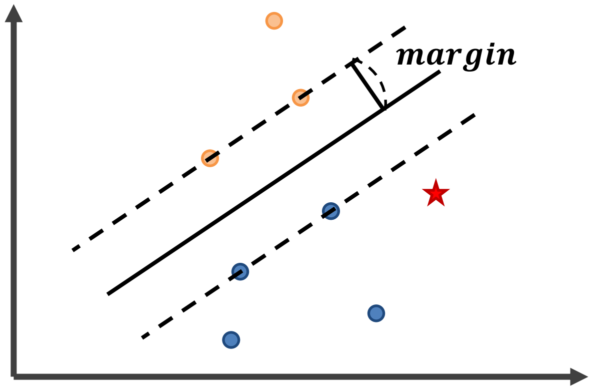
Support Vector Machines

Overview

- ❖ What is support vector machines(SVMs)?
- ❖ Remind: Hyperplane
- ❖ Linear SVMs
- ❖ Soft margin SVMs
- ❖ Non-linear SVMs

What is support vector machines ?

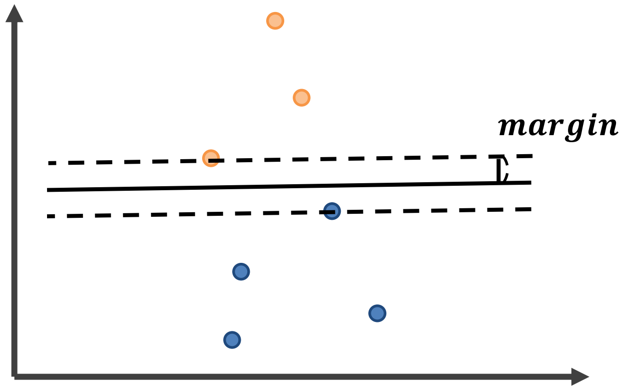
- **Support Vector Machines (SVMs)**
  - Vector space classification (using hyperplane)
  - Large margin classifier
  - Binary classifier (typical)

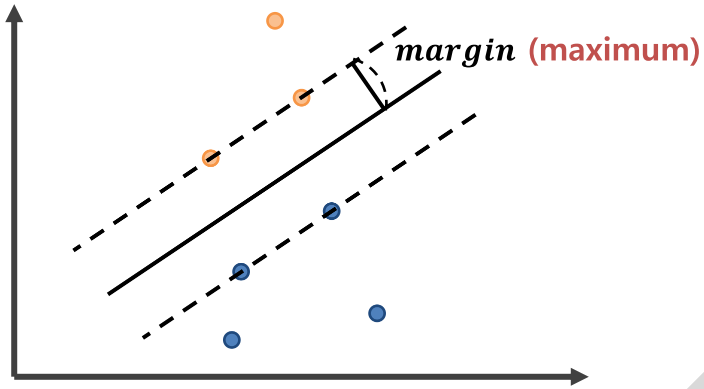


What is support vector machines ?

▪ Strategy of SVMs

1. Calculate hyperplanes that can classify classes
2. Find the hyperplane farthest from any point
3. Classify data based on selected hyperplane

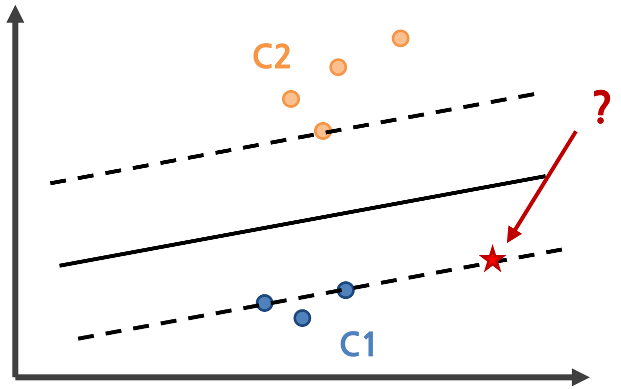




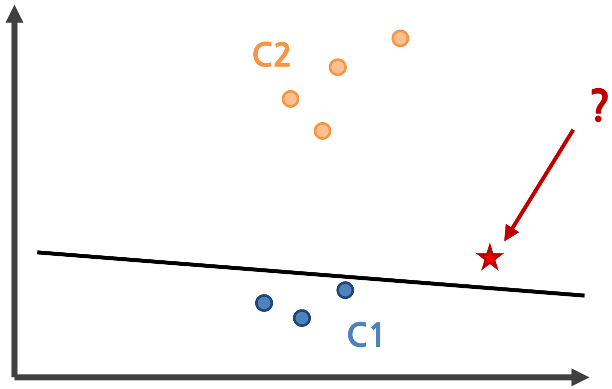
What is support vector machines ?

