SUPPORT VECTOR MACHINES

Ref:

1) V. Vapnik. The Nature of Statistical Learning Theory. 2nd edition, Springer, 1999

2) Cortes and Vapnik

http://homepages.rpi.edu/~bennek/class/mmld/papers/svn.pdf

FURTHER REFERENCES

1) http://www.princeton.edu/~harman/Papers/SLT-tutorial.pdf
2) http://www.mit.edu/~6.454/www_spring_2001/emin/slt.pdf (Vapnik)
3) scikit-learn: http://scikit-learn.org/stable/modules/svm.html (Python3)
4) Kumar (2005), Chapter 5 Section 5.5
5) https://www.youtube.com/watch?v=ueKqDlMxueE&t=3s
6) https://www.youtube.com/watch?v=pS5gXENd3a4&t=336s
7) https://www.youtube.com/watch?v=-Z4aojJ-pdg

TOPICS

1) Understanding SVMs (the math part 1)

2) SVMs in R

3) The Math (part 2) of SVMs

4) Model Evaluation and ROC curves

WHAT IS AN SVM?

An SVM is a supervised machine learning technique.

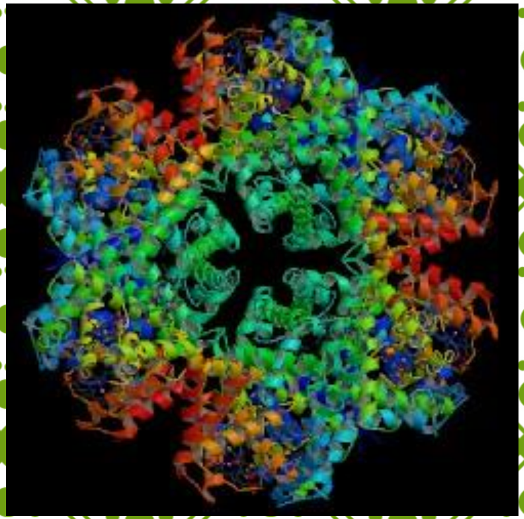
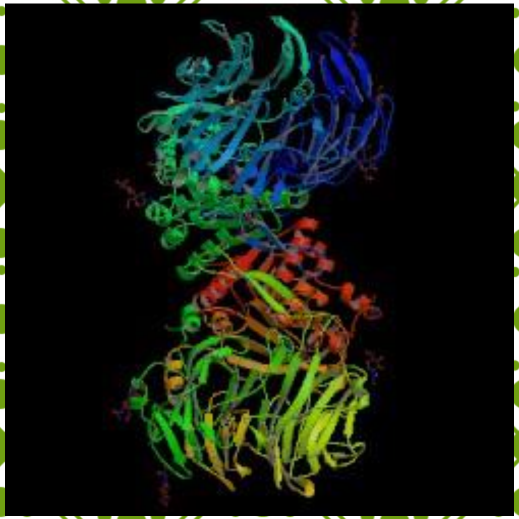
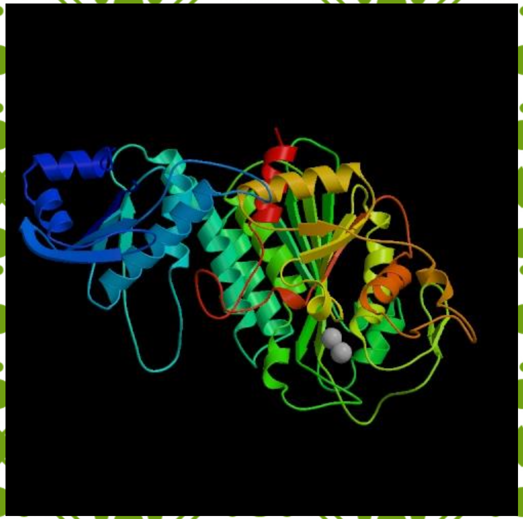
It uses known and labeled data to "train"

It uses different "**kernels**" options such as linear, gamma, sigmoid, and Gaussian.

SVM can perform feature transformation into higher (or infinite Hilbert Space) dimensional space so that input vectors (**x**) are **separable by hyperplane** (high dimensional plane).

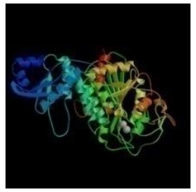
If you need data separated into more than two groups, you will need more than one SVM.

- Most "important" training points are **support vectors;** they define the hyperplane.
- **Quadratic optimization algorithms** can identify which training points x_i are support vectors with non-zero Lagrangian multipliers α_i. *(See the "math" slides for details)*

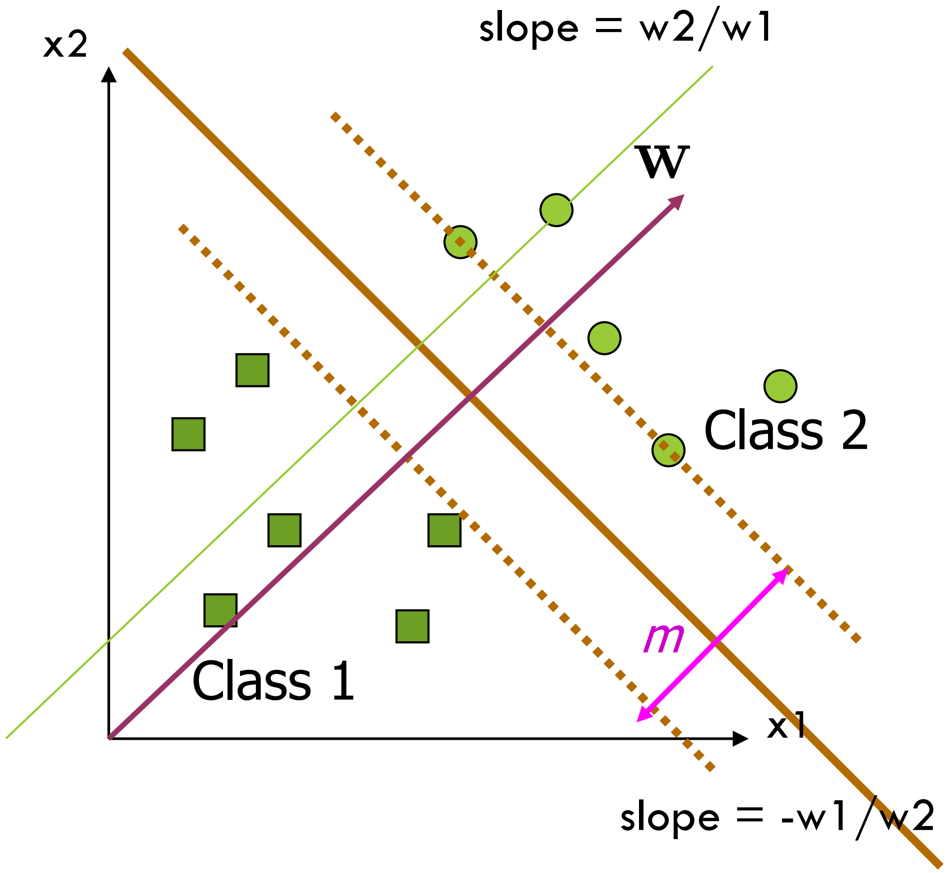
UNDERSTANDING SVMS

Professor Ami M. Gates



A REVIEW OF VECTOR MATH

- Any line in dimension D can be represented as $\mathbf{w}^T\mathbf{x} + b = 0$.
- **w** is a vector of coefficients, **x** is the vector of variables, b is the translation (can be thought of in 2D as the y intercept).

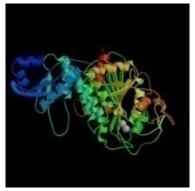


Suppose we are in 2D. (the common Cartesian coordinate system).

Then, a line can be written as
w1x1 + w2x2 + b = 0
$\rightarrow$ x2 = -b/w2 – (w1/w2)x1
$\rightarrow$ The "slope" is **–w1/w2**

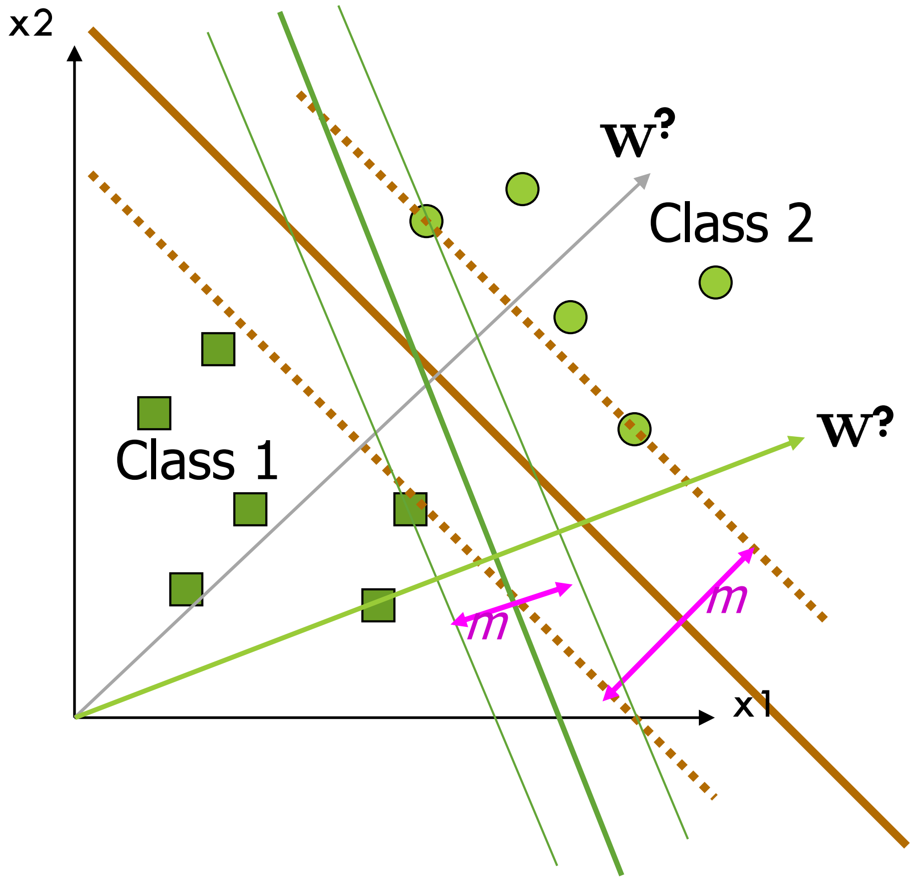
Given any vector **w** (in 2D, [w1,w2]**,** from the origin, a line parallel to **w** has slope w2/w1.

The line **perpendicular** to that vector has slope **-w1/w2**



A Review of Vector Math, cont.

