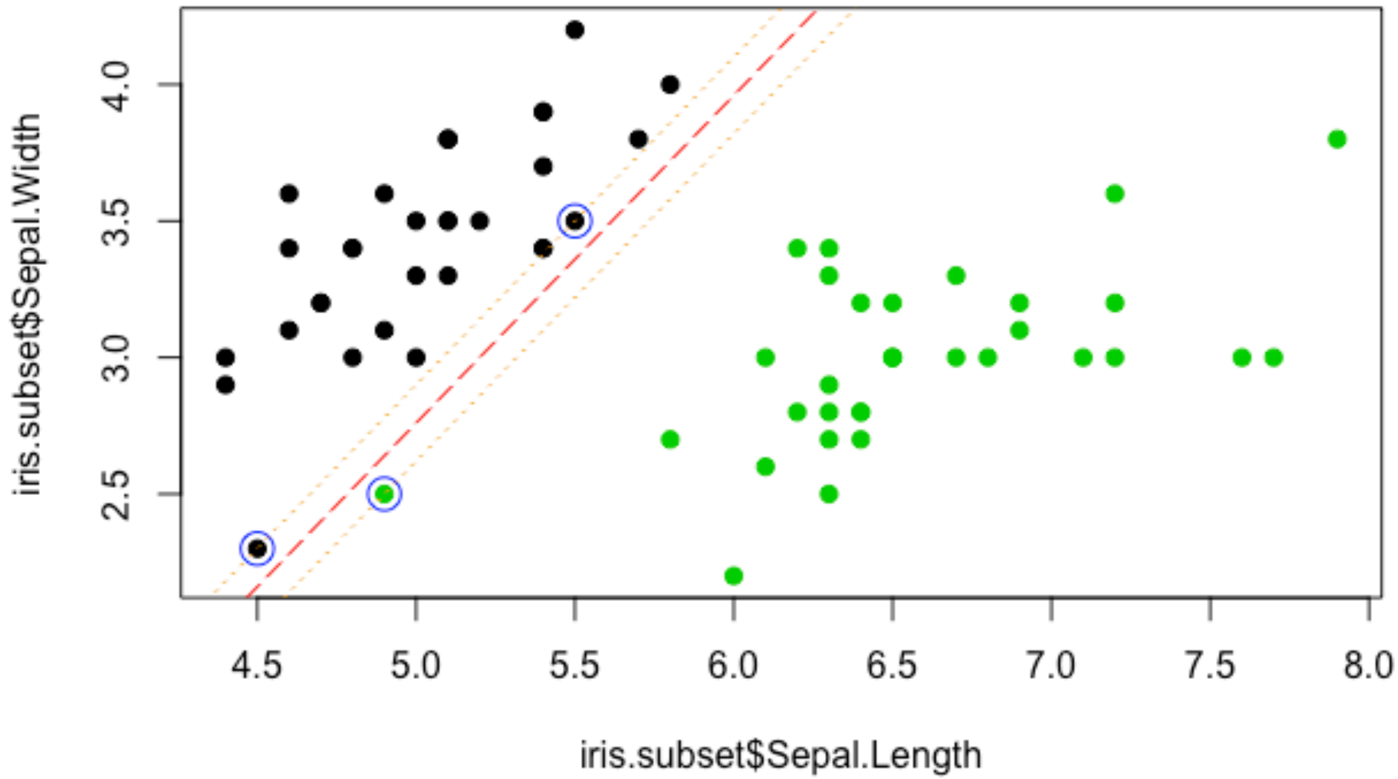
4. Add a separation line on the plot:

```
> w = t(svm.model$coefs) %*% svm.model$SV
> c = -svm.model$rho
> abline (a=-c/w[1,2],b=-w[1,1]/w[1,2], col="red", lty=5)
```

```
> abline(a=(-c-1)/w[1,2], b=-w[1,1]/w[1,2], col="orange", lty=3)
> abline(a=(-c+1)/w[1,2], b=-w[1,1]/w[1,2], col="orange", lty=3)
```

CHOOSE the COST of SVM

- Now, our SVM with c=10,000 looks as follows:



PREDICTION

- We can predict the label of new instances (testset) by using the already trained SVM model:

```
> svm.pred = predict(svm.model, testset)
```

- And create our confusion matrix:

```
> svm.table=table(svm.pred, testset$Species)
> svm.table
```

