CURSOS PIT

CTIC UNI

Fundamentos de Inteligencia Artificial

CURSOS PIT

CTIC UNI

PhD Wester Edison Zela Moraya

PhD en Computer Science – Inteligencia Artificial por la Universidad Politécnica de Madrid. Master en Ingeniería de Software por la Universidad de Oxford. Master en Análisis Financiero y Económico por la Universidad Complutense de Madrid. Ingeniero de Sistemas de la UNI.

Amplia experiencia profesional en Transformación Digital, Machine Learning, RPAs, Data Science, Metodologías Ágiles, Microservices, gestión económica de proyectos. Docente de Inteligencia Artificial en la Universidad Nacional de Ingeniería.

Director de TI en empresas en Peru y Europa

Consultor de IA y Datos en la SGTD en la PCM

Miembro del AI Connect Program (US Department y Atlantic Council)

Creador de Troomes.com

Temas – Sesion 4

- SVM (Maquinas de Soporte Vectorial - Clasificación)
- SMOreg (SVM - Regresión)

Algoritmos de Aprendizaje Supervisado

Algunos algoritmos en el aprendizaje supervisado:

- Arboles de Decisión
- Random Forest
- Redes Bayesianas
- **Máquinas de Vectores de Soporte (SVM)**
- Red Neuronal Artificial
- Aprendizaje profundo (Deep Learning)

Agenda

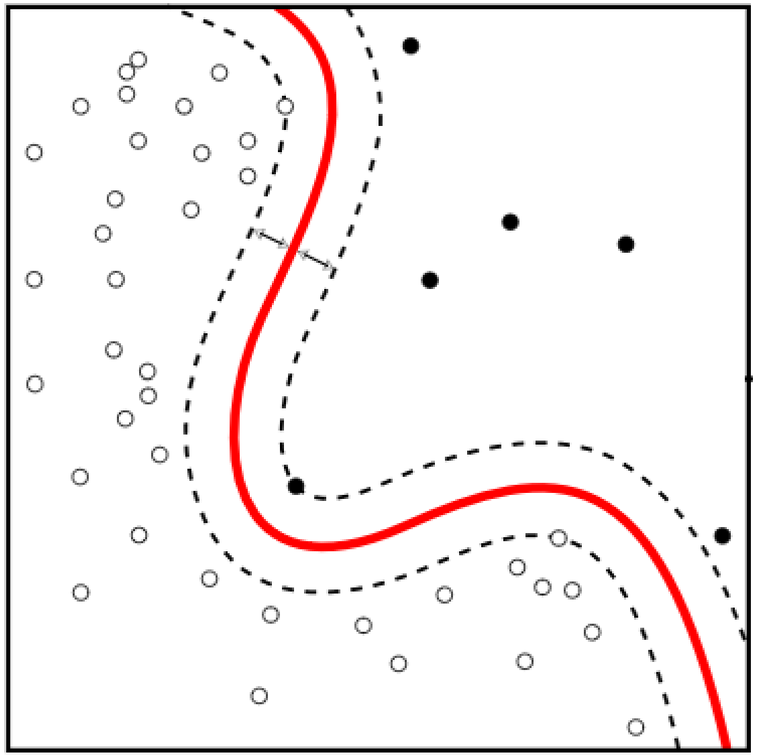
- Support Vector Machines
- Classification and Regression
- SVM in Weka
- SVM in R
- References

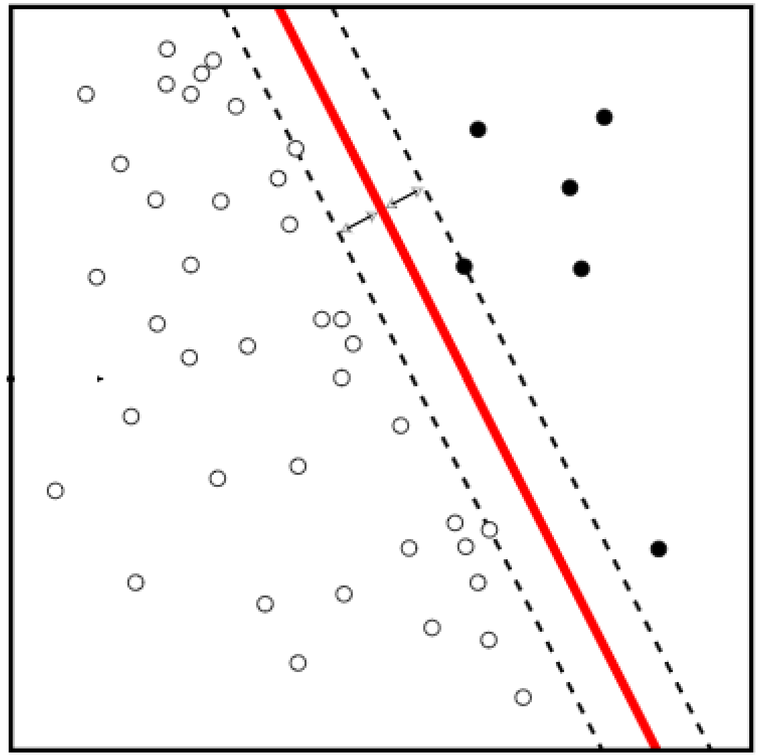
Support Vector Machines

- Support Vector Machines (SVM) are supervised learning models with asociated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis.
- SVM constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks like outliers detection.
- A good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin, the lower the generalization error of the classifier.

Support Vector Machines

- Linear and non-linear classification





Applications of SVM to some problems

- SVMs are helpful in text and hypertext categorization, as their application can significantly reduce the need for labeled training instances in both the standard inductive and transductive settings.
- Hand-written characters can be recognized using SVM.

Applications of SVM to some problems

- Classification of images, experimental results show that SVMs achieve significantly higher search accuracy than traditional query refinement schemes after just three to four rounds of relevance feedback.
- Classification of satellite data like SAR data using supervised SVM.
- The SVM algorithm has been widely applied in the biological and other sciences. They have been used to classify proteins with up to 90% of the compounds classified correctly.