▪ Why choose a maximum margin

- Enable clear classification
- E.g., maximum margin vs. others



[maximum margin]

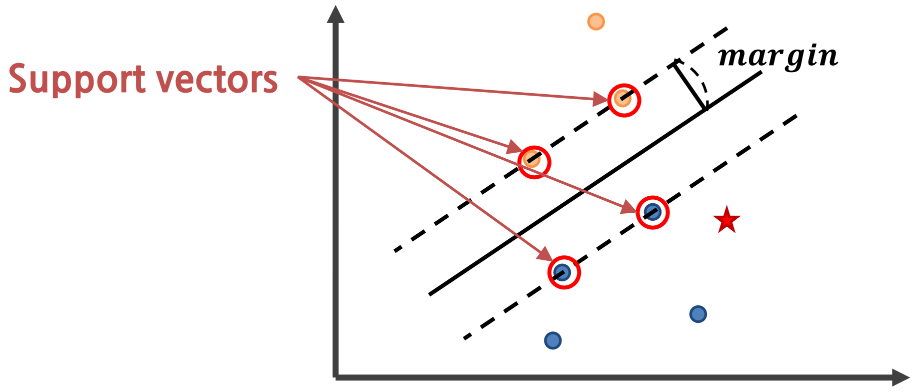


[others]

What is support vector machines ?

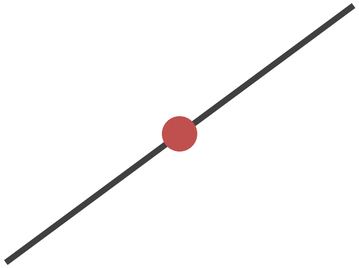
▪ Support vectors

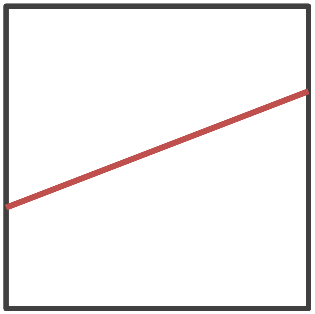
- Vectors that determine the maximum margin
- Vectors on margin lines are called support vectors

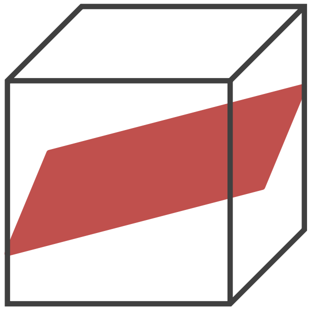


Remind: Hyperplane

- **Hyperplane**
  - An n-dimensional generalization of a plane
  - The hyperplane is an n-dimensional representation of $n-1$ dimensions


[1 dimension]


[2 dimensions]

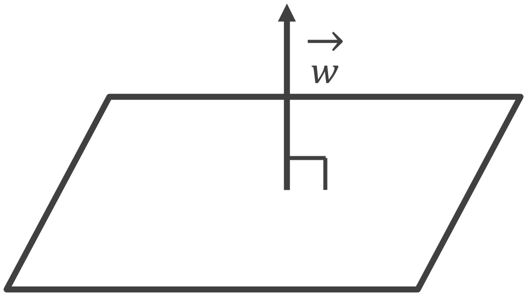

[3 dimensions]

Remind: Hyperplane

How to get

- A hyperplane H in $\mathbb{R}^n$ is the set of points $(x_1, x_2, \dots, x_n)$ that satisfy a linear equation

$$\vec{w}^{\mathrm{T}}\vec{x} + b = 0$$



Remind: Hyperplane

- **What is $\underset{w}{\rightarrow}^{\mathrm{T}}\underset{x}{\rightarrow}$?**

- **Linear equation :** $y = ax + b$

$$y - ax - b = 0$$

$$\underset{w}{\rightarrow}\begin{pmatrix} -b \\ -a \\ 1 \end{pmatrix}, \underset{x}{\rightarrow}\begin{pmatrix} 1 \\ x \\ y \end{pmatrix}$$

$$w^{\mathrm{T}} \cdot x = (-b) * 1 + (-a) * x + 1 * y$$

$$= y - ax - b$$

It's just a different expression !

Linear SVMs

▪ How to choose hyperplane

- First, Margin calculation required

1. Functional margin

   - Calculate margin as the result of the hyperplane function

$$y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) = |(\boldsymbol{w}^T\boldsymbol{x}_i + b)|, \boldsymbol{x}_i \in DataSet$$

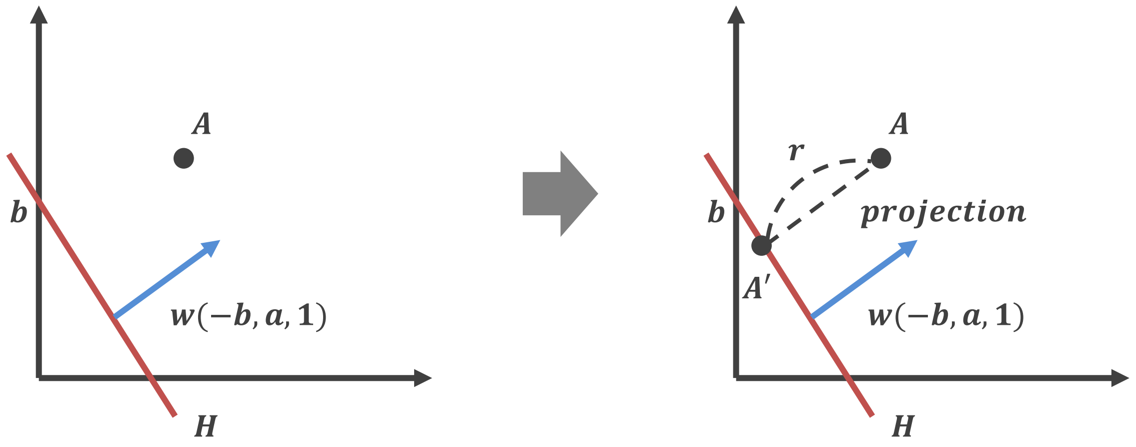
   - There is a problem that the margin can be changed easily