ASSIGNMENT

- Investigate the parameters:
  - **kernel**: radial, linear, polynomial, radial basis, sigmoid.
  - **cost**: the default value is 1; the larger the value, the smaller the margin is.
  - **gamma**: the default value is equal to (1/data dimension); increasing gamma implies an increasing number of support vectors.
    - it controls the shape of the separating hyperplane

ASSIGNMENT

- Define an SVM suitable for the “iris” dataset with the optimal parameter setting
  - Perform automatic tuning
  - With all the classes

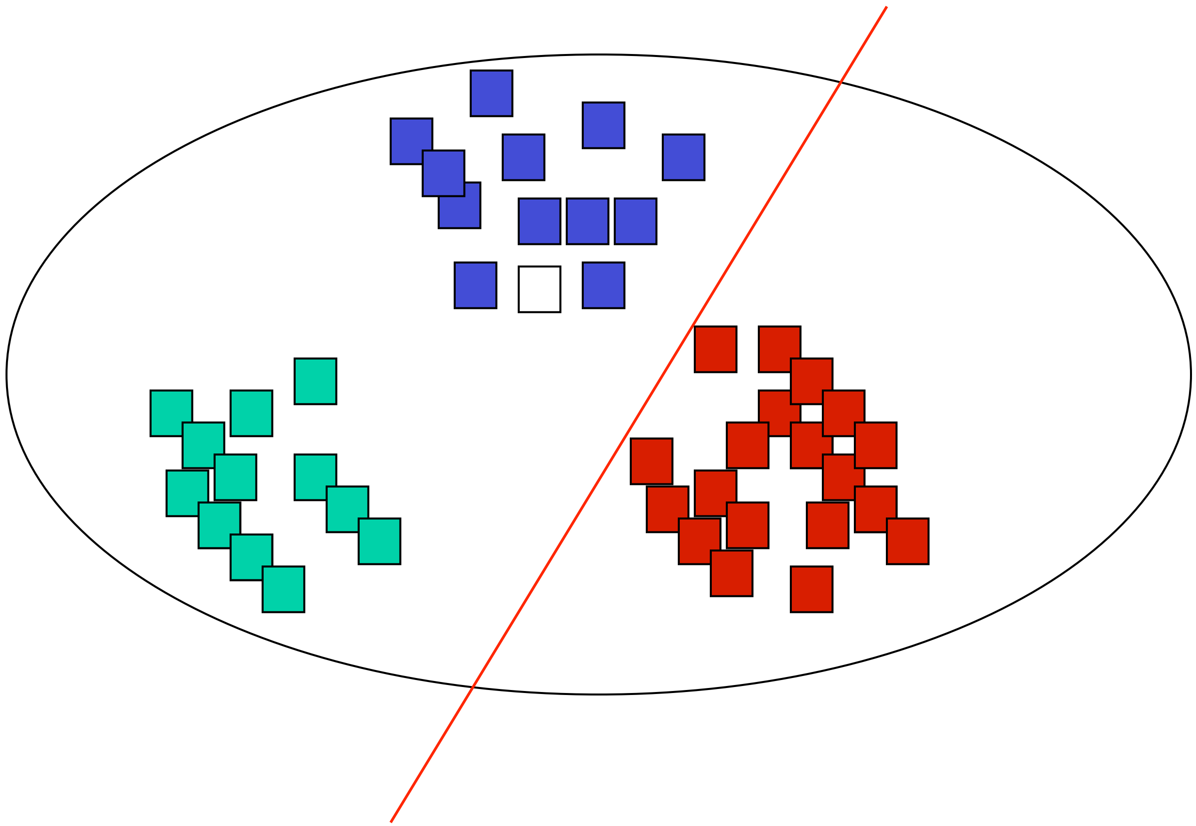
```
> tuned = tune.svm(Species ~ ., data = iris.subset,kernel='linear',
cost=c(0.001, 0.01, 0.1, 1,5,10,100))
```

DOING MULTI-CLASS CLASSIFICATION

- SVMs can only handle two-class outputs (i.e. a categorical output variable with arity 2).
- How to handle multiple classes
  - E.g., classify documents into three categories: *sports*, *business*, *politics*
- Solutions:
  - **one-vs-all**, learn N SVM's
    - SVM 1 learns "Output==1" vs "Output != 1"
    - SVM 2 learns "Output==2" vs "Output != 2"
    - …
    - SVM N learns "Output==N" vs "Output != N"
  - **Pairwise coupling**, learn N SVM's
    - SVM 1 learns "Output==1" vs "Output 2"
    - SVM 2 learns "Output==1" vs "Output != 3"
    - …
    - SVM N learns "Output==2" vs "Output != 3"
    - …..