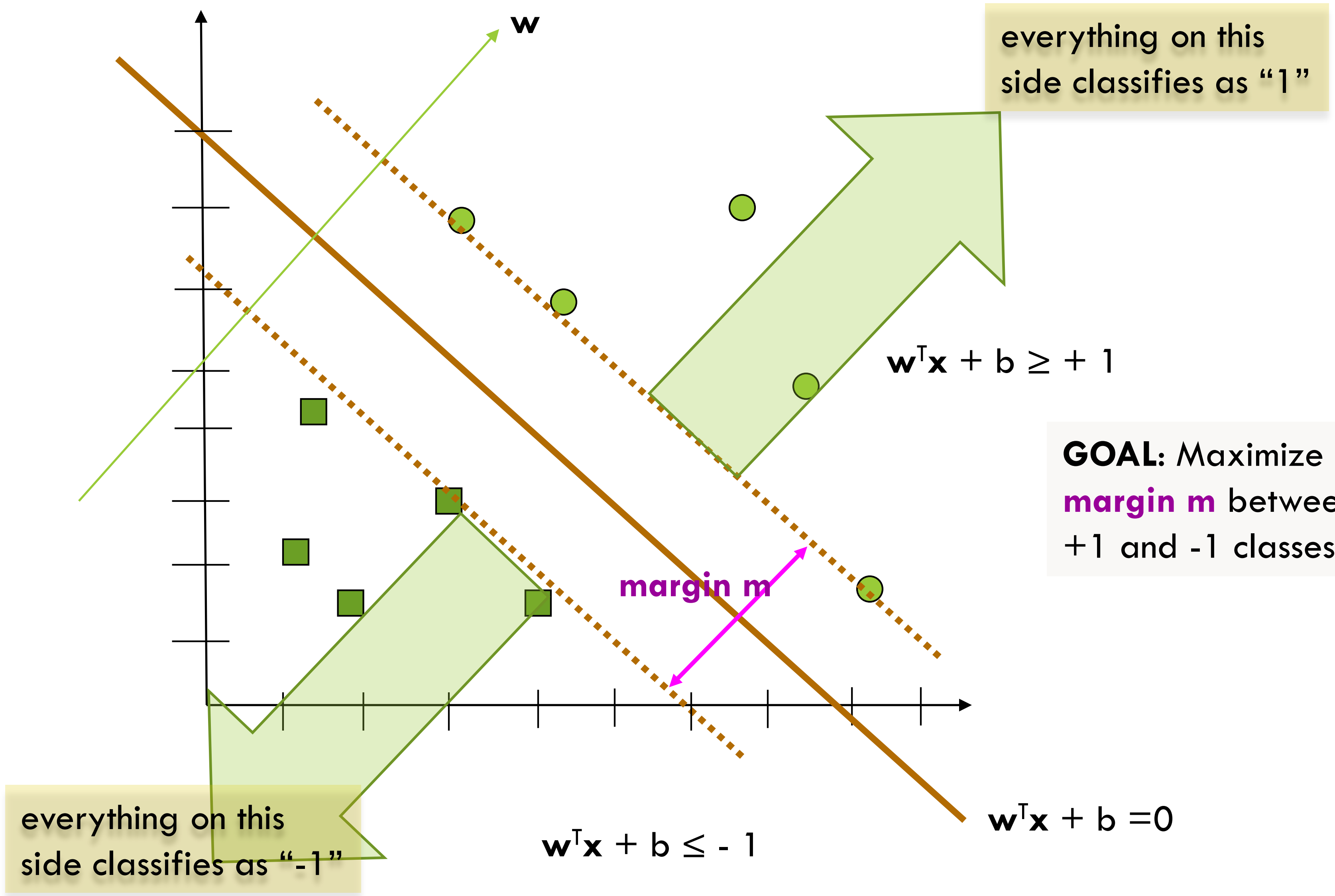
- The goal of an SVM is to determine (via training) the “best” line in 2D (or hyperplane in higher D) that **separates** two classes.
- **There are an infinite number of possible lines (hyperplanes)** that may work.



A general linear equation in two variables is:
$w1x1 + w2x2 + b = 0$
$\rightarrow\ w^{T}x + b = 0$

An SVM algorithm must determine the <u>weight vector w</u> that will result in a line (hyperplane) that <u>best separates the two classes.</u>

SETTING UP THE SVM PROBLEM



w
everything on this side classifies as "1"
$\mathbf{w}^T\mathbf{x} + b \geq +1$
margin m
everything on this side classifies as "-1"
$\mathbf{w}^T\mathbf{x} + b \leq -1$
$\mathbf{w}^T\mathbf{x} + b = 0$

GOAL: Maximize the **margin m** between the +1 and -1 classes

IMPORTANT CONCEPTS

The **margin m** can be created such that

w1x1 + w2x2 + b ≥ 1

→ $\mathbf{w}^T\mathbf{x} + b \geq 1$

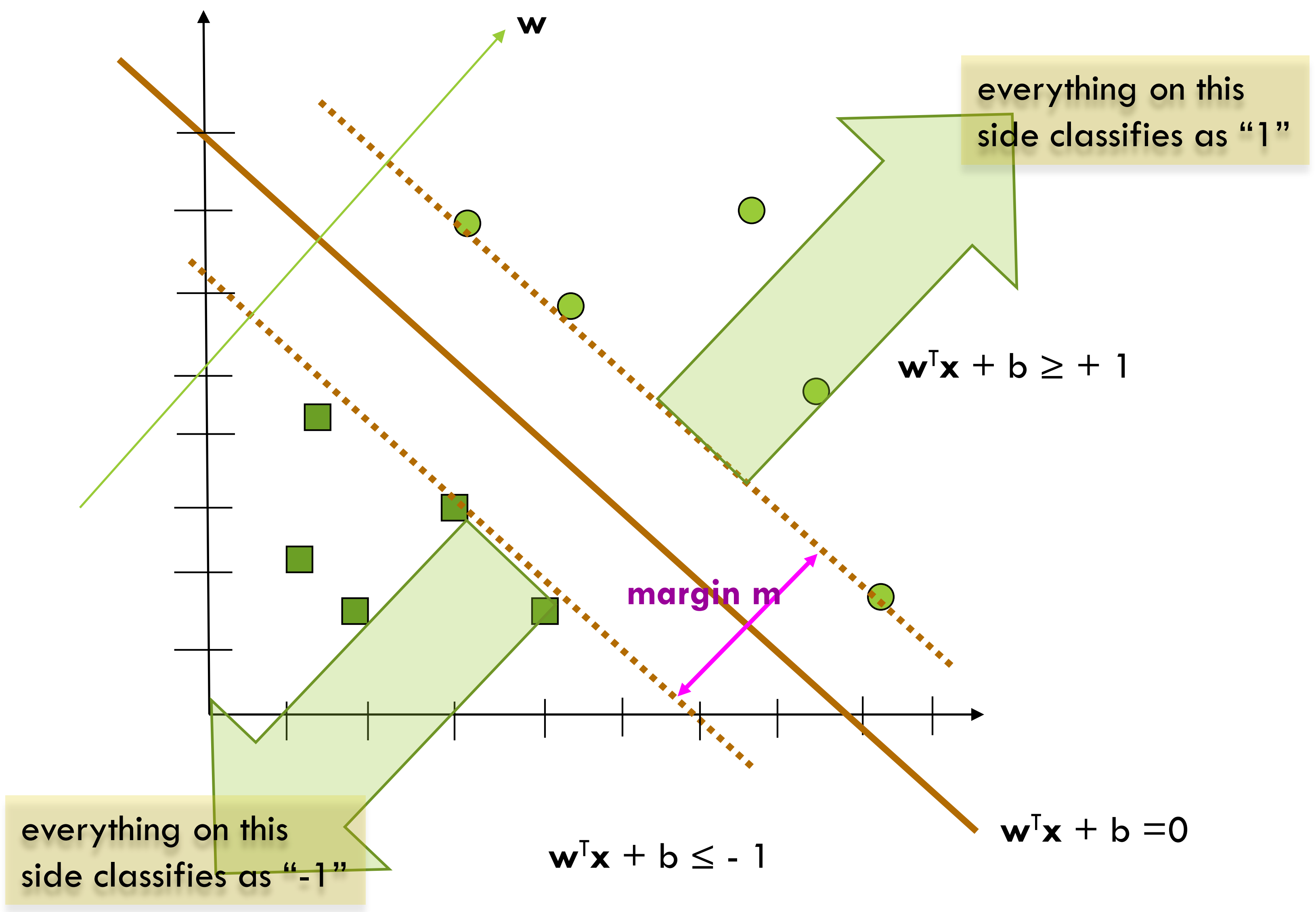
w1x1 + w2x2 + b ≤ -1

→ $\mathbf{w}^T\mathbf{x} + b \leq 1$

Remember that **w** will have to be determined (using the training data) so that the classes are “best” separated.

The value of “b” is a constant that affect the location (but not slope) of the line (in 2D).

THE **MARGIN M** BETWEEN THE TWO CLASSES SHOULD BE MAXIMIZED.


w
everything on this side classifies as "1"
$\mathbf{w}^T\mathbf{x} + b \geq +1$
margin m
$\mathbf{w}^T\mathbf{x} + b = 0$
$\mathbf{w}^T\mathbf{x} + b \leq -1$
everything on this side classifies as "-1"

KEY MATH CONCEPTS

1) Any line in 2D or hyperplane in higher D can be represented as:

$w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + \dots + w_nx_n + b = 0$

$\rightarrow$ same as: $\mathbf{w}^T\mathbf{x} + b = 0$

2) This linear equation can be represented with vector **w**, with vector **x**, and with b.

If vector **w** is plotted, it will be perpendicular to the linear equation above.

If vector **w** is altered, the "slope" (in 2D) or "rotation" (in higher D) is affected. If "b" is altered the "y intercept (in 2D) or the "translation" (in high D) is affected.

3) We can use $\mathbf{w}^T\mathbf{x} + b = 0$ to try different lines. To do this, we need **w** and b and a measure of "best".

THE SVM MARGIN M

- Assume that in some dimension (more on this later) that we have two classes that can be linearly separated.
- Then, there must be a line (hyperplane) that separates the classes.
- This line is defined by $\mathbf{w}^T\mathbf{x} + b = 0$.
- This line can be contained within two parallel lines that create a “margin” between the two classes.
- The two parallel lines can initially be defined as:

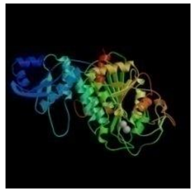
  $\mathbf{w}^T\mathbf{x} + b = 1$ and $\mathbf{w}^T\mathbf{x} + b = -1$

  (Note that b is a constant that can be altered as needed)

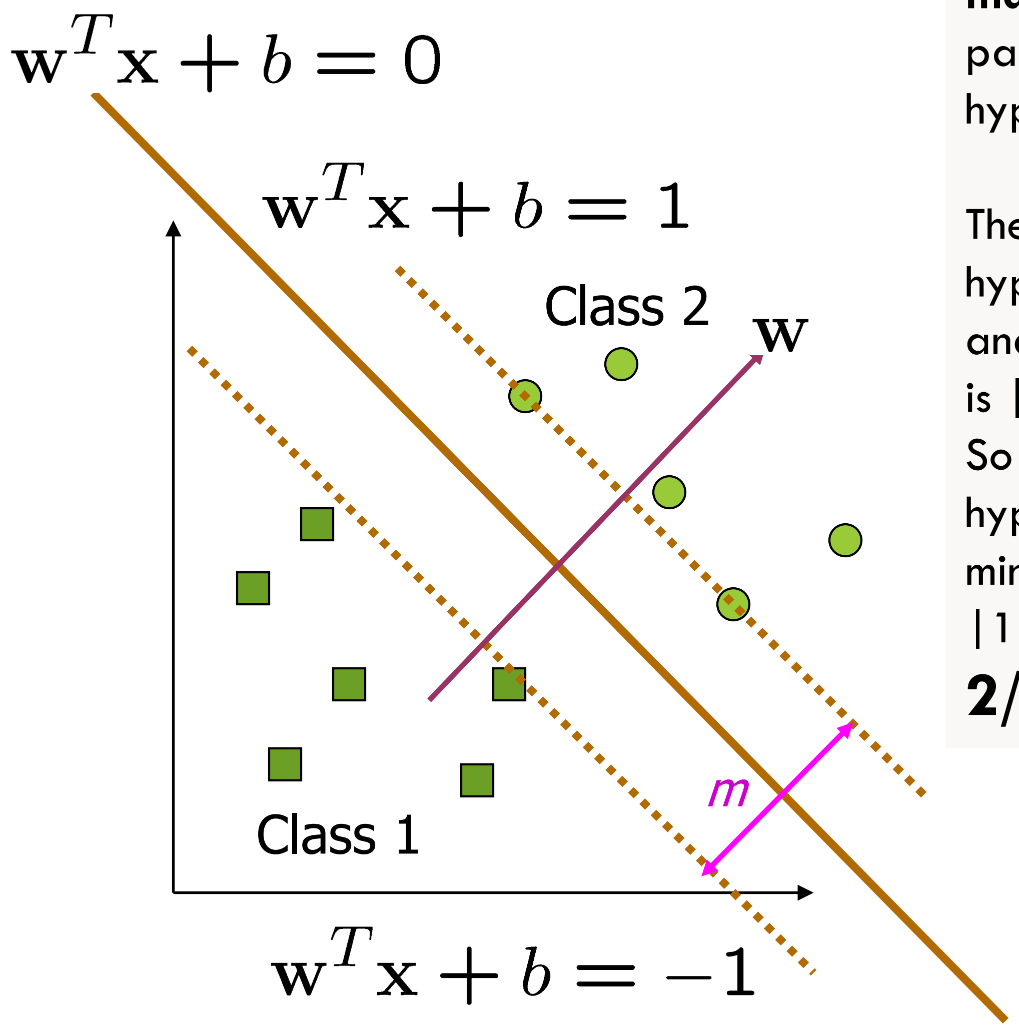
  - Given a weight vector **w** and an input data vector **x**,

  if the value of $\mathbf{w}^T\mathbf{x} + b > 1$, it is classified into one Class

  if $\mathbf{w}^T\mathbf{x} + b < -1$ it is classified into the other Class.