Linear SVMs

▪ How to choose hyperplane

2. Geometric margin

- Euclidean distance between point and hyperplane



Linear SVMs

How to choose hyperplane

- Unit vector : $\boldsymbol{u} = {}^{\boldsymbol{w}}/_{|\boldsymbol{w}|}$
- Orthogonal vector : $r * \boldsymbol{u}$
- Projected vector : $\boldsymbol{x}' = \boldsymbol{x} - yr\,{}^{\boldsymbol{w}}/_{|\boldsymbol{w}|}$
- $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}' + \boldsymbol{b} = 0$

$$\boldsymbol{w}^{\mathrm{T}}\left(\boldsymbol{x} - yr\,{}^{\boldsymbol{w}}/_{|\boldsymbol{w}|}\right) + b = 0$$

$$r = y\,{}^{(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b)}/_{\boldsymbol{w}}$$

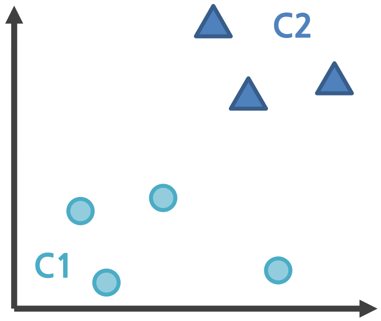
Linear SVMs

▪ How to choose hyperplane

- Find the hyperplane with the maximum margin

1. We have a dataset $\mathcal{D}$ and you want to classify it

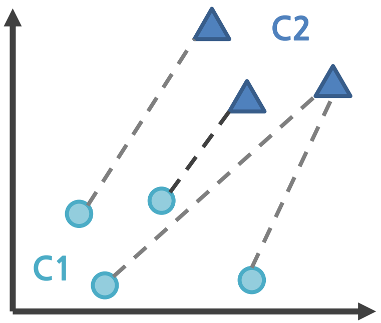
$$\mathcal{D} = \left\{(x_i, y_i) | x_i \in \mathbb{R}^d, y_i \in \{-1,1\}\right\}_{i=1}^{n}$$

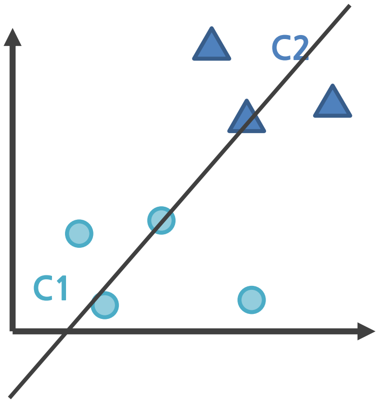


Linear SVMs

- **How to choose hyperplane**

2. Find the minimum distance between data with different class labels

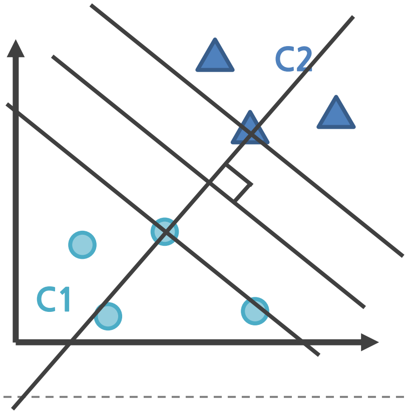




Linear SVMs

▪ How to choose hyperplane

3. Find a hyperplane with the maximum margin perpendicular to the hyperplane connecting the two vectors



Linear SVMs

- **Mathematical summary**

1. We have a dataset $\mathcal{D}$ and you want to classify it

$$\mathcal{D} = \left\{(x_i, y_i) | x_i \in \mathbb{R}^d, y_i \in \{-1,1\}\right\}_{i}\overset{n}{= 1}$$

Linear SVMs

▪ Mathematical summary

2. **We need to select two hyperplanes separating the data with no points between them**