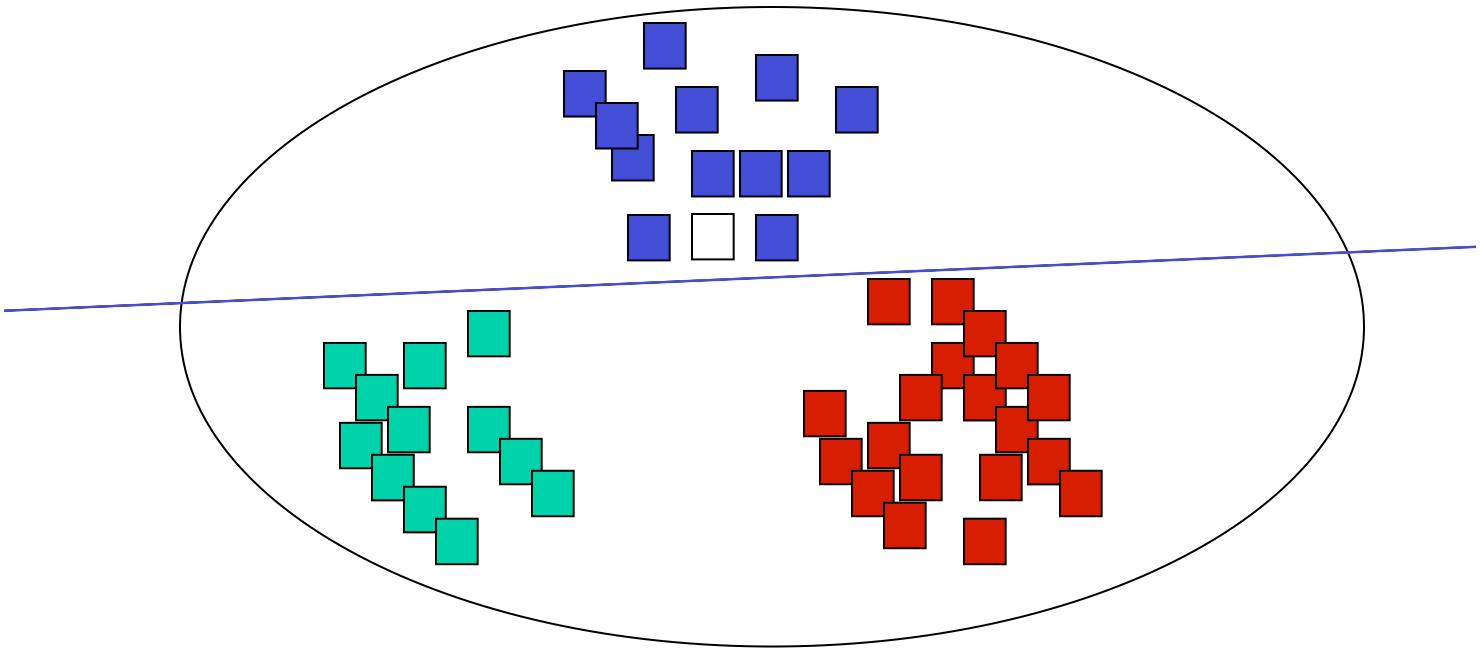
ONE-VS-ALL

- vs the other classes: red(d)