Linear Support Vector Machines

- Given some Data D, a set of n Points of the form:

$$\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in \mathbb{R}^p,\ y_i \in \{-1, 1\}\}_{i=1}^n$$

- Where y_i is -1 or 1, indicating where the class $\mathbf{x}_i$ belongs, each $\mathbf{x}_i$ is a p-dimensional real vector
- We want to find the maximum margin hyperplane that divides the points having y_i = 1 from those having y_i = -1
  - G(X) = W X – b = 1 and G(X) = W X – b = -1

Linear Support Vector Machines

- Geometrically, the distance between these two hyperplane is $\frac{2}{\|\mathbf{w}\|}$, so maxime the distance between the planes we want to minize $\|\mathbf{w}\|$.

- This can be written as $\mathbf{w} \cdot \mathbf{x}_i - b \geq 1$ for $\mathbf{x}_i$ of the first class
- This can be optimization problem $\mathbf{w} \cdot \mathbf{x}_i - b \leq -1$ for $\mathbf{x}_i$ of the second.

$$y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1, \quad \text{for all } 1 \leq i \leq n.$$

Minimize (in $\mathbf{w}, b$)

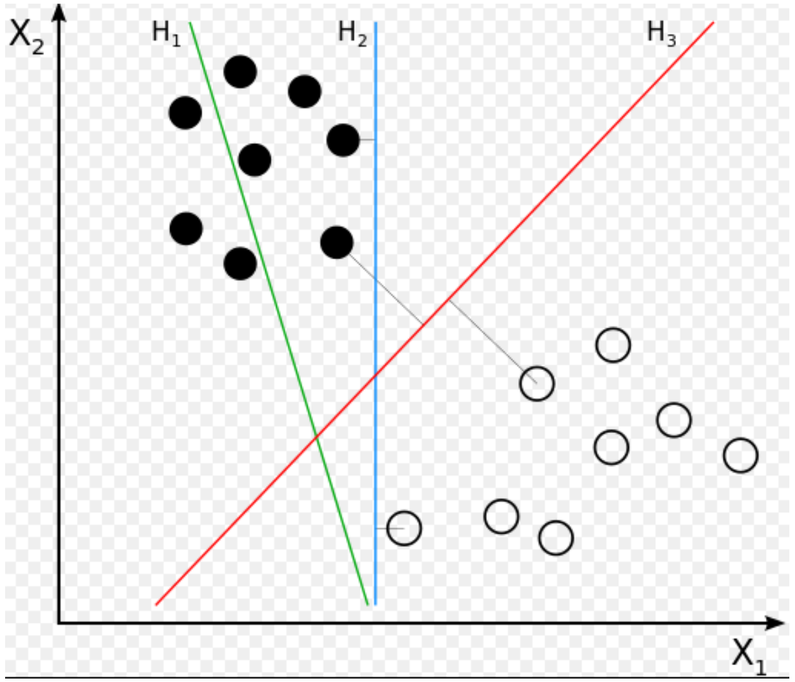
$$\|\mathbf{w}\|$$

subject to (for any $i = 1, \ldots, n$)

$$y_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1.$$

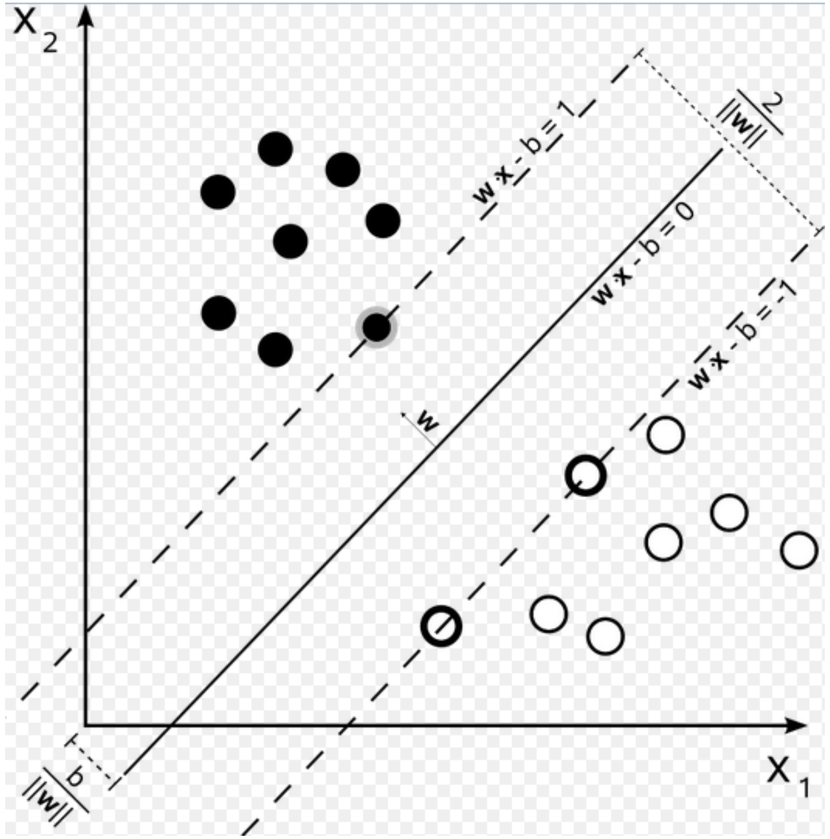
Linear Support Vector Machines

- H_1 does not separate the classes.
- H_2 does, but only with a small margin.
- H_3 separates them with the maximum margin.