$for\ x_i\ having\ the\ class\ -1$

$w \cdot x_i + b \leq -1$

$for\ x_i\ having\ the\ class\ 1$

$w \cdot x_i + b \geq 1$

$And\ multiply\ both\ sides\ by\ y_i, and\ then\ we\ get\ it$

$y_i(w \cdot x_i + b) \geq 1\ \ for\ \forall i\ (1 \leq i \leq n)$

Linear SVMs

▪ Mathematical summary

3. Maximize the distance between the two hyperplanes

$unit\ vector\ \boldsymbol{u} : \boldsymbol{w}/||\boldsymbol{w}||$

$vector\ \boldsymbol{k} = m \cdot \boldsymbol{u}$

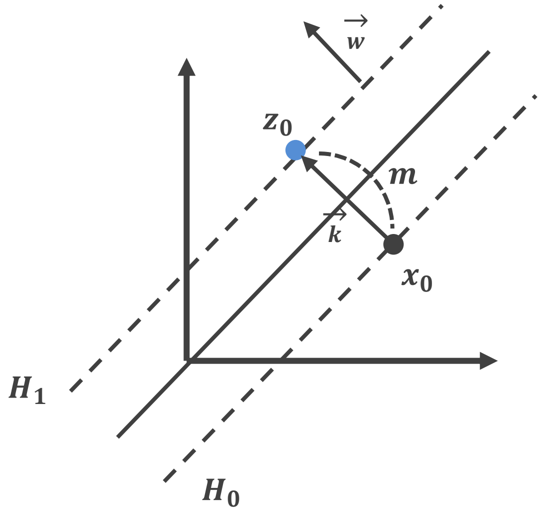
$vector\ \boldsymbol{z_0} = \boldsymbol{k} + \boldsymbol{x_0}$

$in\ H_1, w \cdot z_0 = -b + \delta$

$w \cdot (x_0 + k) = -b + \delta$

$w \cdot x_0 + m||w|| = -b + \delta$

$m = 2\delta/||w||$



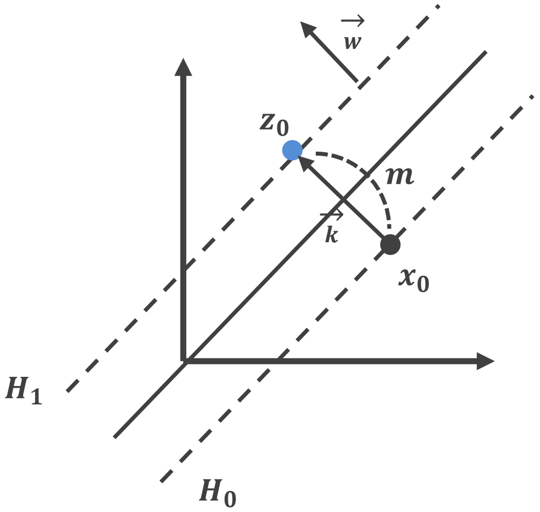
Linear SVMs

▪ Mathematical summary

3. Maximize the distance between the two hyperplanes

$$m = {}^{2\delta}/_{||w||}\ \text{is maximized,}$$

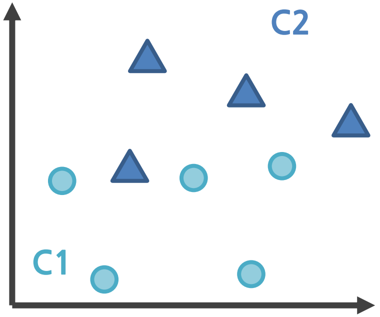
$$\text{oppositely,} \frac{1}{2} w^{\mathrm{T}} \cdot w \text{ is minimized}$$



Linear SVMs issue

Weakness of linear SVMs

- When data can’t be classified linearly,

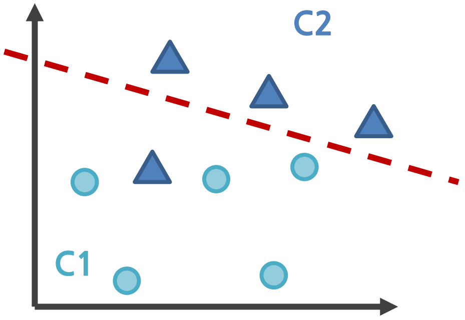


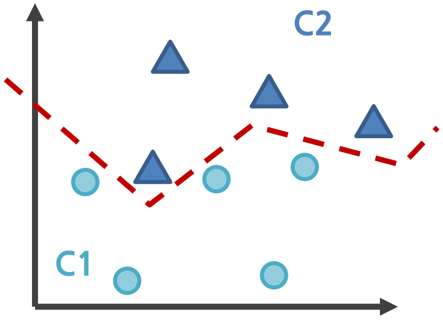
The problem is that this is a common case !

