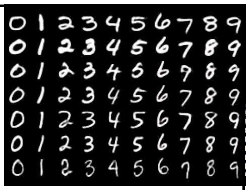


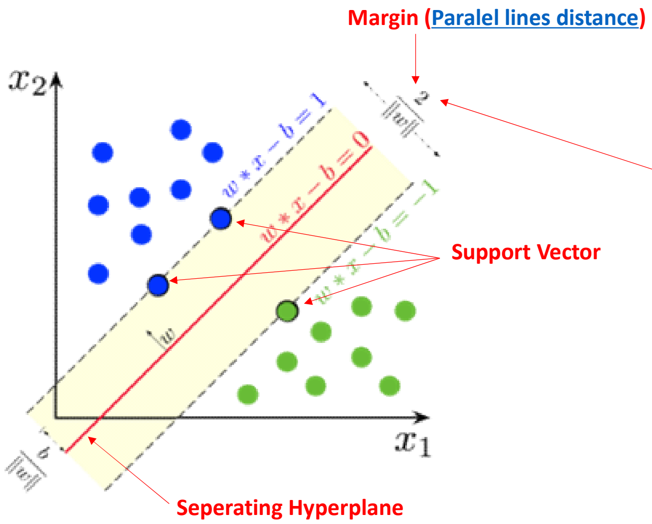
ROB 311-Task 4
SVM (Support Vector Machine)

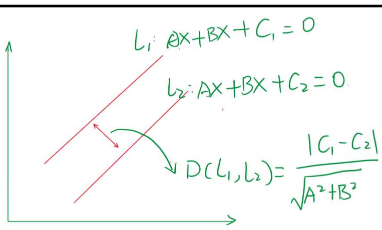
Adriana TAPUS & Chuang YU

adriana.tapus@ensta-paris.fr & chuang.yu@ensta-paris.fr

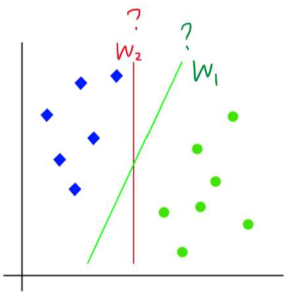
10-2020

1. Introduction of SVM





Objective: update W, Maximize Margin



Linear SVM Classifier

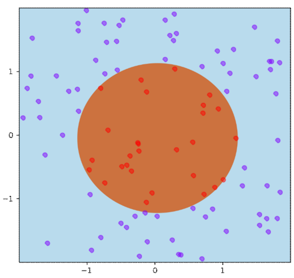
1. Introduction of SVM

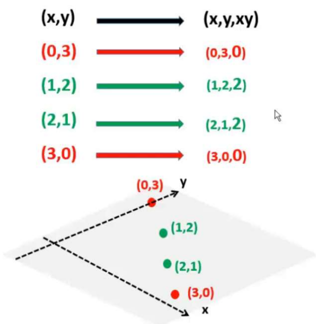
Non-linear SVM Classifier

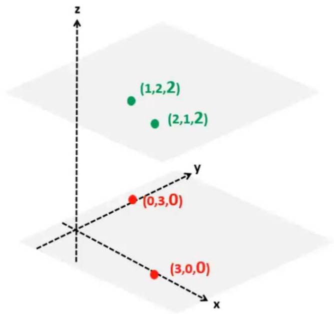
Non-linearly Separable Data

Solution: transfer the lower-dimensional feature e.g. (x,y) into higher-dimensional feature e.g.(x, y, xy) space.

$$\Phi: X \rightarrow \varphi(X)$$







https://www.youtube.com/watch?v=vMmG_7Jcflc&t=29s
https://www.youtube.com/watch?time_continue=2&v=3liCbRZPrZA&feature=emb_logo

1. Introduction of SVM

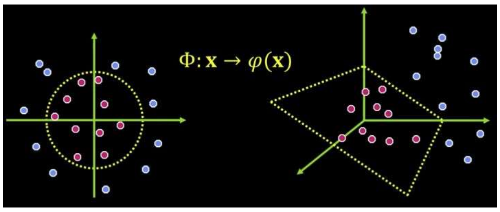
Kernel Trick in non-linear SVM

Non-linear SVM Classifier:

Dot Product (similarity)

$$f(\mathbf{x}) = sign(\sum_i \alpha_i \mathbf{x_i} \cdot \mathbf{x} + \mathbf{b})$$

Support Vector



$$f(\mathbf{x}) = sign(\sum_i \alpha_i K(\mathbf{x_i}, \mathbf{x}) + \mathbf{b})$$

$$f(\mathbf{x}) = sign(\sum_i \alpha_i \boxed{\varphi(\mathbf{x_i}) \cdot \varphi(\mathbf{x})} + \mathbf{b}) \longrightarrow K(\mathbf{x_1}, \mathbf{x_2}) = \varphi(\mathbf{x_1}) \cdot \varphi(\mathbf{x_2})$$

Kernel Function

So we do not need to get higher-domainal features,

we only need the dot product of them.

http://cs229.stanford.edu/notes/cs229-notes3.pdf

1. Introduction of SVM

Kernel Trick in non-linear SVM

Some common kernels include:

- Polynomial (homogeneous): $k(\vec{x_i}, \vec{x_j}) = (\vec{x_i} \cdot \vec{x_j})^d$.
- Polynomial (inhomogeneous): $k(\vec{x_i}, \vec{x_j}) = (\vec{x_i} \cdot \vec{x_j} + 1)^d$.
- Gaussian radial basis function: $k(\vec{x_i}, \vec{x_j}) = \exp(-\gamma \|\vec{x_i} - \vec{x_j}\|^2)$

$$f(\mathbf{x}) = sign(\sum_i \alpha_i K(\mathbf{x_i}, \mathbf{x}) + \mathbf{b})$$

$$f(\mathbf{x}) = sign(\sum_i \alpha_i \boxed{\varphi(\mathbf{x_i}) \cdot \varphi(\mathbf{x})} + \mathbf{b}) \longrightarrow K(\mathbf{x_1}, \mathbf{x_2}) = \varphi(\mathbf{x_1}) \cdot \varphi(\mathbf{x_2})$$

Kernel Function

So we do not need to get higher-domainal features,

we only need the dot product of them.