Linear Support Vector Machines

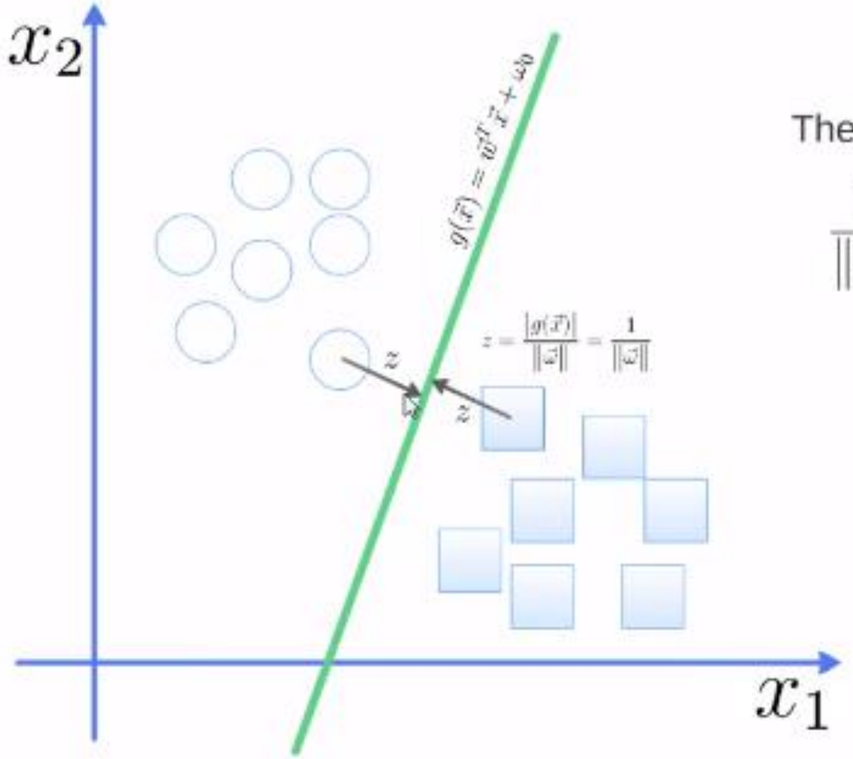
- Maximum-margin hyperplane and margins for an SVM trained with samples from two classes. Samples on the margin are called the support vectors.



Linear Support Vector Machines

- $g(x) = w\,x + w_0$
- $Z = |g(x)| \,/\, ||w||$

$$g(\vec{x}) \geqslant 1, \quad \forall \vec{x} \in \text{class } 1$$
$$g(\vec{x}) \leqslant -1, \quad \forall \vec{x} \in \text{class } 2$$



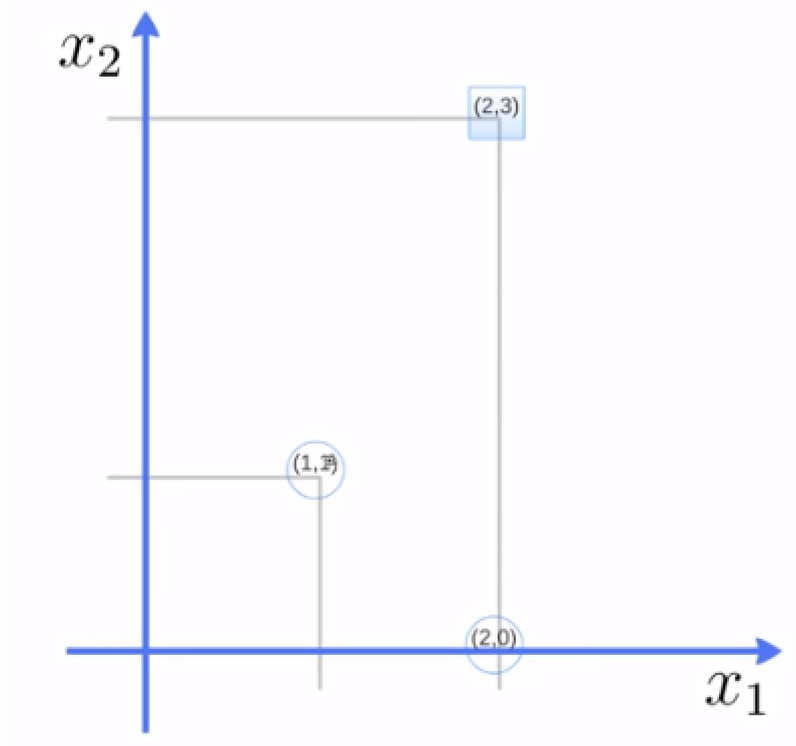
The total margin is computed by

$$\frac{1}{\|\vec{\omega}\|} + \frac{1}{\|\vec{\omega}\|} = \frac{2}{\|\vec{\omega}\|}$$

Minimizing this term will maximize the separability

Linear Support Vector Machines

- Example:
  - Three points:
    - (1,1)
    - (2,3)
    - (2,0)

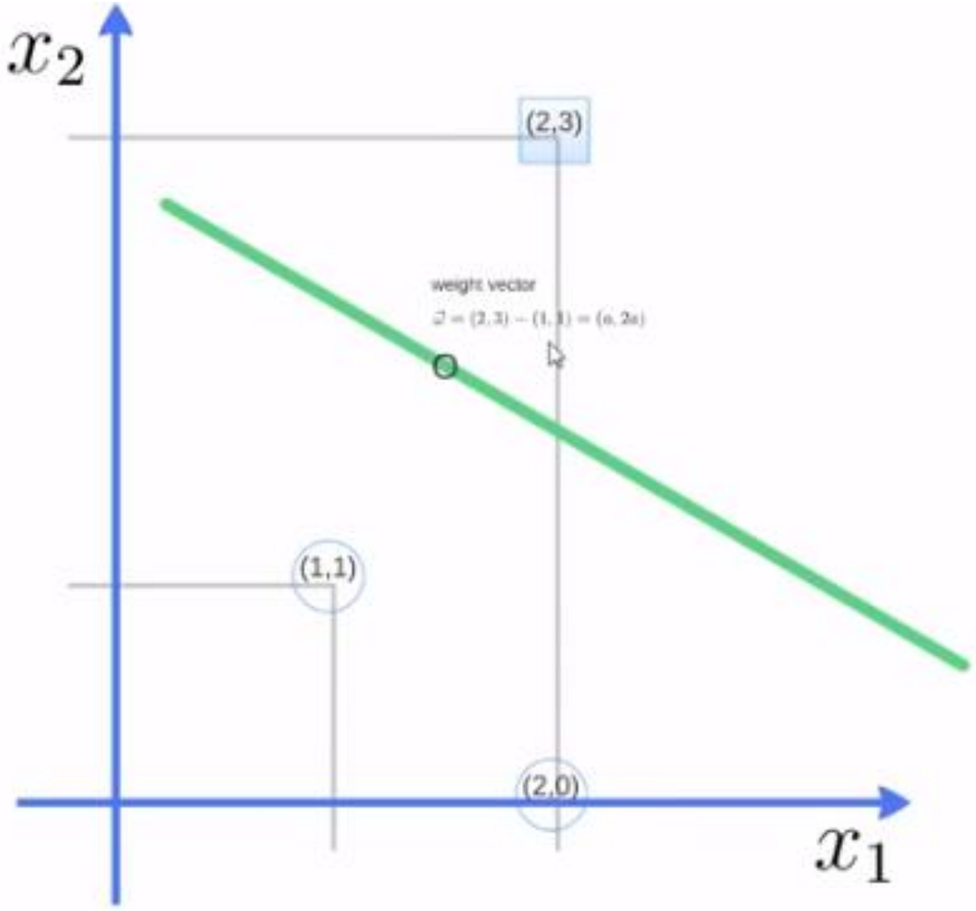


Linear Support Vector Machines

- To Calculate vector W difference

of two vectors:

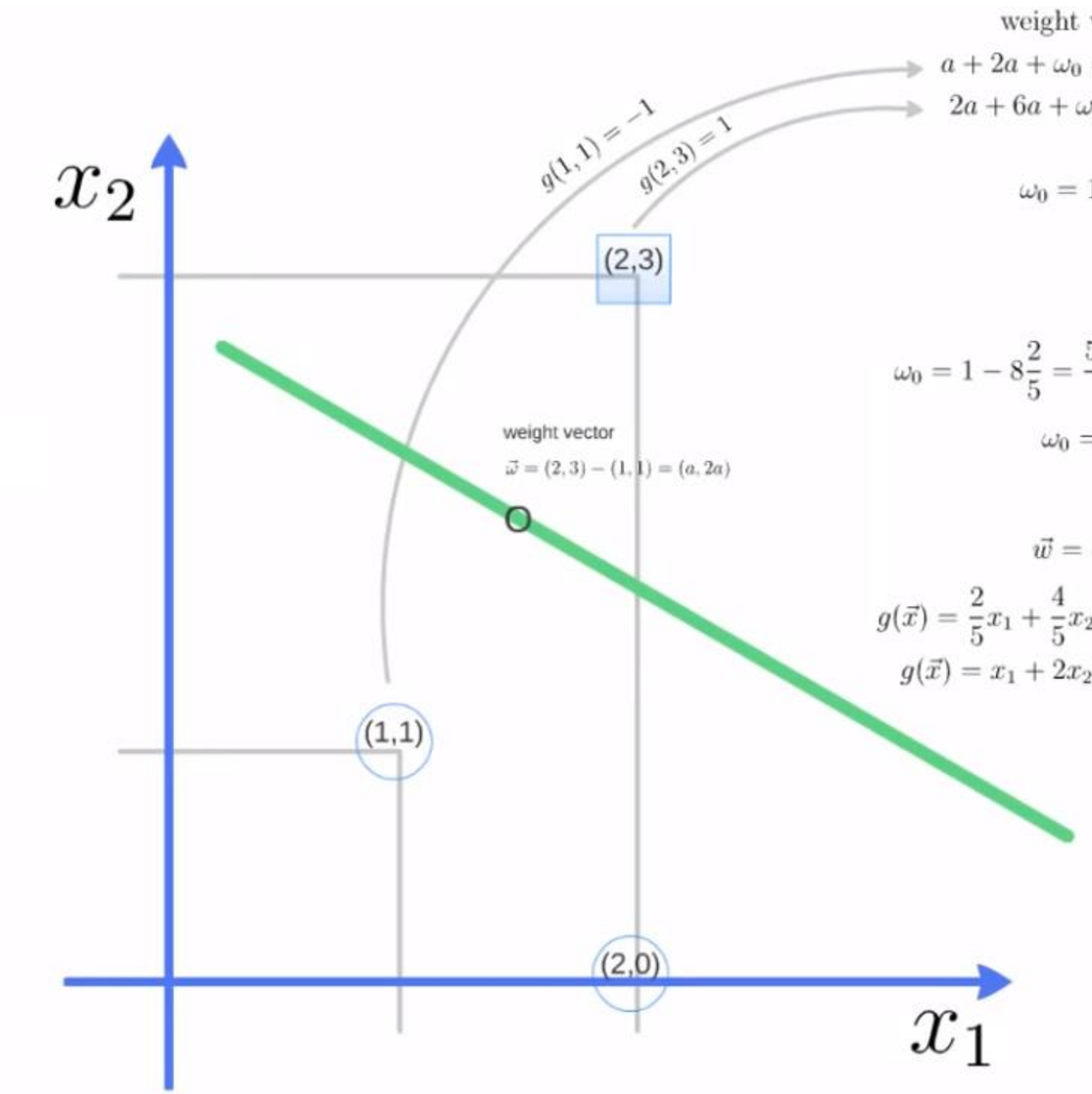
- W = (2,3) – (1,1) = (a,2a)



Linear Support Vector Machines

- Support Vector:

w = (2/5, 4/5)



weight vector $\vec{w} = (a, 2a)$

$a + 2a + \omega_0 = -1,$ using point (1,1)

$2a + 6a + \omega_0 = 1,$ using point (2,3)

$\dots$

$\omega_0 = 1 - 8a$ $\quad 3a + 1 - 8a = -1$

$\vdots$ $\quad 5a = 2$

$a = \frac{2}{5}$

$\omega_0 = 1 - 8\frac{2}{5} = \frac{5-16}{5}$

$\omega_0 = -\frac{11}{5}$

$\dots$

$\vec{w} = (\frac{2}{5}, \frac{4}{5})$

$g(\vec{x}) = \frac{2}{5}x_1 + \frac{4}{5}x_2 - \frac{11}{5}$

$g(\vec{x}) = x_1 + 2x_2 - 5.5$

Non-Linear Classification

The original maximum-margin hyperplane algorithm was constructed a linear classifier. However, in 1992, was suggested a way to create nonlinear classifiers by applying the kernel trick to maximum-margin hyperplanes. The resulting algorithm is formally similar, except that every dot product is replaced by a nonlinear kernel function. This allows the algorithm to fit the maximum-margin hyperplane in a transformed feature space.

The transformation may be nonlinear and the transformed space high-dimensional; although the classifier is a hyperplane in the transformed feature space, it may be nonlinear in the original input space.