Linear SVMs issue

- **How to solve it ?**

1. Allow some errors
2. Using non-linear hyperplane (Decision boundary)





Soft Margin SVMs

Strategies

- Allow some errors
- A penalty is given for errors: slack variables ξ_i

$$ {}^{1}\!/_{2}\, w^{\mathrm{T}} \cdot w + C \sum_i \xi_i \ \ \textit{is minimized} $$

$$ \textit{and for all } \{(\boldsymbol{x_i}, y_i)\}, y_i\left(\boldsymbol{w^T} \cdot \boldsymbol{x_i} + b\right) \geq 1 - \xi_i $$

$$ \xi_i \geq 1 - y_i\left(\boldsymbol{w^T} \cdot \boldsymbol{x_i} + b\right) $$

$$ \textit{if } \xi_i = 0, \textit{correct classification} $$

$$ \textit{else if } 0 < \xi_i < 1, \textit{correct, but exceeded} $$

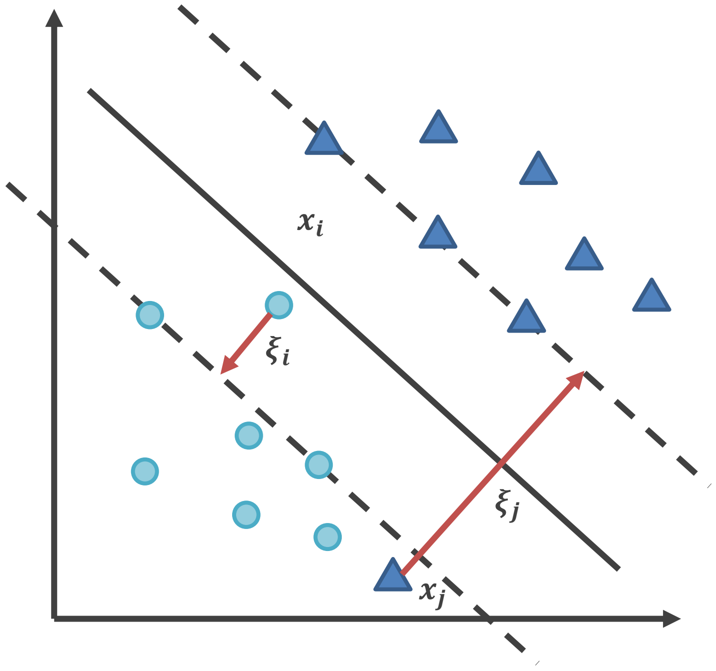
$$ \textit{else } \xi_i \geq 1, \textit{misclassified} $$

Soft Margin SVMs

- **How much error does it allow ?**
  - Tuning parameter: C $(regularization\ term)$
  - The threshold for the errors
  - Typically, C is a user input parameter
  - If C is too large, underfitting occurs
  - If C is too small, overfitting occurs

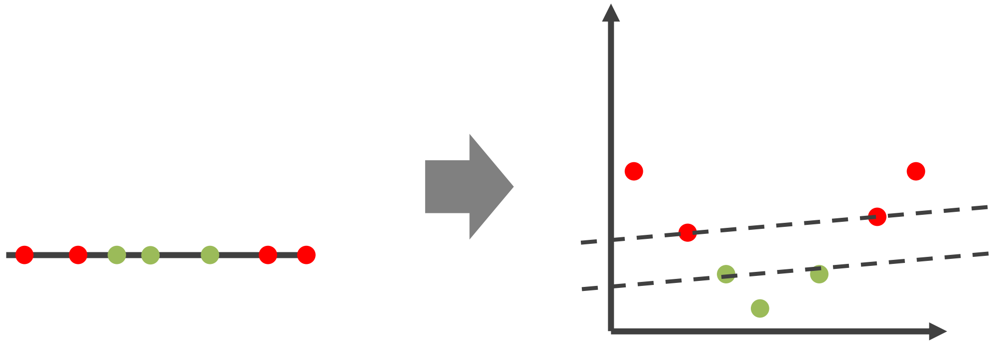
Soft Margin SVMs

- Example figure



Non-linear SVMs

- It does not allow errors
  - Use a strong margin as is
  - How ? $Kernel\ trick: K(x, y)$
  - Map a dataset to a higher dimensional space



Non-linear SVMs

▪ How to apply *kernel trick* ?

- Must be converted to applicable form first
- Lagrange dual problem
  - ✓ Converting a minimization problem to a maximization problem
  - ✓ Satisfy *KKT condition* to reduce duality gap

    (For more information, search Karush-Kuhn-Tucker conditions)

Non-linear SVMs

- How to apply *kernel trick* ?

  - Lagrange dual problem

$$\min_{w,b} ||\boldsymbol{w}||$$

$$s.t. \left(wx_j + b\right)y_j \geq 1, \forall j$$

Transformation

$$L(w, b, \alpha) = \frac{1}{2} w \cdot w - \sum_j \alpha_j \left[\left(wx_j + b\right)y_j - 1\right]$$

$$= \sum_j \alpha_j - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j x_i x_j y_i y_j$$

$$\therefore \max_{\boldsymbol{\alpha} \geq 0} \boldsymbol{L}(\boldsymbol{x}, \boldsymbol{\alpha})$$

Non-linear SVMs

- How to apply *kernel trick* ?

$$\varphi(x_i)\varphi(x_j) = K(x_i, x_j)$$

$$L(w, b, \alpha) = \sum_j \alpha_j - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

$$w = \sum_j \alpha_j \varphi(x_j) y_j \, , b = y_j - w x_j \Rightarrow b = y_j - \sum_i \alpha_i \varphi(x_i) y_i \varphi(x_j)$$

$$f(\varphi(x)) = sign\left(\sum_i \alpha_i y_i K(x_i, x) + y_j - \sum_i \alpha_i y_i K(x_i, x_j)\right)$$

Non-linear SVMs

- How to apply ***kernel trick*** ?