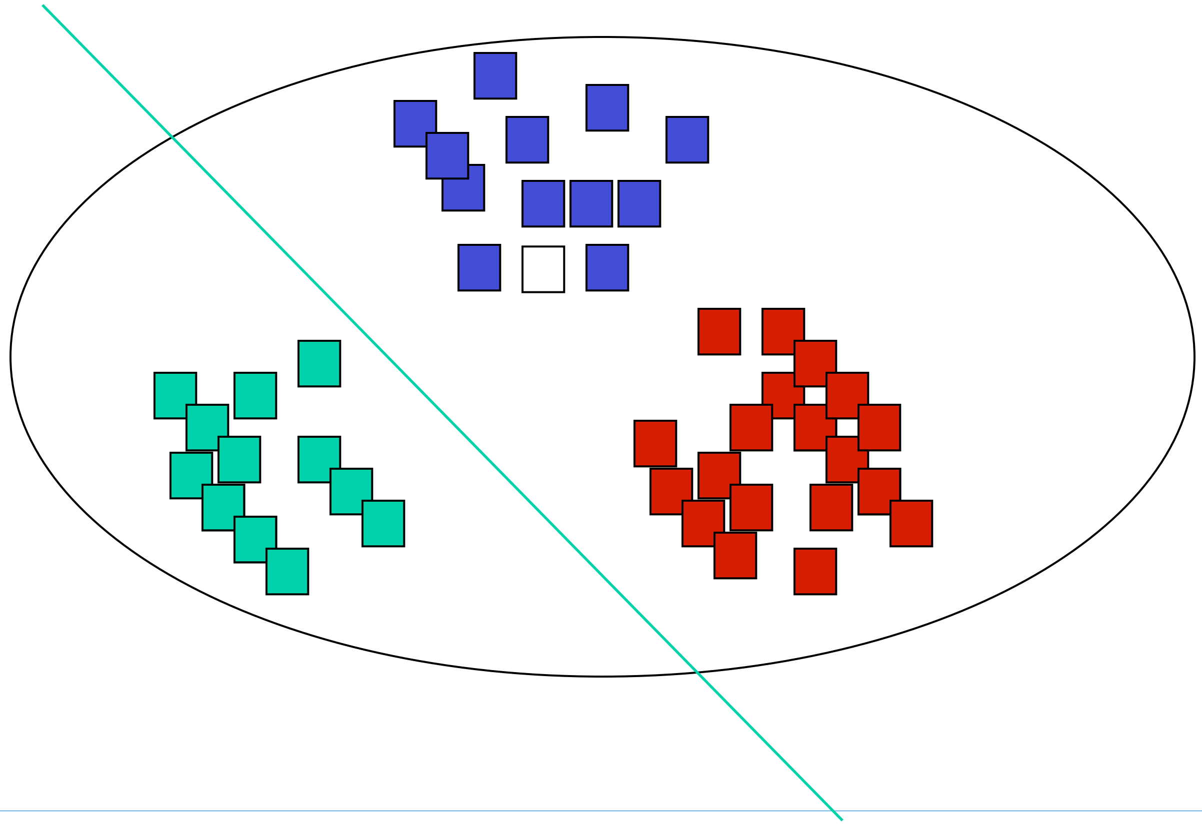
ONE-VS-ALL

- vs the other classes: red(d)
- vs the other classes: blue (d)



ONE-VS-ALL

- ■ vs the other classes: red(d)
- ■ vs the other classes: blue(d)
- ■ vs the other classes: cyan(d)



ONE-VS-ALL

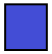
- vs the other classes: red(d)
- vs the other classes: blue(d)
- vs the other classes: cyan(d)

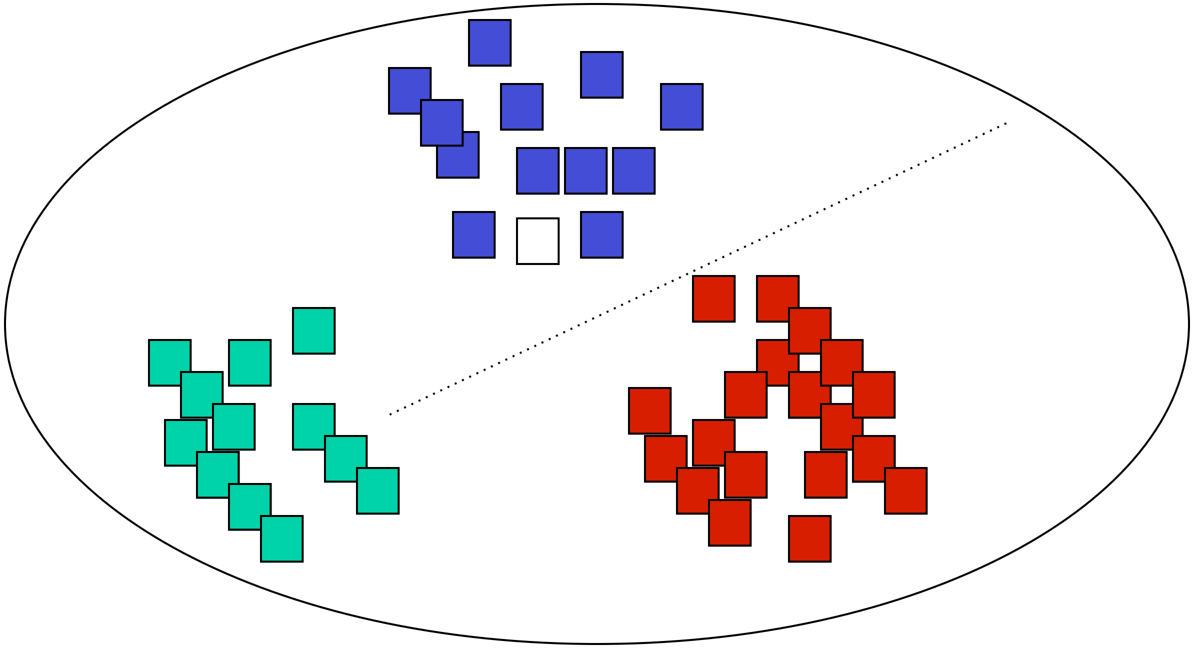
Given a test sample d', how to decide its color ?