MAXIMIZING MARGIN M WILL CREATE THE "BEST" SEPARATION BETWEEN THE CLASSES.

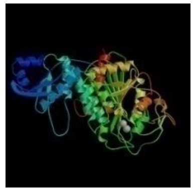


$\mathbf{w}^T\mathbf{x} + b = 0$
$\mathbf{w}^T\mathbf{x} + b = 1$
Class 2
$\mathbf{w}$
m
Class 1
$\mathbf{w}^T\mathbf{x} + b = -1$

margin m is the **distance** between the two parallel boundary lines (or two parallel hyperplanes)

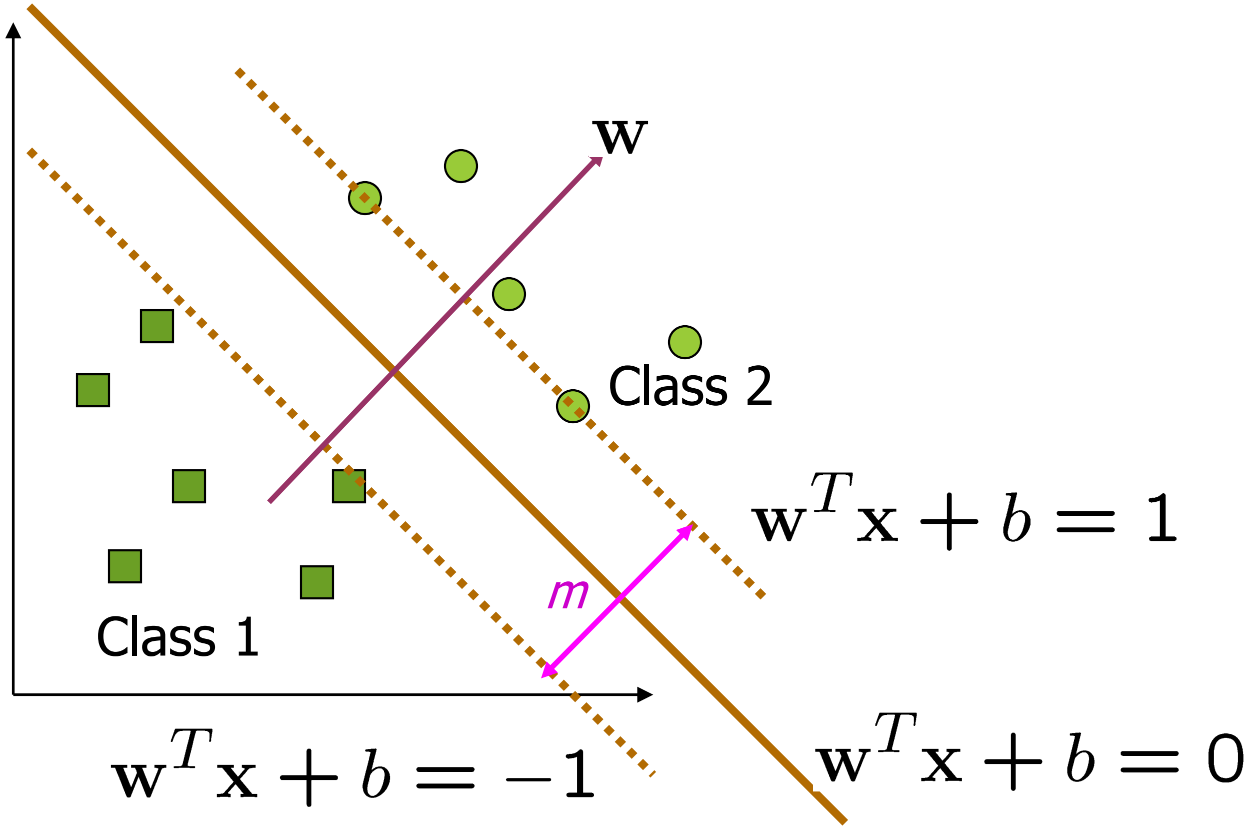
The distance between the origin and a hyperplane in 2D is the dist between (0,0) and $\mathbf{w}^T\mathbf{x}$ + b = 1
is |1-b| / ||**w**||
So the distance between the two hyperplanes is dist from origin to the first minus the dist from origin to other.
|1 – b| - |-1 – b| / ||w|| =
2/||w||

Image: Martin Law Michigan State university



MAXIMIZING MARGIN M

To maximize the margin m = 2/||w||
(same as $2/w^Tw$) → We need to minimize w^Tw



$\mathbf{w}$
Class 2
Class 1
m
$\mathbf{w}^T\mathbf{x} + b = 1$
$\mathbf{w}^T\mathbf{x} + b = 0$
$\mathbf{w}^T\mathbf{x} + b = -1$

Image: Martin Law Michigan State university

THE OPTIMIZATION PROBLEM

Find the **<u>min</u>** of $\mathbf{w}^T\mathbf{w}$ and **b**

Such that:

$\mathbf{yi}\ (\mathbf{w}^T\mathbf{xi} + \mathbf{b}) \geq \mathbf{1}$ for all input vectors **xi**

Recall that yi **will be +1 or -1 (depending on the class)**

This is a **quadratic** optimization problem.

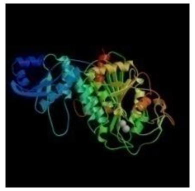
THE SVM PROBLEM

Let $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be our data set and let $y_i \in \{1,-1\}$ be the **class** label of some $\mathbf{x}_i$

Note that is a $\mathbf{x}_i$ vector.

The decision boundary should classify all points correctly and will maximize the margin (minimize $||\mathbf{w}||$ with constraint:)

The decision boundary can be found by **solving the following constrained optimization problem**



A SOFT MARGIN HYPERPLANE SVM

If we minimize $\sum_i \xi_i$,

ξ_i can be computed by

$$\begin{cases} \mathbf{w}^T\mathbf{x}_i + b \geq 1 - \xi_i & y_i = 1 \\ \mathbf{w}^T\mathbf{x}_i + b \leq -1 + \xi_i & y_i = -1 \\ \xi_i \geq 0 & \forall i \end{cases}$$

ξ_i are "**slack variables**" in optimization

- Note that $\xi_i=0$ if there is no error for $\mathbf{x}_i$

ξ_i is an upper bound of the number of errors

- We want to **minimize: $\mathbf{w}^T\mathbf{w} + C \sum_i \xi_i$**

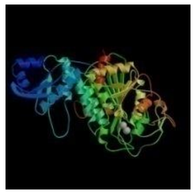
- C : tradeoff parameter between error and margin

The optimization problem becomes

$$\mathbf{w}^T\mathbf{w} + C \sum_i \xi_i$$

$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

Image: Martin Law Michigan State university



EXAMPLE NON-LINEAR

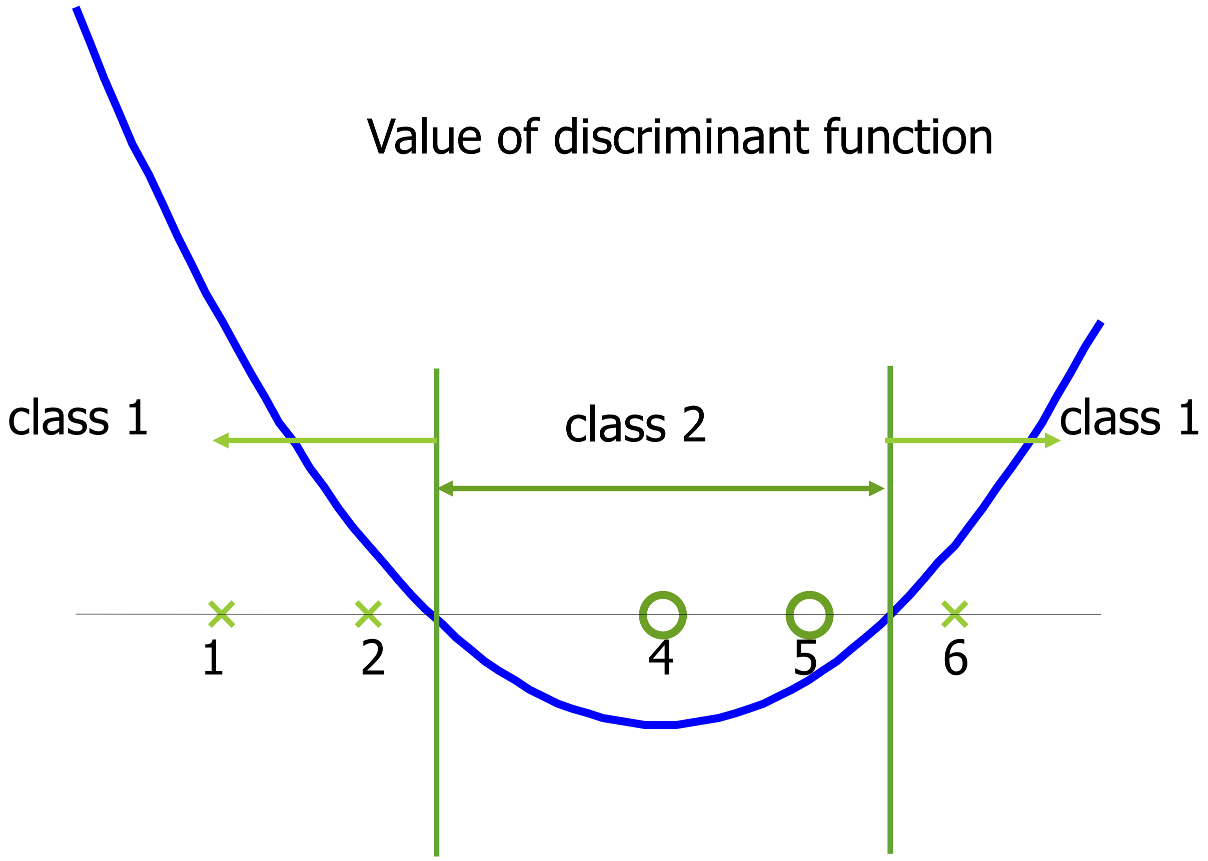


Image: Martin Law Michigan State university

KERNELS

Because the solution to the SVM optimization problem (the Dual Form) involves only **inner products**, the data values can be "transformed" to a different dimension using a kernel.

The Dual Representation:

$$L'(a) = \Sigma_{n=1 \text{ to } N} \; a_n - \tfrac{1}{2} \Sigma_{n=1 \text{ to } N} \Sigma_{m=1 \text{ to } N} a_n a_m t_n t_m \phi(x_n)^T \phi(x_m)$$

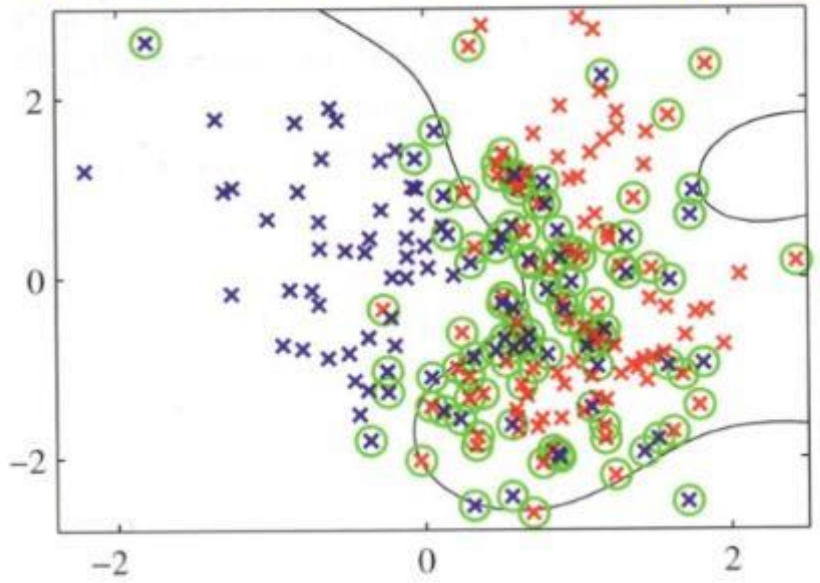
Kernel: $K(x_{n,} x_m) = \phi(x_n)^T \phi(x_m)$

This can be thought of as a virtual transformation as the points themselves do not have to be projected.