- Typically, choose from three main kernels

1. Quadratic kernel

$$K(x, y) = (xy + 1)^p$$

2. Radial basis function (rdf)

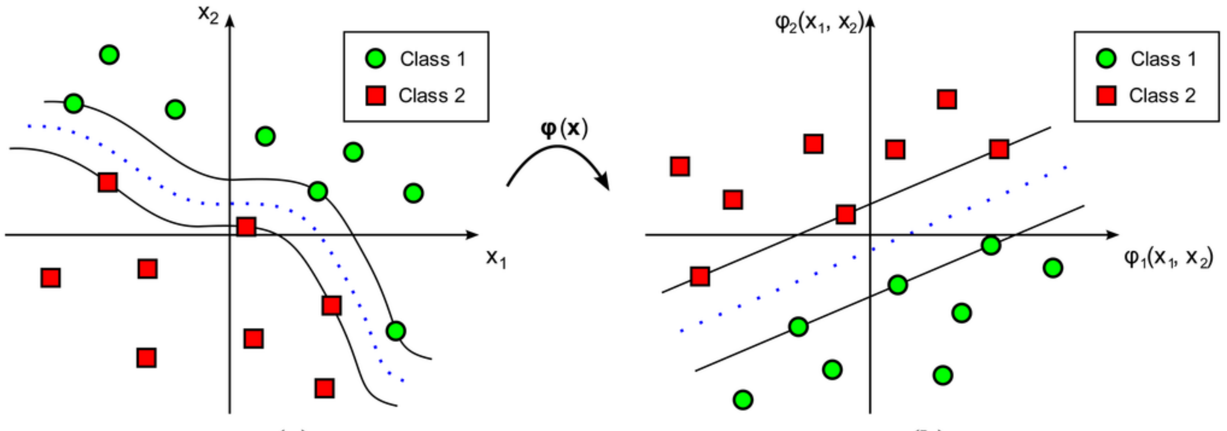
$$K(x, y) = e^{-\frac{|x-y|^2}{2\sigma^2}}$$

3. Hyperbolic tangent

$$K(x, y) = \tanh(axy + \beta)\,, commonly\ \alpha = 2, \beta = 1$$

Non-linear SVMs

- The result of the *kernel trick*



(a) (b)

Other issue

- **Multiclass SVMs**
  - It is generally a binary classifier
  - It can classify multiclass in a naïve approach
  - Recursively classify 1:N
  - Data point that is not classified or has multi-class labels may exist

Example – performing OCR with SVMs

- Perform Optical Character Recognition   (OCR)

- Purpose
  - Process paper-based documents by converting printed or handwritten text into an electronic form

- This is a difficult problem due to the many variants in handwritten style and printed fonts

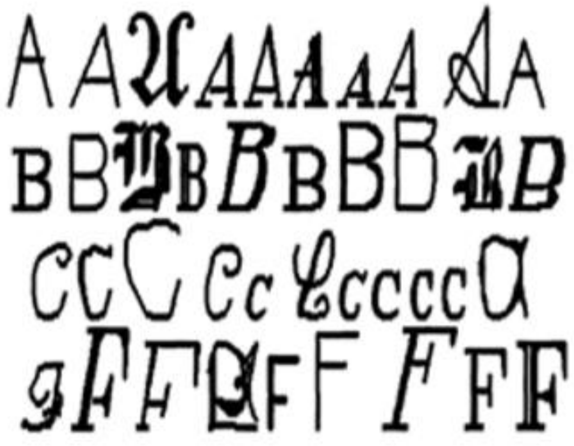
- Errors or typos can result in embarrassing or costly mi stakes in a business environmen

Example – performing OCR with SVMs

- Step 1 – collecting data
  - Dataset
    - Letter dataset
    - Can be downloaded from UCI Machine Learning Data Repository
      - http://archive.ics.uci.edu/ml
  - Characteristics of the dataset
    - The dataset contains 20,000 examples of 26 English al phabet capital letters as printed using 20 different ran domly reshaped and distorted black and white fonts

Example – performing OCR with SVMs

- Step 1 – collecting data
  - The following figure provides an example of some of the printed glyphs



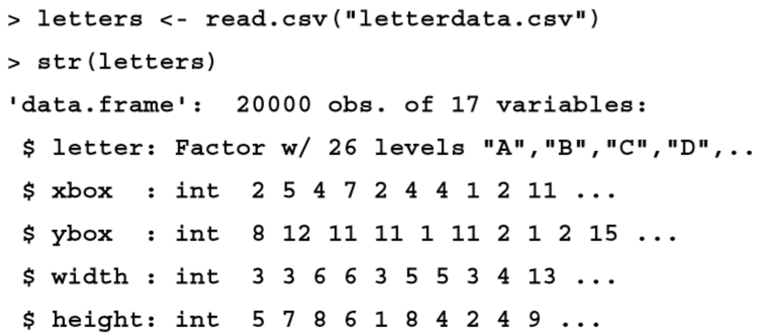
Example – performing OCR with SVMs

- Step 2 – exploring and preparing the data
  - Import the CSV data file

    > letters <- read.csv("letterdata.csv")