Non-Linear Classification

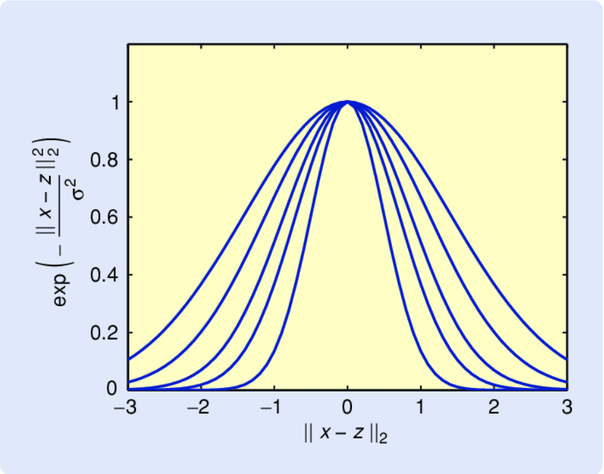
It is noteworthy that working in a higher-dimensional feature space increases the generalization error of support-vector machines, although given enough samples the algorithm still performs well.

Some common kernels include:

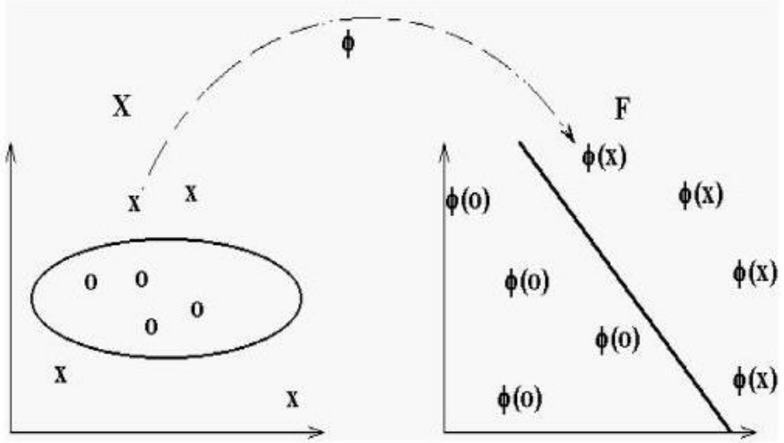
- Polynomial (homogeneous): $k(\overrightarrow{x_i}, \overrightarrow{x_j}) = (\overrightarrow{x_i} \cdot \overrightarrow{x_j})^d$

  for example: $x^5 + 2x^3y^2 + 9x^1y^4$
- Polynomial (inhomogeneous): $k(\overrightarrow{x_i}, \overrightarrow{x_j}) = (\overrightarrow{x_i} \cdot \overrightarrow{x_j} + 1)^d$
- Gaussian radial basis function: $k(\overrightarrow{x_i}, \overrightarrow{x_j}) = \exp(-\gamma \|\overrightarrow{x_i} - \overrightarrow{x_j}\|^2)$ for $\gamma > 0$

  Sometimes parametrized using $\gamma = 1/(2\sigma^2)$
- Hyperbolic tangent: $k(\overrightarrow{x_i}, \overrightarrow{x_j}) = \tanh(\kappa \overrightarrow{x_i} \cdot \overrightarrow{x_j} + c)$ for some (not every) $\kappa > 0$



Non-Linear Classification



The kernel is related to the transform $\varphi(\overrightarrow{x_i})$ by the equation $k(\overrightarrow{x_i}, \overrightarrow{x_j}) = \varphi(\overrightarrow{x_i}) \cdot \varphi(\overrightarrow{x_j})$

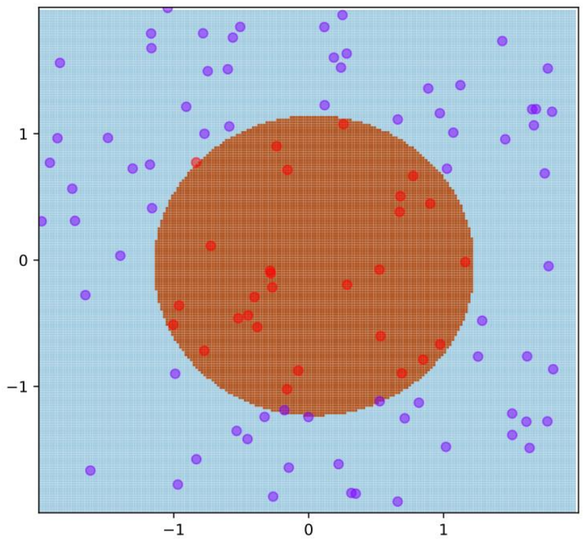
. The value ***w*** is also in the transformed space, with $\vec{w} = \sum_i \alpha_i y_i \varphi(\vec{x}_i)$

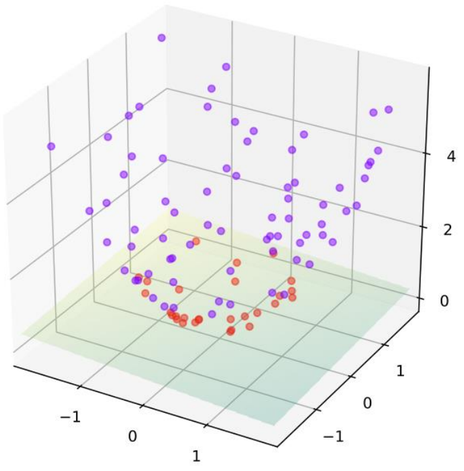
. Dot products with ***w*** for classification can again be computed by the kernel trick, i.e.

$\vec{w} \cdot \varphi(\vec{x}) = \sum_i \alpha_i y_i k(\vec{x}_i, \vec{x})$

Non-Linear Classification

SVM with kernel given by $\phi((a, b)) = (a, b, a^2 + b^2)$ and thus $K(\mathbf{x}, \mathbf{y}) = \mathbf{x} \cdot \mathbf{y} + \|\mathbf{x}\|^2 \|\mathbf{y}\|^2$. The training points are mapped to a 3-dimensional space where a separating hyperplane can be easily found.