This is th

$$K(x_i, x_j) = exp(-\gamma \parallel x_i - x_j \parallel^2), \gamma > 0,$$

SVM Soft Margin Decision Surface using Gaussian Kernel



[from Bishop, figure 7.4]

$$\hat{f}(\mathbf{x}) = \mathbf{w}^T \Phi(\mathbf{x}) + b$$

Circled points are the *support vectors*: training examples with non-zero α_l

Points plotted in original 2-D space.

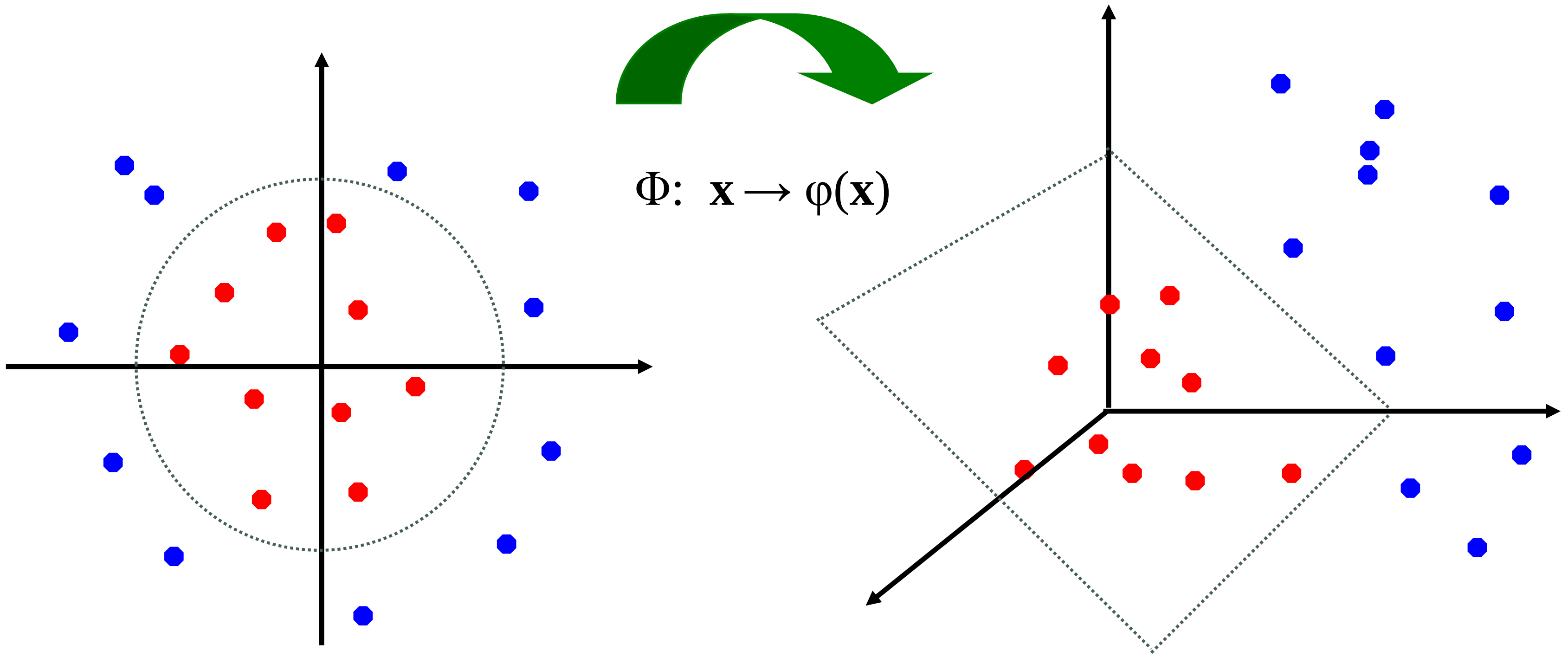
Contour lines show constant $\hat{f}(\mathbf{x})$

$$\hat{f}(\mathbf{x}) = b + \sum_{l=1}^{M} \alpha_l \; y_l \; \kappa(\mathbf{x}, \mathbf{x}_l) = b + \sum_{l=1}^{M} \alpha_l \; y_l \exp(-\|\mathbf{x} - \mathbf{x}_l\|^2 / 2\sigma^2)$$

Compliments of: https://www.cs.cmu.edu/~tom/10701_sp11/slides/Kernels_SVM2_04_12_2011-ann.pdf

Non-linear SVMs: Feature spaces

- General idea: the original input space can always be **mapped to some higher-dimensional feature space** where the training set is separable:



ADVANTAGES AND DISADVANTAGES

SVM:

1) supervised learning method
2) for classification, regression, outlier detection

Advantages:

1) Effective in high D space
2) Effective when dimension > # samples
3) Uses "subset" of training data – just the "support vectors"
4) Allows or the use of kernel functions

Disadvantages:

1) If features >> samples → poor performance
2) No prob estimates – these must be "done by hand" using at least 5-fold cross validation

PROPERTIES OF SVM

Flexibility in choosing a similarity function

Sparseness of solution when dealing with large data sets

- only support vectors are used to specify the separating hyperplane

Ability to handle large feature spaces

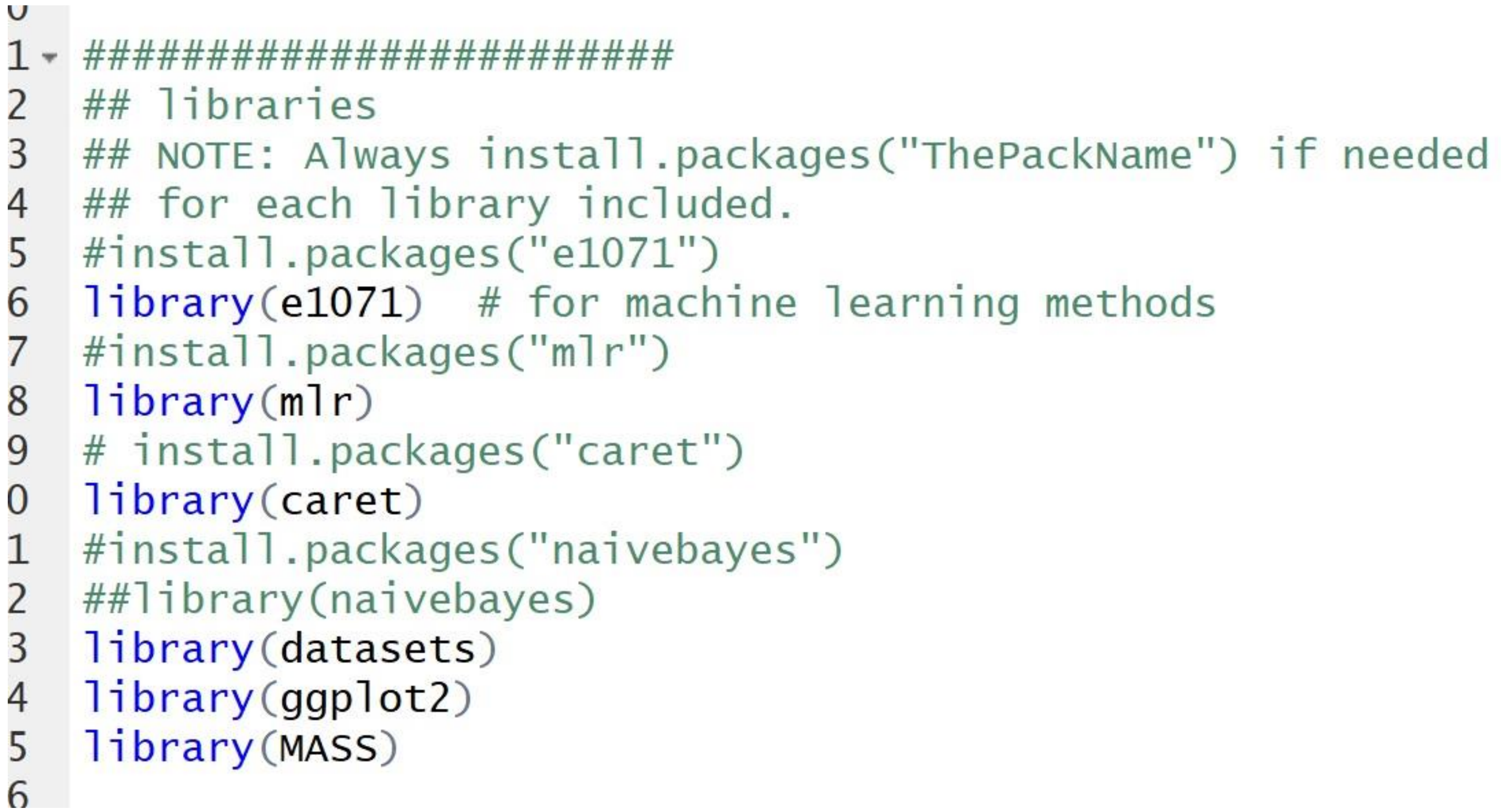
- complexity does not depend on the dimensionality of the feature space

Overfitting can be controlled by soft margin approach

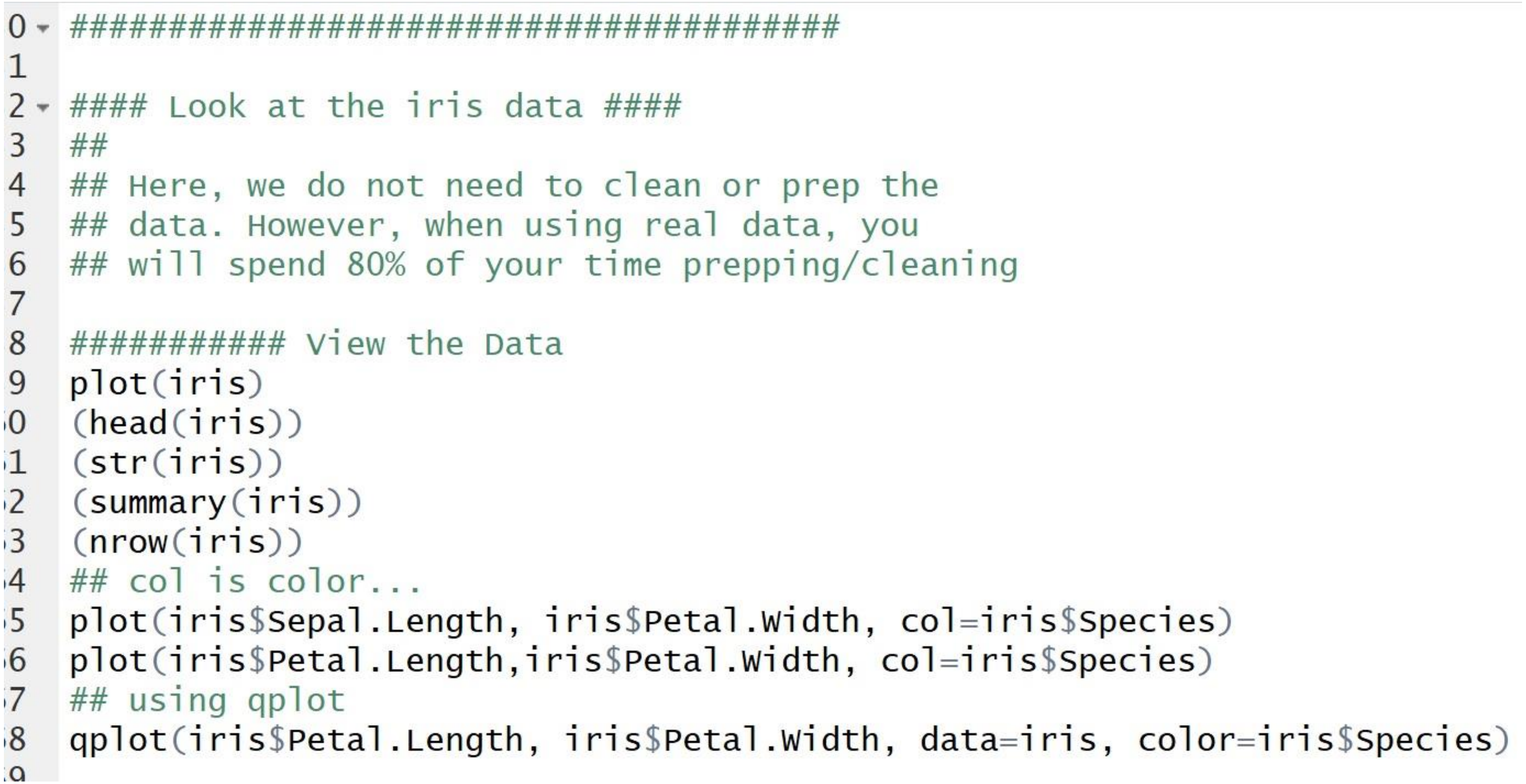
Nice math property: a simple convex optimization problem which is guaranteed to converge to a single global solution