  - Confirm that we have received the data with the 16 feat
    ures that define each example of the letter class

    > str(letters)



```
> letters <- read.csv("letterdata.csv")
> str(letters)
'data.frame':  20000 obs. of 17 variables:
 $ letter: Factor w/ 26 levels "A","B","C","D",..
 $ xbox  : int  2 5 4 7 2 4 4 1 2 11 ...
 $ ybox  : int  8 12 11 11 1 11 2 1 2 15 ...
 $ width : int  3 3 6 6 3 5 5 3 4 13 ...
 $ height: int  5 7 8 6 1 8 4 2 4 9 ...
```

Example – performing OCR with SVMs

- Step 2 – exploring and preparing the data
  - The first 16,000 records (80 percent) to build the mod el

```
> letters_train <- letters[1:16000, ]
```

  - The next 4,000 records (20 percent) to test

```
> letters_test  <- letters[16001:20000, ]
```

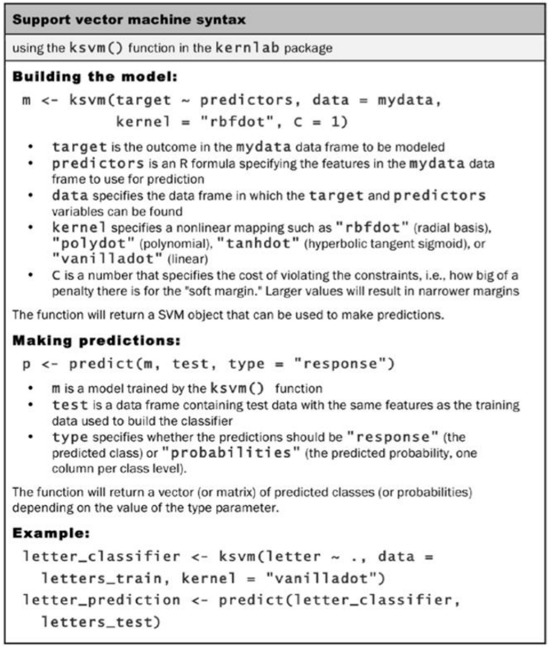
  - The data have already randomized, so no need to pe rform random function

Example – performing OCR with SVMs

- Step 3 – training a model on the data
  - When it comes to fitting an SVM model in R, there are se veral outstanding packages to choose from
  - The e1071 package from the Department of Statistics at the Vienna University of Technology
    - Provides an R interface to the award winning LIBSVM library, a wid ely used open source SVM program written in C++
  - The klaR package from the Department of Statistics at the Dortmund University of Technology
    - Provides functions to work with this SVM implementation directly within R
  - kernlab package

Example – performing OCR with SVMs

- Step 3 – training a model on the data



Support vector machine syntax

using the `ksvm()` function in the `kernlab` package

Building the model:

```
m <- ksvm(target ~ predictors, data = mydata,
          kernel = "rbfdot", C = 1)
```

- `target` is the outcome in the `mydata` data frame to be modeled
- `predictors` is an R formula specifying the features in the `mydata` data frame to use for prediction
- `data` specifies the data frame in which the `target` and `predictors` variables can be found
- `kernel` specifies a nonlinear mapping such as `"rbfdot"` (radial basis), `"polydot"` (polynomial), `"tanhdot"` (hyperbolic tangent sigmoid), or `"vanilladot"` (linear)
- C is a number that specifies the cost of violating the constraints, i.e., how big of a penalty there is for the "soft margin." Larger values will result in narrower margins

The function will return a SVM object that can be used to make predictions.

Making predictions:

```
p <- predict(m, test, type = "response")
```

- `m` is a model trained by the `ksvm()` function
- `test` is a data frame containing test data with the same features as the training data used to build the classifier
- `type` specifies whether the predictions should be `"response"` (the predicted class) or `"probabilities"` (the predicted probability, one column per class level).

The function will return a vector (or matrix) of predicted classes (or probabilities) depending on the value of the type parameter.

Example:

```
letter_classifier <- ksvm(letter ~ ., data =
  letters_train, kernel = "vanilladot")
letter_prediction <- predict(letter_classifier,
  letters_test)
```