SVM in Weka

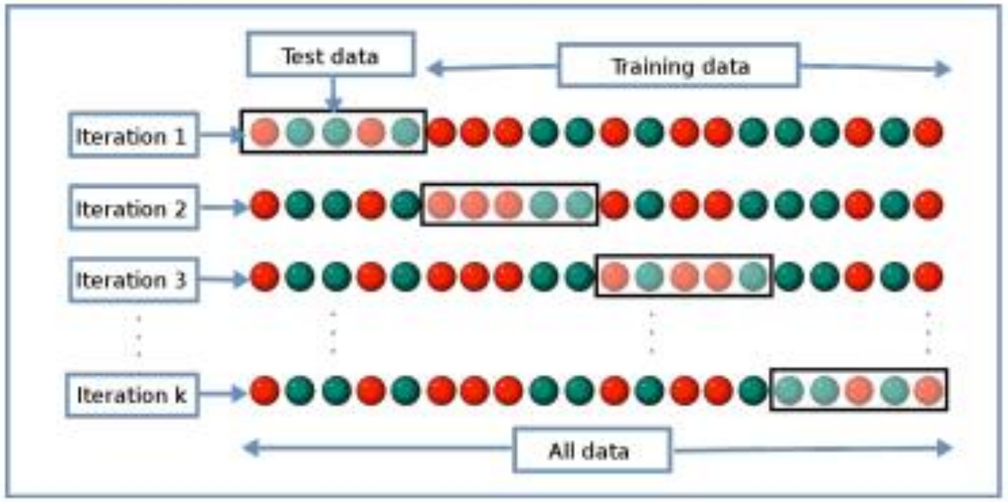
For weka-3-6

- Download library from: https://github.com/cjlin1/libsvm
- Unzip
- Copy libsvm.jar to the directory of weka. i.e /program files/weka-3-6
- Execute weka:
  - Java -classpath “%CLASSPATH%;weka.jar;libsvm.jar” weka.gui.GUIChooser

- For Weka-3-8 download libSVM from:
  http://weka.sourceforge.net/packageMetaData/LibSVM/Latest.html

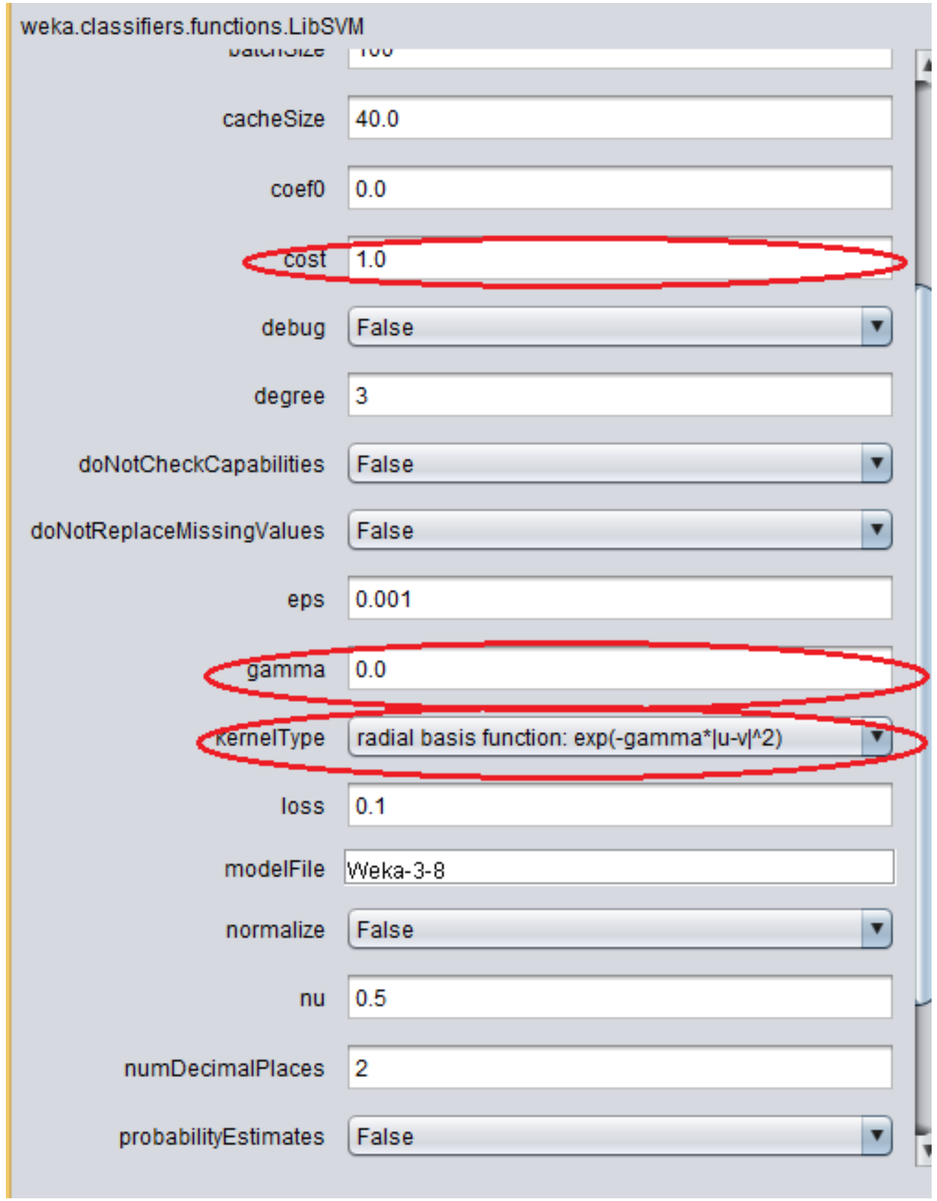
Cross-Validation k Folds

1. Shuffle the dataset randomly.
2. Split the dataset into k groups
3. For each unique group:
   a. Take the group as a hold out or test data set
   b. Take the remaining groups as a training data set
   c. Fit a model on the training set and evaluate it on the test set
   d. Retain the evaluation score and discard the model
4. Summarize the skill of the model using the sample of model evaluation scores



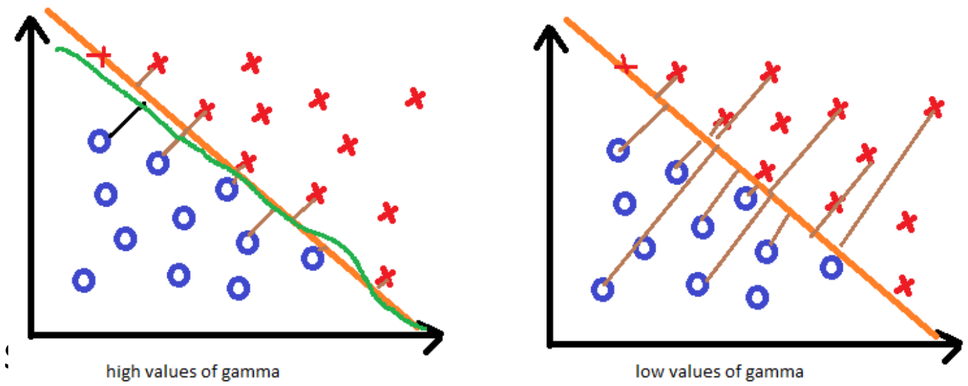
SVM in Weka

- Upload file diabetes.arff
- Select Classify->functions->LibSVM
- In the options of function LibSVM: Kernel Type: Linear
- Select Test Options: Percentage Split 10%
- Run