SVMS IN R

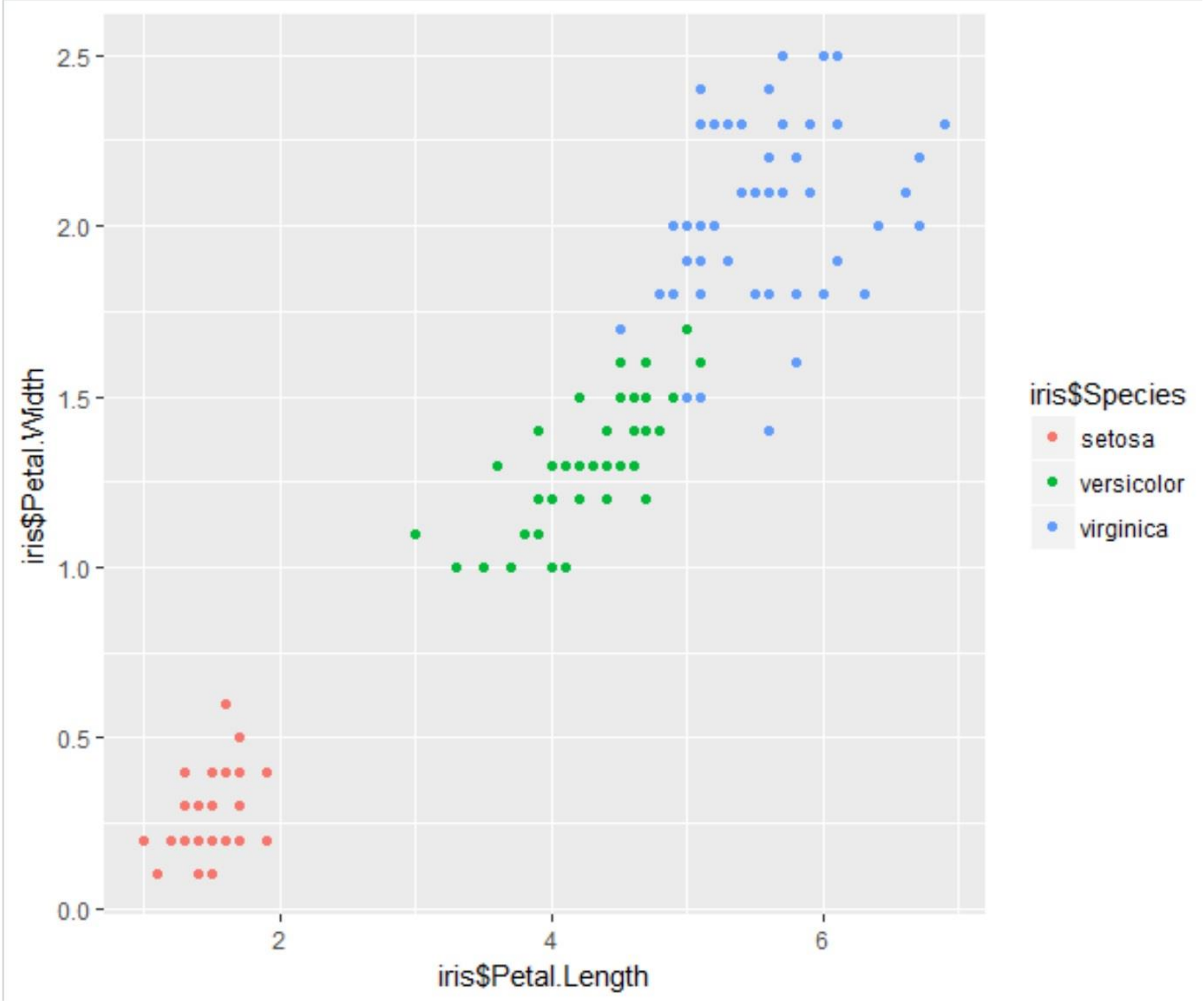
Gates



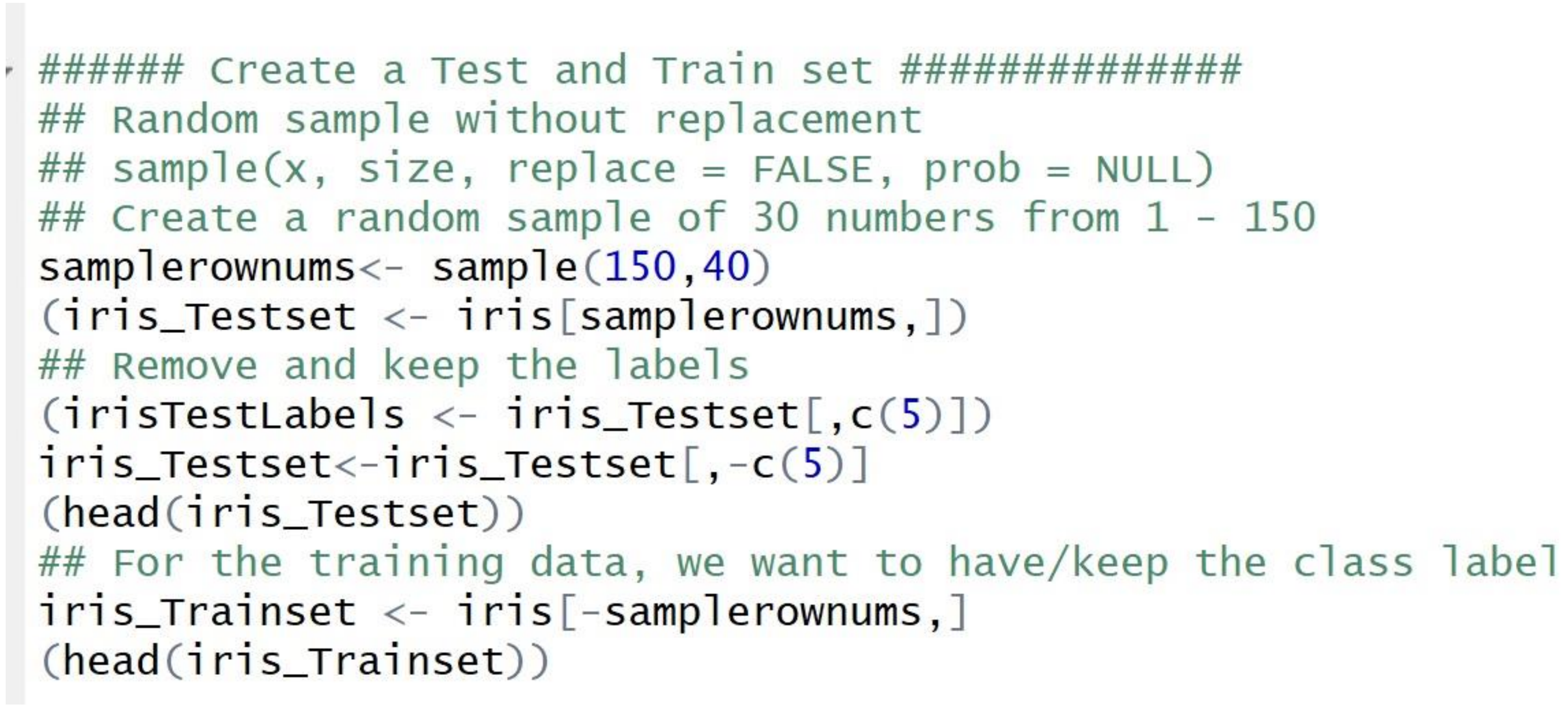
```r
#########################
## libraries
## NOTE: Always install.packages("ThePackName") if needed
## for each library included.
#install.packages("e1071")
library(e1071)  # for machine learning methods
#install.packages("mlr")
library(mlr)
# install.packages("caret")
library(caret)
#install.packages("naivebayes")
##library(naivebayes)
library(datasets)
library(ggplot2)
library(MASS)
```



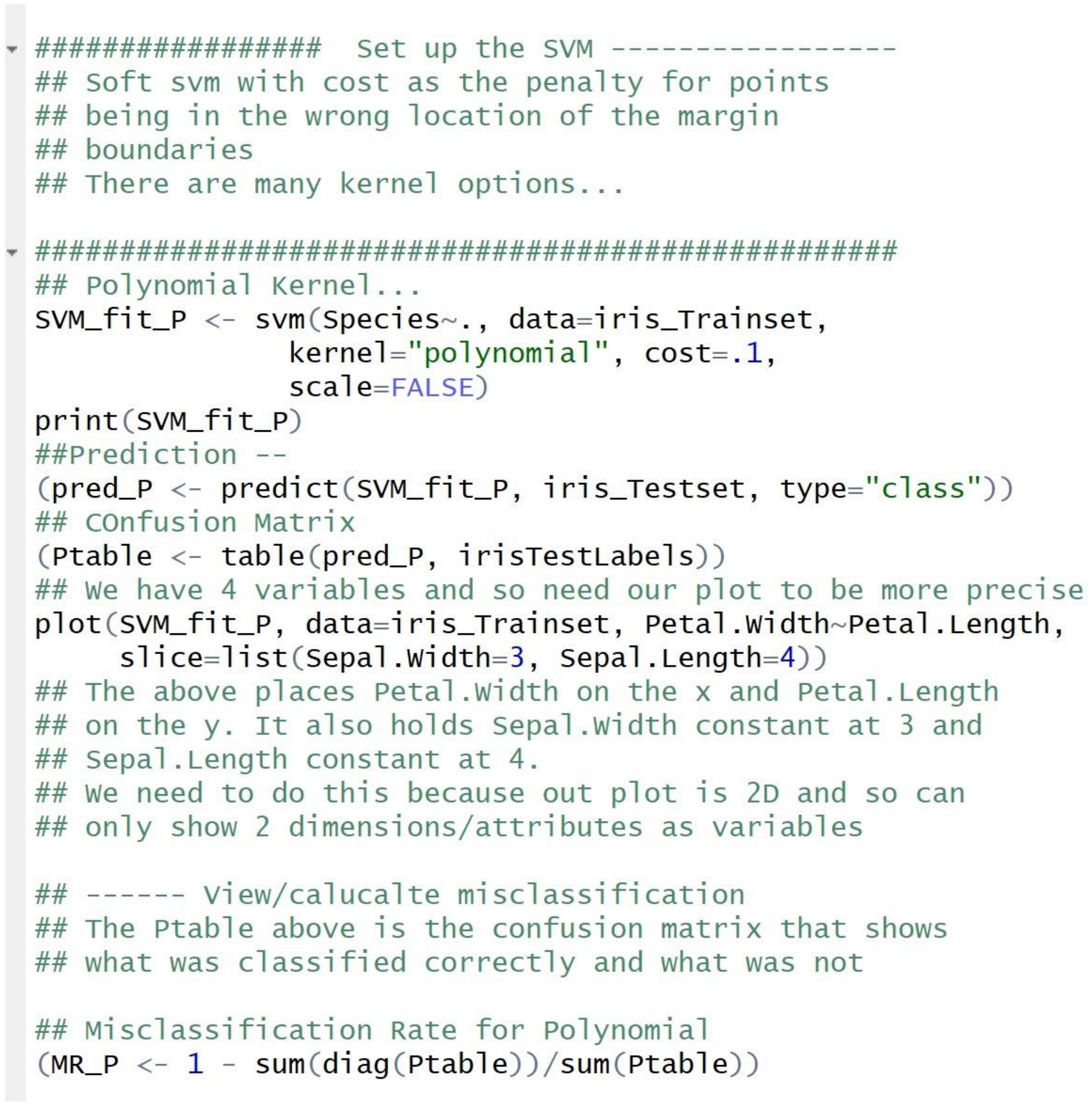
```r
########################################

#### Look at the iris data ####
##
## Here, we do not need to clean or prep the
## data. However, when using real data, you
## will spend 80% of your time prepping/cleaning

########### View the Data
plot(iris)
(head(iris))
(str(iris))
(summary(iris))
(nrow(iris))
## col is color...
plot(iris$Sepal.Length, iris$Petal.Width, col=iris$Species)
plot(iris$Petal.Length,iris$Petal.Width, col=iris$Species)
## using qplot
qplot(iris$Petal.Length, iris$Petal.Width, data=iris, color=iris$Species)
```