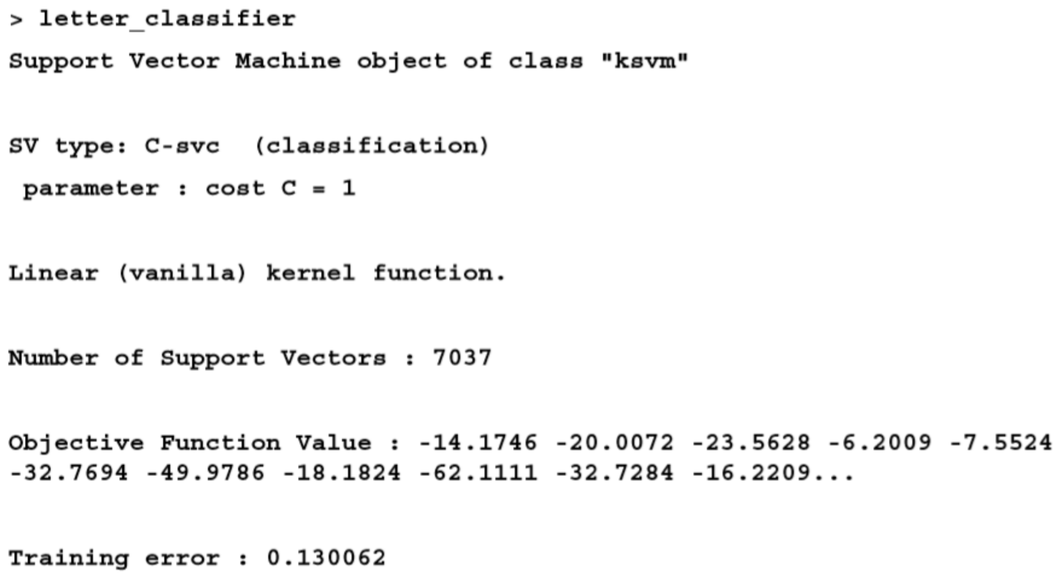
Example – performing OCR with SVMs

- Step 3 – training a model on the data
  - Call the ksvm() function on the training data and sp ecify the linear (that is, vanilla) kernel using the vani lladot option

```
> install.packages("kernlab")
> library(kernlab)
> letter_classifier <- ksvm(letter ~ ., data = letters_train,
                              kernel = "vanilladot")
```

Example – performing OCR with SVMs

- Step 3 – training a model on the data
  - Result of ksvm() function



```
> letter_classifier
Support Vector Machine object of class "ksvm"


SV type: C-svc  (classification)
 parameter : cost C = 1


Linear (vanilla) kernel function.


Number of Support Vectors : 7037


Objective Function Value : -14.1746 -20.0072 -23.5628 -6.2009 -7.5524
-32.7694 -49.9786 -18.1824 -62.1111 -32.7284 -16.2209...


Training error : 0.130062
```

Example – performing OCR with SVMs

- Step 4 – evaluating model performance
  - The predict() function allows us to use the letter classification mod el to make predictions on the testing dataset

```
> letter_predictions <- predict(letter_classifier, letters_test)
```

  - Because we didn't specify the type parameter, the type = "respon se" default was used

  - This returns a vector containing a predicted letter for each row o f values in the test data

  - Using the head() function, we can see the following result

```
> head(letter_predictions)
[1] U N V X N H
Levels: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
```

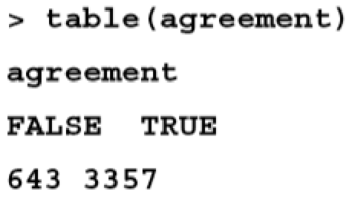
Example – performing OCR with SVMs

- Step 4 – evaluating model performance
  - To examine how well our classifier performed, we n
    eed to compare the predicted letter to the true lett
    er in the testing dataset
    - table() function

```
> table(letter_predictions, letters_test$letter)
letter_predictions   A   B   C   D   E
                 A 144   0   0   0   0
                 B   0 121   0   5   2
                 C   0   0 120   0   4
                 D   2   2   0 156   0
                 E   0   0   5   0 127
```

Example – performing OCR with SVMs

- Step 4 – evaluating model performance
  - The following command returns a vector of TRUE or FALS
    E values, indicating whether the model's predicted letter a
    grees with the actual letter in the test dataset
    - > agreement <- letter_predictions == letters_test$letter
  - From result, we see that the classifier correctly identified t
    he letter in 3,357 out of the 4,000 test records



```
> table(agreement)
agreement
FALSE  TRUE
643 3357
```

Example – performing OCR with SVMs

- Step 5 – improving model performance
  - By using a more complex kernel function, we can map the dat a into a higher dimensional space, and potentially obtain a be tter model fit
    - Gaussian RBF kernel

```
> letter_classifier_rbf <- ksvm(letter ~ ., data = letters_train, kernel = "rbfdot")
```

    - Next, we make predictions as done earlier

```
> letter_predictions_rbf <- predict(letter_classifier_rbf, letters_test)
```

    - Finally, we'll compare the accuracy to our linear SVM

```
> agreement_rbf <- letter_predictions_rbf == letters_test$letter
> table(agreement_rbf)

agreement_rbf
FALSE  TRUE
  275  3725
```

Referencec

- [1] An Introduction to Information Retrieval, Stanford press
- [2] An SVM-Based Classifier for Estimating the State of Various Rotating Components in Agro-Industrial Machinery with a Vibration Signal Acquired from a Single Point on the Machine Chassis, Sensors, MDPI
- [3] SVM – Understanding the math, www.svm-tutorial.com
- [4] Linear Algebra, LadislauFernandes, Youtube
- [5] Learning: Support Vector Machine, MIT OpenCourseWare, Youtube

Q n A