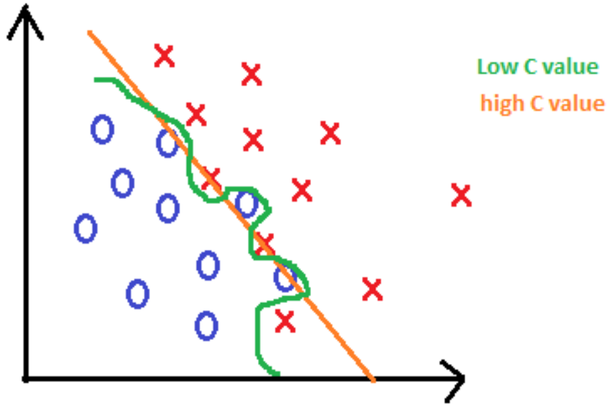


SVM in Weka

- Gamma: Defines how far the influence of the single training examples reaches
  - High value of Gamma – close points
  - Low value of gamma – far points



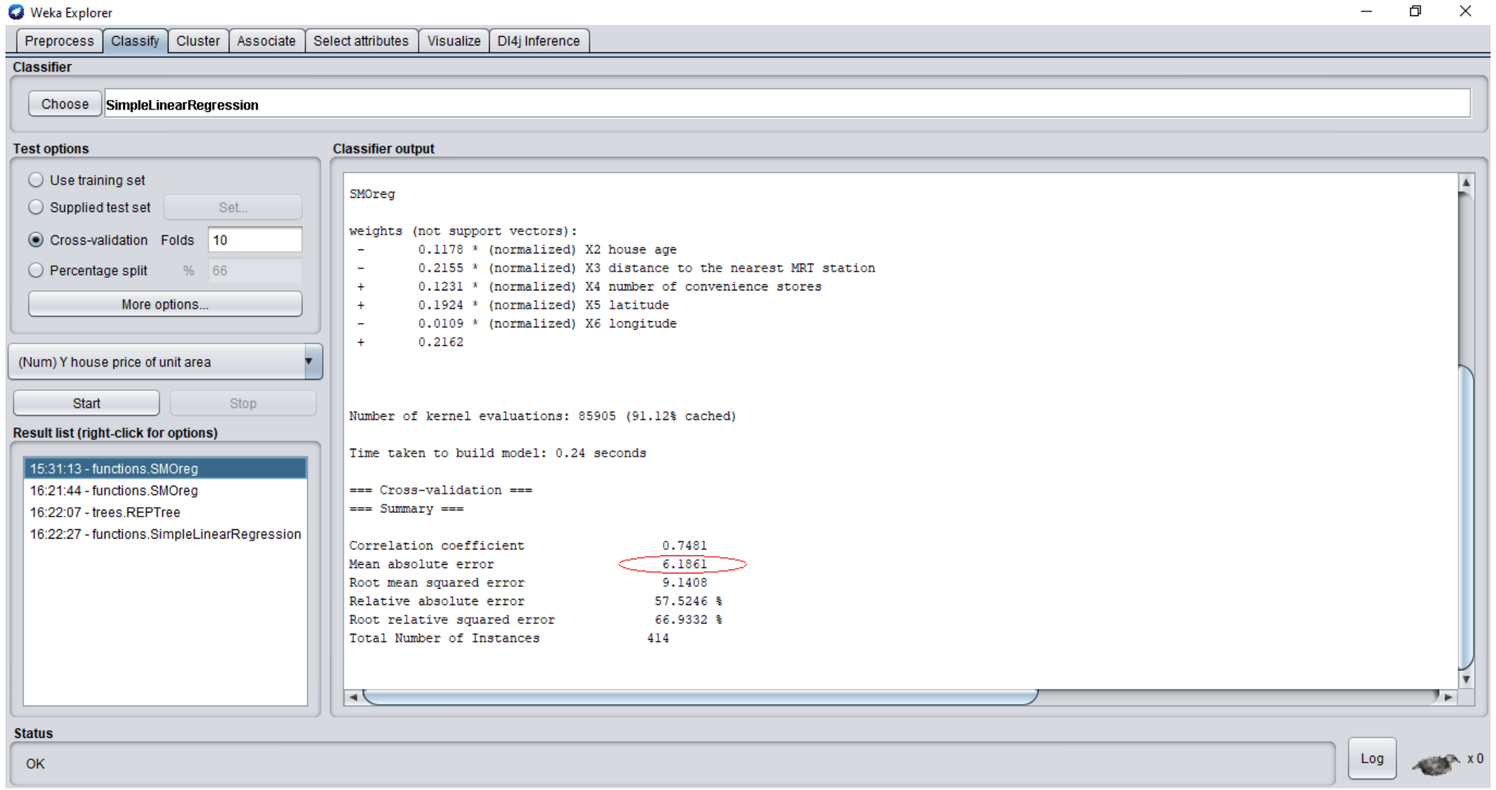
- Cost:Control tradeoff between smooth decision boundary and cla points correctly.
  - Large C will try to find a hyperplane and margin so that there are few very points within the margin. which could mean an overly complex model with a small margin if the points aren't easily separable.
  - A lower C gives higher error on the training set, but finds a larger margin that might be more robus



- kernelType:The type of kernel to use
- https://www.csie.ntu.edu.tw/~cjlin/libsvm/#GUI

SMOreg – Regression in Weka

- File: Real estate valuation data set.csv
- Compare with REPTree and Linear Regression.