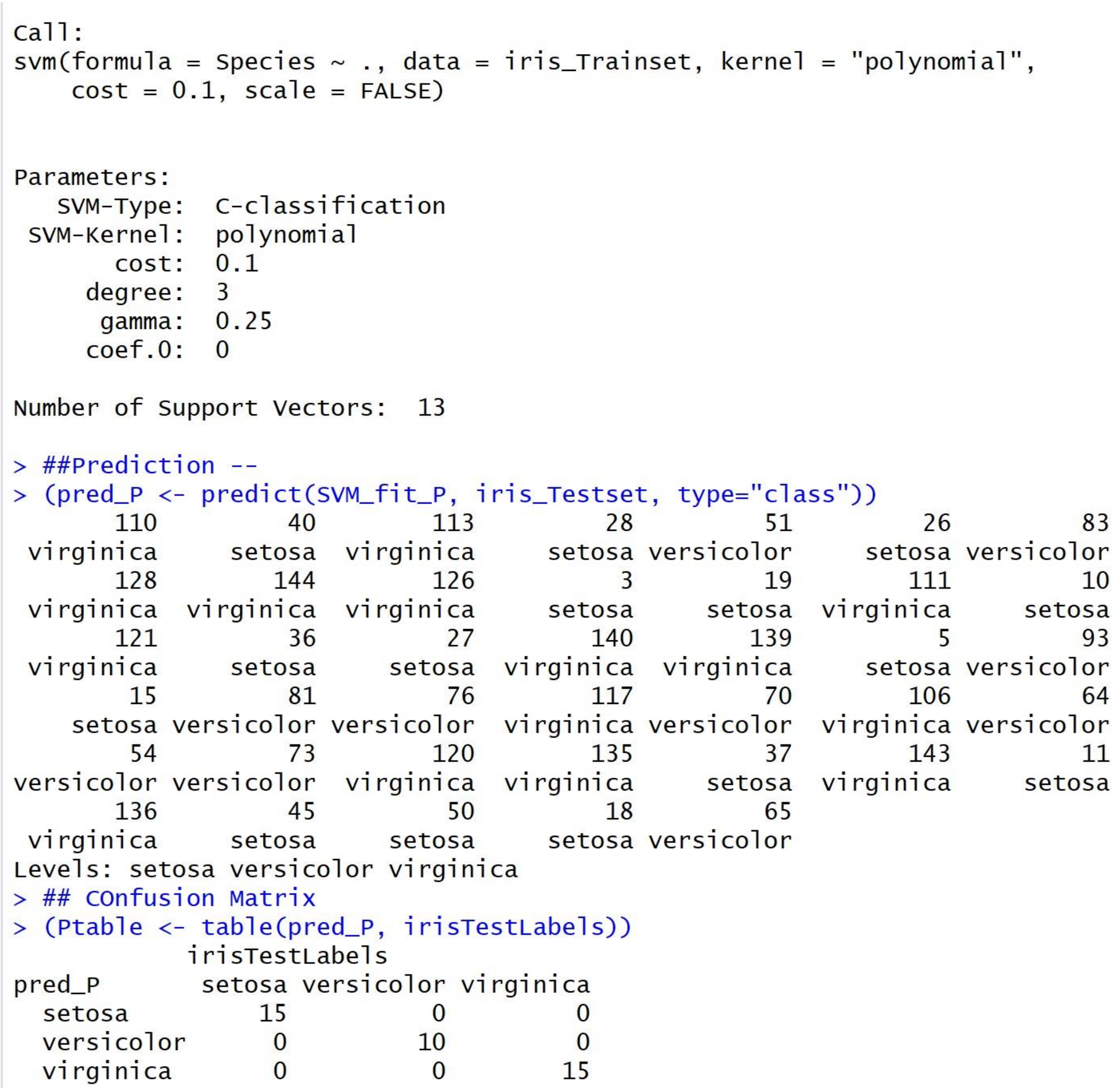
iris$Petal.Width
2.5
2.0
1.5
1.0
0.5
0.0
2
4
6
iris$Petal.Length
iris$Species
setosa
versicolor
virginica



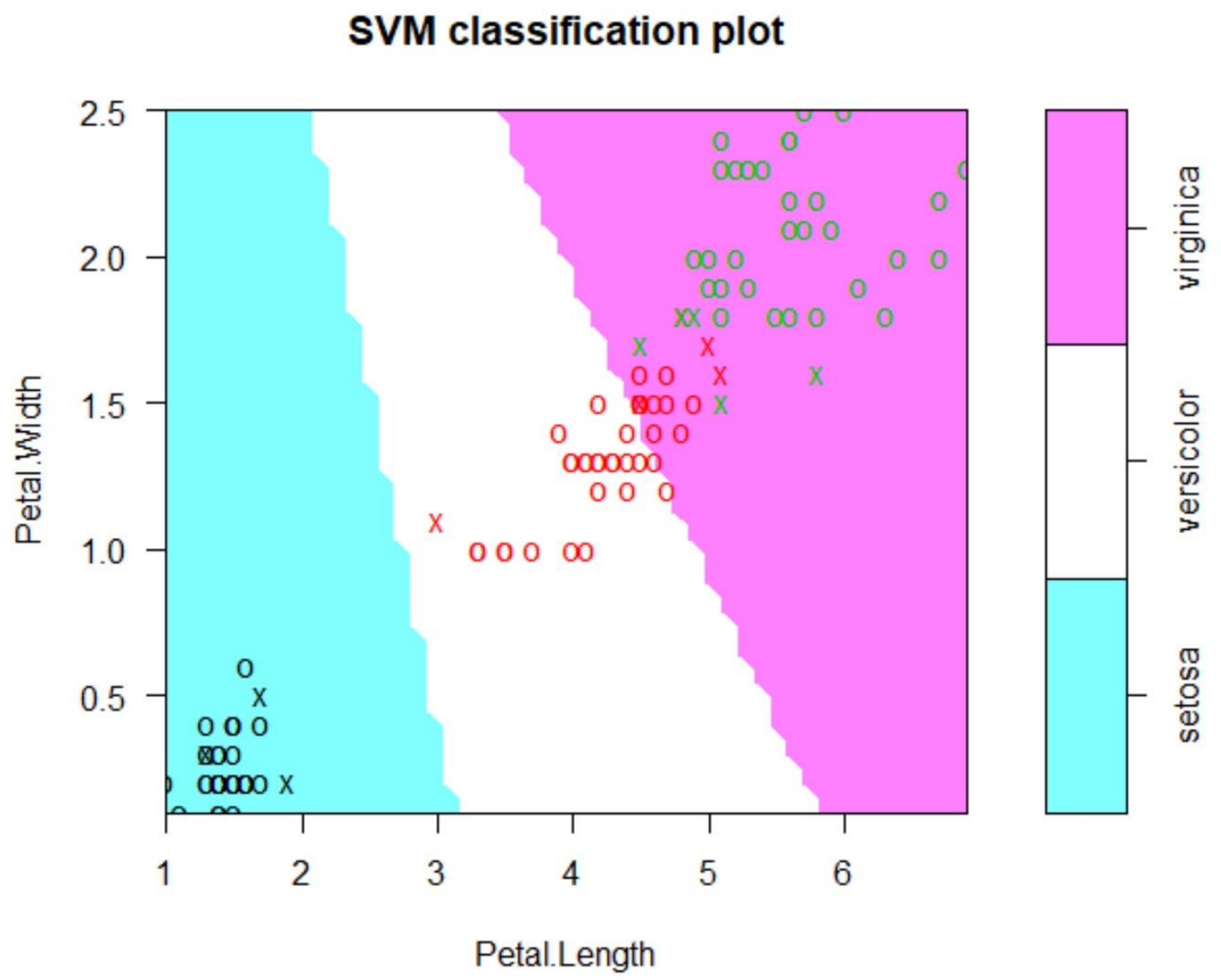
```r
###### Create a Test and Train set ##############
## Random sample without replacement
## sample(x, size, replace = FALSE, prob = NULL)
## Create a random sample of 30 numbers from 1 - 150
samplerownums<- sample(150,40)
(iris_Testset <- iris[samplerownums,])
## Remove and keep the labels
(irisTestLabels <- iris_Testset[,c(5)])
iris_Testset<-iris_Testset[,-c(5)]
(head(iris_Testset))
## For the training data, we want to have/keep the class label
iris_Trainset <- iris[-samplerownums,]
(head(iris_Trainset))
```

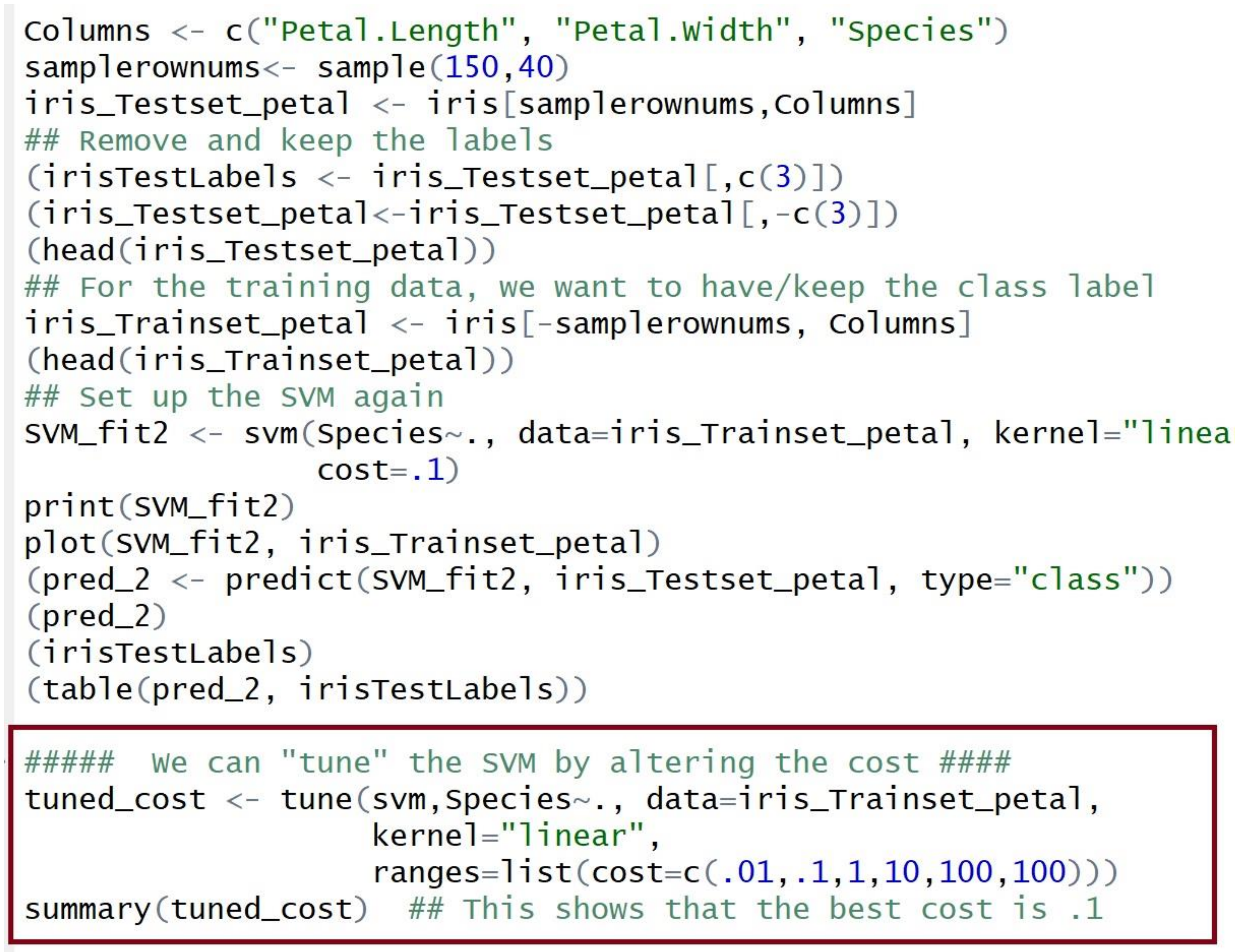


```
################  Set up the SVM -----------------
## Soft svm with cost as the penalty for points
## being in the wrong location of the margin
## boundaries
## There are many kernel options...

###################################################
## Polynomial Kernel...
SVM_fit_P <- svm(Species~., data=iris_Trainset,
               kernel="polynomial", cost=.1,
               scale=FALSE)
print(SVM_fit_P)
##Prediction --
(pred_P <- predict(SVM_fit_P, iris_Testset, type="class"))
## COnfusion Matrix
(Ptable <- table(pred_P, irisTestLabels))
## We have 4 variables and so need our plot to be more precise
plot(SVM_fit_P, data=iris_Trainset, Petal.Width~Petal.Length,
     slice=list(Sepal.Width=3, Sepal.Length=4))
## The above places Petal.Width on the x and Petal.Length
## on the y. It also holds Sepal.Width constant at 3 and
## Sepal.Length constant at 4.
## We need to do this because out plot is 2D and so can
## only show 2 dimensions/attributes as variables

## ------ View/calucalte misclassification
## The Ptable above is the confusion matrix that shows
## what was classified correctly and what was not

## Misclassification Rate for Polynomial
(MR_P <- 1 - sum(diag(Ptable))/sum(Ptable))
```