*Assign d to the color function with the **largest score**!*

$$\boxed{\mathbf{w}^{\mathbf{T}}\mathbf{x}_{\mathbf{i}} + b}$$

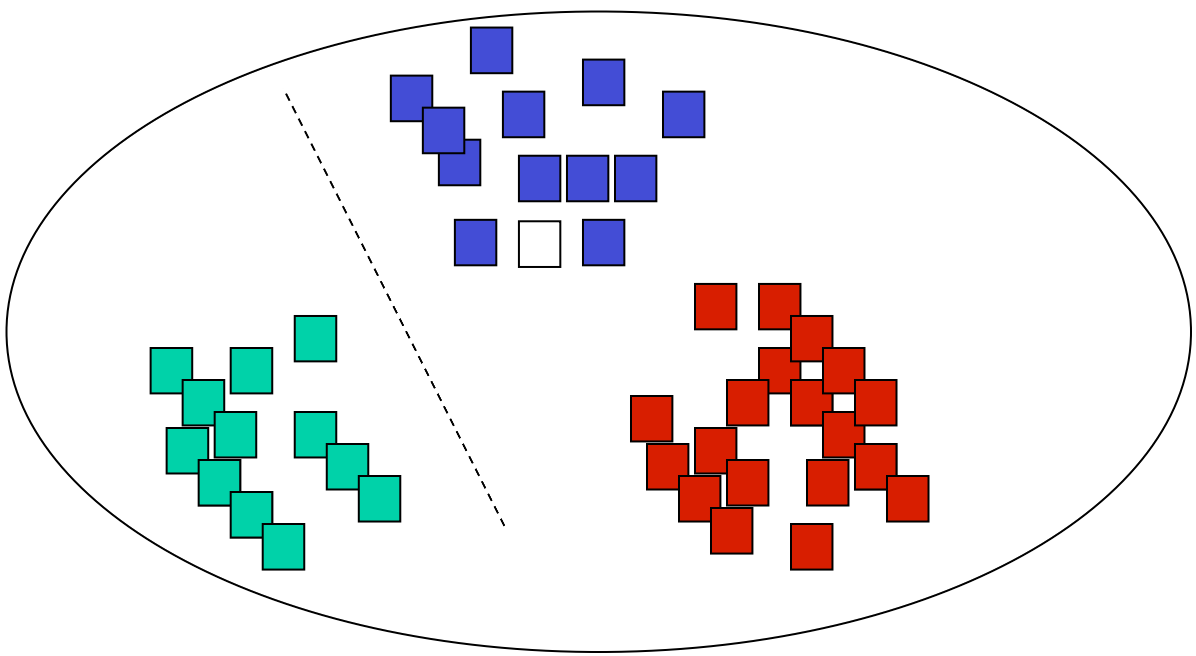
PAIRWISE COUPLING

-  vs  ....

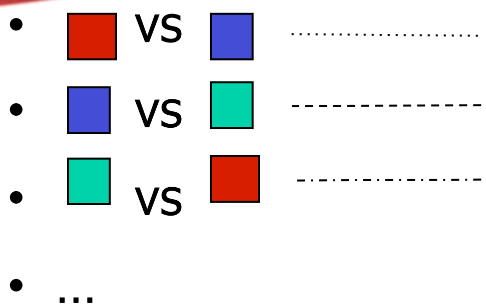


PAIRWISE COUPLING

- VS
- VS



PAIRWISE COUPLING



Which class for ☐ ?

Assign to the class that wins the most pairwise comparison