SMOreg – Regression in Weka

- Correlation Coeficient:
- Root mean squared error:

SVM in Java - Example

- Open Eclipse
- Create a new java project LibSVMTest
- Create a new class LibSVMTest
- Copy the file LibSVMTest.Java (provided in the course)
- Copy the file vehicledata.arff to the project workspace.
- Run

In R Studio

- Install R
  - https://cran.r-project.org/mirrors.html
  - https://www.rstudio.com/products/rstudio/download/

SVM – R Classification

https://www.rdocumentation.org/packages/e1071/versions/1.7-4/topics/svm

In Rstudio: with the example of iris.

```
library("e1071")
head(iris,5) # print the first records
attach(iris)  # to call the names of the variables directly
x <- subset(iris, select=-Species) # create the matrix x
y <- Species # create the matrix y
plot (Species ~ . , data = iris)   # the tilde ('~') separates the left side of a formula with the right side of the formula.
svm_model <- svm(Species ~ ., data=iris)
summary(svm_model) # summary of model
svm_model1 <- svm(x,y) # create a model using svm
summary(svm_model1)
pred <- predict(svm_model1,x)
system.time(pred <- predict(svm_model1,x))
```

SVM – R Classification

```
# confusion matrix
table(pred,y)
# Tune the model trying different values of cost and gamma.
svm_tune <- tune(svm, train.x=x, train.y=y,kernel="radial", ranges=list(cost=10^(-1:2),
gamma=c(.5,1,2)))
print(svm_tune)
svm_model_after_tune <- svm(Species ~ ., data=iris, kernel="radial", cost=1, gamma=0.5)
summary(svm_model_after_tune)
pred <- predict(svm_model_after_tune,x)
system.time(predict(svm_model_after_tune,x))
table(pred,y)
```

SVM – R Classification - diabetes

```
# Import diabetes.csv (file -> Import Dataset)
library("e1071")
head(diabetes)
train <-diabetes
train[which(train=="?",arr.ind=TRUE)]<-NA
train=unique(train)
y=factor(train[,length(train)])
# convert to numeric. factors are integer fields anyway behind the scenes.
train <- data.frame(lapply(train,as.numeric))
train <- as.matrix(train[-length(train)])
model_svm_dia<-svm(x= train ,y=y,scale=F)
pred_model_svm_dia <- predict(model_svm_dia,train)
table(pred_model_svm_dia,y)
```

SVM R - Regression

In RStudio: example of regression using random data x, y.

```
x=c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20)
y=c(3,4,5,4,8,10,10,11,14,20,23,24,32,34,35,37,42,48,53,60)
#Create a data frame of the data
train=data.frame(x,y)
#Let’s see how our data looks like. For this we use the plot function
#Plot the dataset
plot(train,pch=16)
#Linear regression
model <- lm(y ~ x, train)
#Plot the model using abline
abline(model)
```

SVM R - Regression

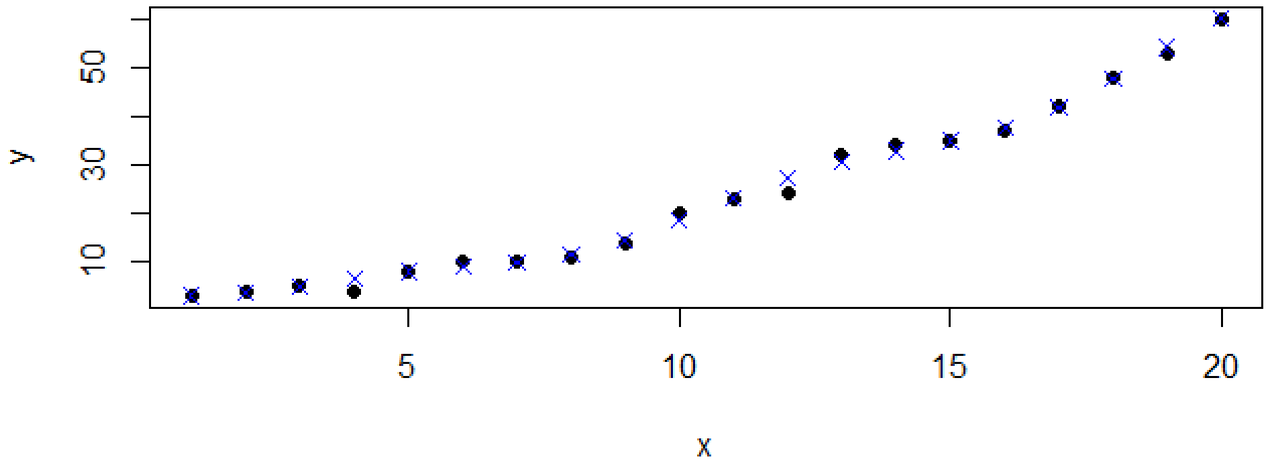
```
# using SVM, install library e1071
library(e1071)
#Fit a model. The function syntax is very similar to lm function
model_svm <- svm(y ~ x , train)
#Use the predictions on the data
pred <- predict(model_svm, train)
#Plot the predictions and the plot to see our model fit
points(train$x, pred, col = "blue", pch=4)
#Linear model has a residuals part which we can extract and directly calculate rmse
error <- model$residuals
lm_error <- sqrt(mean(error^2)) # 3.832974
```

SVM R - Regression

```
#For svm, we have to manually calculate the difference between actual values (train$y) with our predictions (pred)
error_2 <- train$y - pred
svm_error <- sqrt(mean(error_2^2)) # 2.696281
svm_tune <- tune(svm, y ~ x, data = train,  ranges = list(epsilon = seq(0,1,0.01), cost = 2^(2:9)))
print(svm_tune)
#Printing gives the output:
#Parameter tuning of ‘svm’:
# - sampling method: 10-fold cross validation
#- best parameters: # epsilon = 0.09  cost = 256  best performance: 2.569451
#This best performance denotes the MSE. The corresponding RMSE is 1.602951 which is square root of MSE
#The best model
best_mod <- svm_tune$best.model
best_mod_pred <- predict(best_mod, train)
error_best_mod <- train$y - best_mod_pred
```

SVM R - Regression

```
# this value can be different on your computer
# because the tune method randomly shuffles the data
best_mod_RMSE <- sqrt(mean(error_best_mod^2)) # 1.290738
plot(svm_tune)
plot(train,pch=16)
points(train$x, best_mod_pred, col = “red", pch=4)
```



Links where to find Papers

- Where to find technical articles:

  http://citeseerx.ist.psu.edu/index
  https://www.academia.edu/
  http://www.elsevier.com/
  http://ieeexplore.ieee.org/Xplore/home.jsp
  http://dl.acm.org/
  http://www.tdx.cat/
  http://www.capes.gov.br/
  http://search.ieice.org/bin/index.php?c ... g=E&curr=1