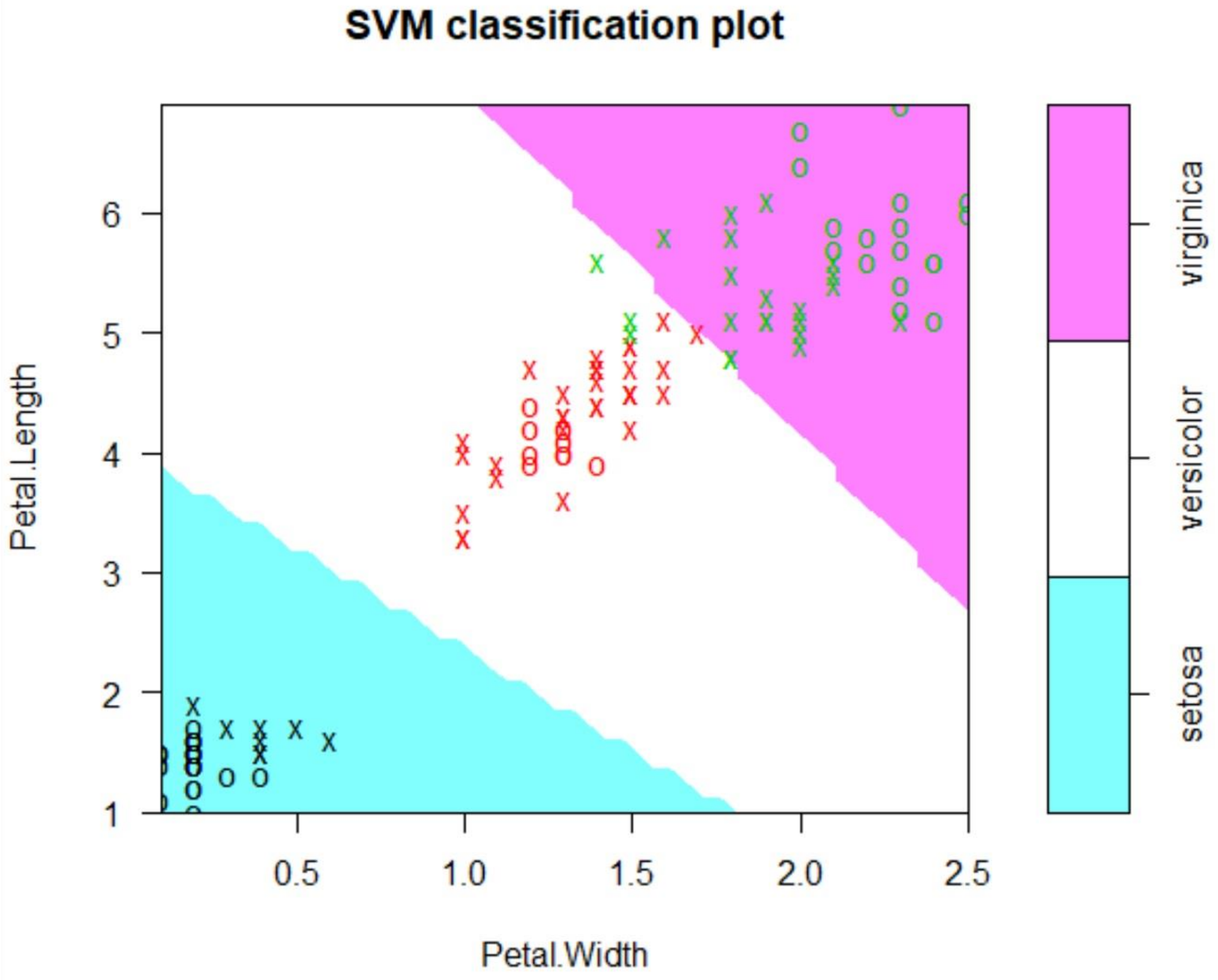
```
Call:
svm(formula = Species ~ ., data = iris_Trainset, kernel = "polynomial",
    cost = 0.1, scale = FALSE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  polynomial
       cost:  0.1
     degree:  3
      gamma:  0.25
     coef.0:  0

Number of Support Vectors:  13

> ##Prediction --
> (pred_P <- predict(SVM_fit_P, iris_Testset, type="class"))
       110         40        113         28         51         26         83
 virginica     setosa  virginica     setosa versicolor     setosa versicolor
       128        144        126          3         19        111         10
 virginica  virginica  virginica     setosa     setosa  virginica     setosa
       121         36         27        140        139          5         93
 virginica     setosa     setosa  virginica  virginica     setosa versicolor
        15         81         76        117         70        106         64
    setosa versicolor versicolor  virginica versicolor  virginica versicolor
        54         73        120        135         37        143         11
versicolor versicolor  virginica  virginica     setosa  virginica     setosa
       136         45         50         18         65
 virginica     setosa     setosa     setosa versicolor
Levels: setosa versicolor virginica
> ## COnfusion Matrix
> (Ptable <- table(pred_P, irisTestLabels))
            irisTestLabels
pred_P       setosa versicolor virginica
  setosa         15          0         0
  versicolor      0         10         0
  virginica       0          0        15
```


SVM classification plot
Petal.Width
Petal.Length
2.5
2.0
1.5
1.0
0.5
1
2
3
4
5
6
virginica
versicolor
setosa



```
Columns <- c("Petal.Length", "Petal.Width", "Species")
samplerownums<- sample(150,40)
iris_Testset_petal <- iris[samplerownums,Columns]
## Remove and keep the labels
(irisTestLabels <- iris_Testset_petal[,c(3)])
(iris_Testset_petal<-iris_Testset_petal[,-c(3)])
(head(iris_Testset_petal))
## For the training data, we want to have/keep the class label
iris_Trainset_petal <- iris[-samplerownums, Columns]
(head(iris_Trainset_petal))
## Set up the SVM again
SVM_fit2 <- svm(Species~., data=iris_Trainset_petal, kernel="linear",
                cost=.1)
print(SVM_fit2)
plot(SVM_fit2, iris_Trainset_petal)
(pred_2 <- predict(SVM_fit2, iris_Testset_petal, type="class"))
(pred_2)
(irisTestLabels)
(table(pred_2, irisTestLabels))
```

```
#####  We can "tune" the SVM by altering the cost ####
tuned_cost <- tune(svm,Species~., data=iris_Trainset_petal,
                   kernel="linear",
                   ranges=list(cost=c(.01,.1,1,10,100,100)))
summary(tuned_cost)  ## This shows that the best cost is .1
```


SVM classification plot
Petal.Length
Petal.Width
1
2
3
4
5
6
0.5
1.0
1.5
2.0
2.5
virginica
versicolor
setosa

IMPORTANT NOTES:

1) When you create an SVM:

(a) Always create a separate Training and Testing set. NEVER test and train on the same set ☺

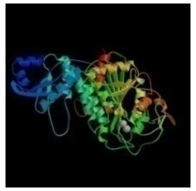
(b) In R, the class or label must be a **factor**.

(c) To visualize (plot) the SVM, you may need to reduce the dimensions. There are various ways to do this.

(d) Always test MANY kernels and values for C. You can tune C is you wish.

SVM MATH

Just FYI



THE SVM

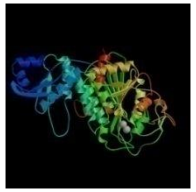
Linear two-class classification:
$y(x) = w^T\phi(x)+b$, where $\phi(x)$ is a fixed feature **space transformation.**

Training data: **$N=\{x_1, \ldots, x_n\}$**

Corresponding class or target values: **$T=\{t_1, \ldots, t_n\}$, $t_i \in \{-1,+1\}$**

Assume that the dataset is linearly separable in the feature space.
Therefore, on w and b there exists
$t_n y(x_n) > 0$ for all x_i

Maximize the margin: The distance between the decision boundary and any of the data points. (**The optimal hyperplane has the max margin)**



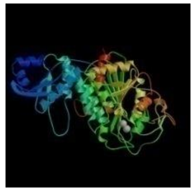
THE SVM MATH - FYI

The **perpendicular distance** from a point to hyperplane **$y(x)=0$**, where **$y(x) = w^T\phi(x)+b$**, is defined as **$t_n(w^T \phi(x_n)+b) / ||w||$**, since only correctly predicted points are of interest.

The **margin** is given by the perpendicular distance to the closest point x_n, and **w** and **b** can be **optimized to maximize** the distance.

The max margin solution can be found by solving:
arg max w, b $\{ 1/||w|| \ \min n\ [t_n(w^T\phi(x_n)+b)]\}$

Note: rescaling w and b by the same constant does not affect the distance between x_n and the decision boundary.
Therefore, set **$t_n(w^T\phi(x_n)+b) = 1$ for the closest point and**
$t_n(w^T\phi(x_n)+b) \geq 1$ for all other points.



THE SVM MATH - FYI

$t_n(w^T\phi(x_n)+b) \geq 1$ is the canonical representation of the hyperplane.

To maximize the margin, **minimize w^Tw**

Therefore, the **Quadratic Programming** problem can be stated as:

arg min w, b : w^Tw

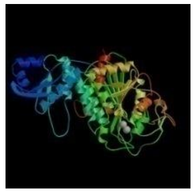
such that **$t_n(w^T\phi(x_n)+b) \geq 1$**

Lagrangian to solve:

$$L(w,b,a)=w^Tw \; - \; \Sigma_{n=1 \text{ to } N} \; a_n\{t_n(w^T\phi(x_n)+b) - 1\}$$

Take derivatives of L(w,b,a) w.r.t. w and b to get:

$$w = \Sigma_{n=1 \text{ to } N} \; a_n t_n \phi(x_n) \quad \text{and} \quad \Sigma_{n=1 \text{ to } N} \; a_n t_n = 0$$



THE SVM: THE KERNEL

Substituting:

$w = \Sigma_{n=1 \text{ to } N} \; a_n t_n \phi(x_n)$ **and** $\Sigma_{n=1 \text{ to } N} \; a_n t_n = 0$

into

$L(w,b,a) = w^T w \; - \; \Sigma_{n=1 \text{ to } N} \; a_n\{t_n(w^T\phi(x_n)+b) \; - 1\}$

The Dual Representation:

$L'(a) = \Sigma_{n=1 \text{ to } N} \; a_n - ½ \, \Sigma_{n=1 \text{ to } N} \Sigma_{m=1 \text{ to } N} \, a_n a_m t_n t_m \phi(x_n)^T \, \phi(x_m)$

Kernel:

$K(x_{n,} x_m) = \phi(x_n)^T \, \phi(x_m)$

NOTE: This portion of the math allows for the use of a kernel to represent the inner product of data points.

MODEL EVALUATION METHODS

Gates

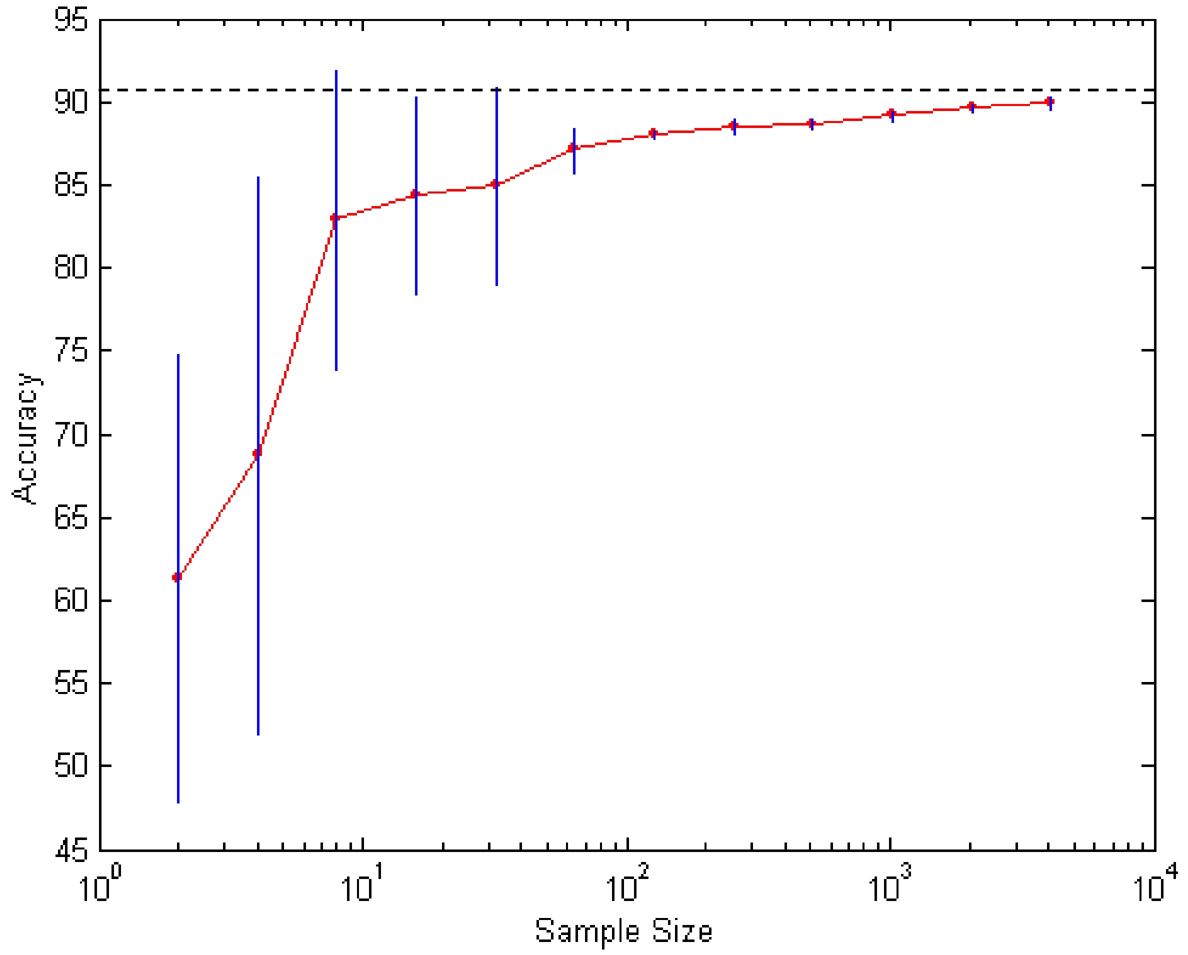
METHODS FOR PERFORMANCE EVALUATION

How to obtain a **reliable estimate of performance**?

Performance of a model may depend on other factors besides the learning algorithm:

- Class distribution
- Cost of misclassification
- Size of training and test sets

LEARNING CURVE



METHODS OF ESTIMATION

Holdout

- Reserve 2/3 for training and 1/3 for testing

Random subsampling

- Repeated holdout

Cross validation

- Partition data into k disjoint subsets
- k-fold: train on k-1 partitions, test on the remaining one
- **Leave-one-out:** k=n

ROC (RECEIVER OPERATING CHARACTERISTIC)

Performance of each classifier represented as a point on the ROC curve

- changing the threshold of algorithm, sample distribution or cost matrix changes the location of the point

Developed in 1950s for signal detection theory to analyze noisy signals

- Characterize the trade-off between **positive hits** and **false alarms**

ROC curve plots TP (on the y-axis) against FP (on the x-axis)

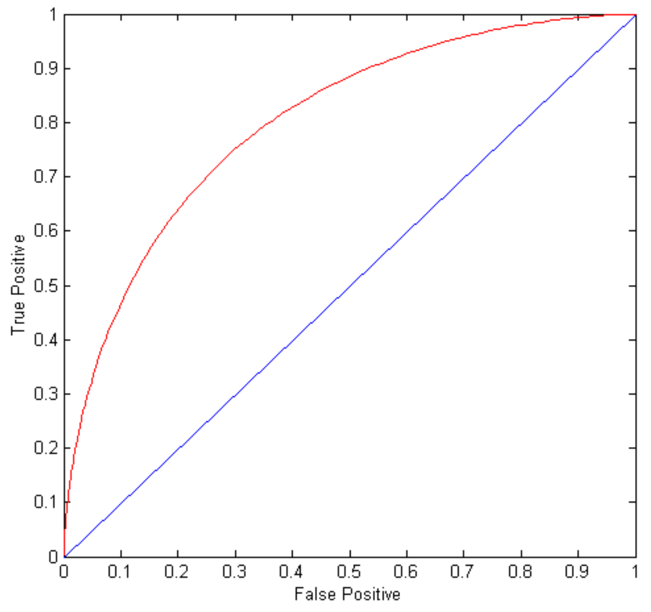
ROC CURVE

(TP,FP):

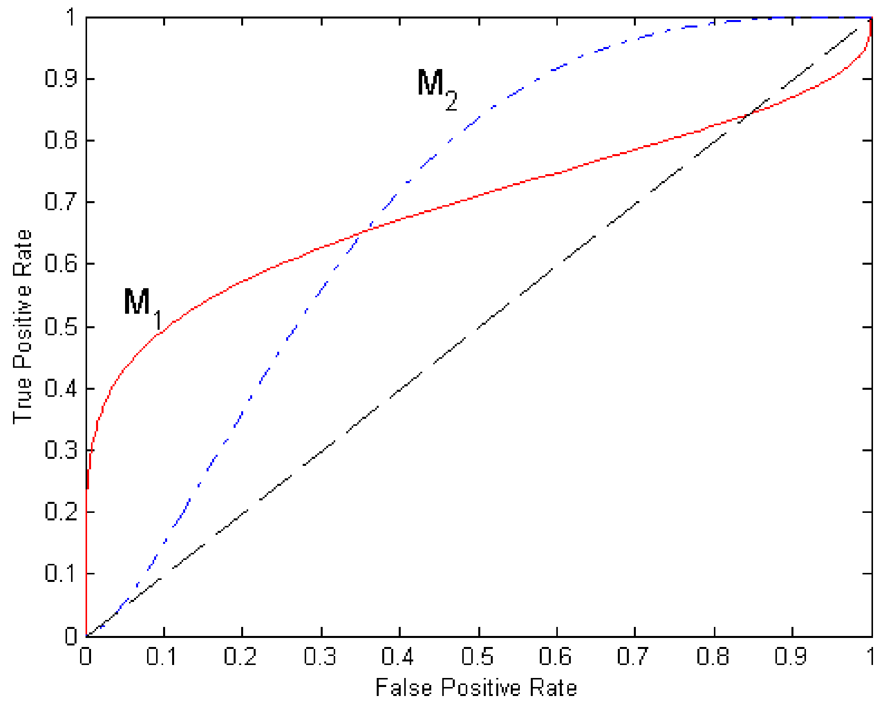
(0,0): declare everything to be negative class

(1,1): declare everything to be positive class

(1,0): ideal



USING ROC FOR MODEL COMPARISON



HOW TO CONSTRUCT AN ROC CURVE

Instance	P(+|A)	True Class
1	0.95	+
2	0.93	+
3	0.87	-
4	0.85	-
5	0.85	-
6	0.85	+
7	0.76	-
8	0.53	+
9	0.43	-
10	0.25	+

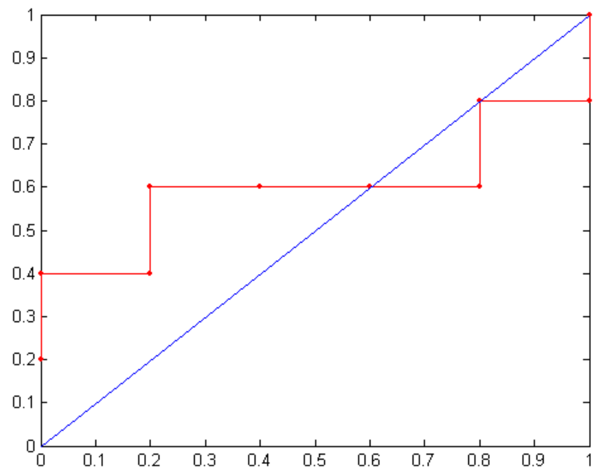
- Use classifier that produces posterior probability for each test instance P(+|A)
- Sort the instances according to P(+|A) in decreasing order
- Apply threshold at each unique value of P(+|A)
- Count the number of TP, FP, TN, FN at each threshold
- TP rate, TPR = TP/(TP+FN)
- FP rate, FPR = FP/(FP + TN)

HOW TO CONSTRUCT AN ROC CURVE

Threshold >=

Class	+	-	+	-	-	-	+	-	+	+	
	0.25	0.43	0.53	0.76	0.85	0.85	0.85	0.87	0.93	0.95	1.00
TP	5	4	4	3	3	3	3	2	2	1	0
FP	5	5	4	4	3	2	1	1	0	0	0
TN	0	0	1	1	2	3	4	4	5	5	5
FN	0	1	1	2	2	2	2	3	3	4	5
TPR	1	0.8	0.8	0.6	0.6	0.6	0.6	0.4	0.4	0.2	0
FPR	1	1	0.8	0.8	0.6	0.4	0.2	0.2	0	0	0

ROC Curve:



COMPARING PERFORMANCE OF 2 MODELS

Given two models, say M1 and M2, which is better?

- M1 is tested on D1 (size=n1), found error rate = e_1
- M2 is tested on D2 (size=n2), found error rate = e_2
- Assume D1 and D2 are independent
- If n1 and n2 are sufficiently large, then
- Approximate:

$$e_1 \sim N(\mu_1, \sigma_1)$$
$$e_2 \sim N(\mu_2, \sigma_2)$$

$$\hat{\sigma}_i = \frac{e_i(1-e_i)}{n_i}$$

COMPARING PERFORMANCE OF 2 MODELS

To test if performance difference is statistically significant: d = e1 – e2

- d ~ $N(d_t,\sigma_t)$ where d_t is the true difference
- Since D1 and D2 are independent, their variance adds up:

$$\sigma_t^2 = \sigma_1^2 + \sigma_2^2 \cong \hat{\sigma}_1^2 + \hat{\sigma}_2^2$$

$$= \frac{e1(1-e1)}{n1} + \frac{e2(1-e2)}{n2}$$

- At (1-α) confidence level,

$$d_t = d \pm Z_{\alpha/2}\hat{\sigma}_t$$

AN ILLUSTRATIVE EXAMPLE

Given: M1: n1 = 30, e1 = 0.15
M2: n2 = 5000, e2 = 0.25

d = |e2 – e1| = 0.1   (2-sided test)

$$\hat{\sigma}_d = \frac{0.15(1-0.15)}{30} + \frac{0.25(1-0.25)}{5000} = 0.0043$$

At 95% confidence level, $Z_{\alpha/2}$=1.96

$$d_t = 0.100 \pm 1.96 \times \sqrt{0.0043} = 0.100 \pm 0.128$$

COMPARING PERFORMANCE OF 2 ALGORITHMS

Each learning algorithm may produce k models:

- L1 may produce M11 , M12, …, M1k
- L2 may produce M21 , M22, …, M2k

If models are generated on the same test sets D1,D2, …, Dk (e.g., via cross-validation)

- For each set: compute $d_j = e_{1j} - e_{2j}$
- d_j has mean d_t and variance σ_t
- Estimate:

$$\hat{\sigma}_t^2 = \frac{\sum_{j=1}^{k}(d_j - \overline{d})^2}{k(k-1)}$$

$$d_t = d \pm t_{1-\alpha,k-1}\hat{\sigma}_t$$