1. Introduction of SVM

Kernel Trick in non-linear SVM

$$f(\mathbf{x}) = sign(\sum_i \alpha_i \varphi(\mathbf{x_i}) \cdot \varphi(\mathbf{x}) + \mathbf{b}) \xrightarrow[\text{Kernel Function}]{K(\mathbf{x_1}, \mathbf{x_2}) = \varphi(\mathbf{x_1}) \cdot \varphi(\mathbf{x_2})} f(\mathbf{x}) = sign(\sum_i \alpha_i K(\mathbf{x_i}, \mathbf{x}) + \mathbf{b})$$

Kernel Function?

Lower-domain. $x = [x_1, x_2]^T$

$\Phi: X \to \varphi(X)$

higher-domain: $\phi(x) = [x_1x_1, x_1x_2, x_2x_1, x_2x_2]$

$x = [x_1, x_2]^T$

eg. $x = [1,2]^T$, $y = [3,4]^T$.

$\phi(x) = \phi(1,2) = [1,2,2,4]^T$

$\phi(y) = \phi(3,4) = [9,12,12,16]^T$

$\Rightarrow \phi(x) \cdot \phi(y) = [1,2,2,4] \cdot [9,12,12,16]$

$= 1\times9 + 1\times12 + 1\times12 + 1\times16$
$+ 2\times9 + 2\times12 + 2\times12 + 2\times16$
$+ 2\times9 + 2\times12 + 2\times12 + 2\times16$
$+ 4\times9 + 4\times12 + 4\times12 + 4\times16$
$= 121$

Time complexity ↑

if use kernel trick

$K(x,y) = k([1,2],[3,4]) = (x^T y)^2$

$= (1\times3 + 2\times4)^2 = 121$ Time complexity ↓

3. Task 4: SVM

ROB311 – TP4 – SVM Digit Recognition

Introduction

Today, you will use **Support Vector Machines** and **Python** in order to implement a **digit recognition** algorithm. The database used to train and test your algorithm is the MNIST dataset, containing grayscale (8-bit), 28x28 pixels images of hand written digits. This is one of the reference digit recognition datasets in the world and, as you can see, state-of-the-art SVM algorithms can achieve error rates as low as 0.56 to 1.4 %.

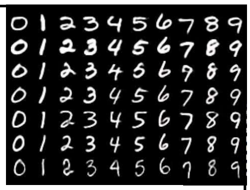
Files

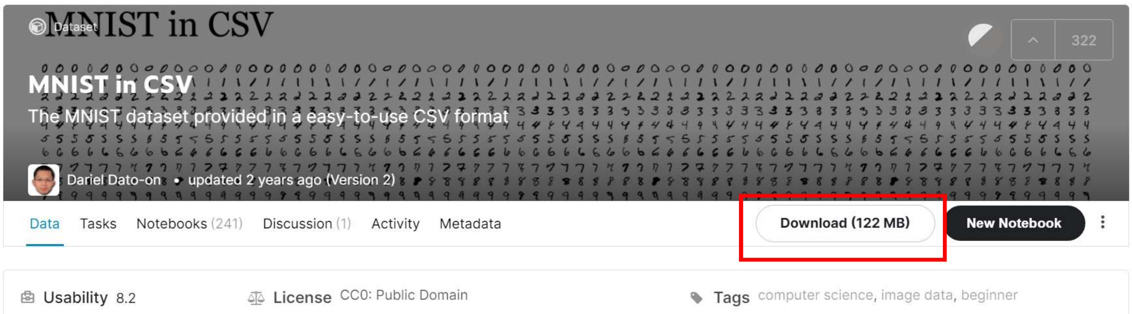
The two .csv files containing the MNIST dataset (both training and test set) can be downloaded here (using the Download button). Each of the two files contains 785 columns, the first column corresponding to the **label** of each sample (a digit from 0 to 9), while the other 784 columns contain the **colour intensity value** (8-bit, 0 to 255) for each of the pixels of a 28x28 image.

The training set (*mnist_train.csv*) contains 60.000 samples, while the test set (*mnist_test.csv*) contains 10.000 samples.

ENSTA IP PARIS

3. Task 4: SVM





https://www.kaggle.com/oddrationale/mnist-in-csv

ENSTA IP PARIS

3. Task 4: SVM

Objectives

Implement the digit recognition algorithm using Support Vector Machines trained on the MNIST hand written digit dataset contained in the *mnist_train.csv* file. Then, test your algorithm on the provided *mnist_test.csv* file.

You will have to compute:

- **the overall detection accuracy** (the percentage of correctly recognised digits from the test set)
- **a confusion matrix** (of size 10x10)

Both the detection accuracy and confusion matrix can be simply displayed in a terminal, but feel free to use any graphic libraries you want to display them.

Good news!

You can use sklearn or any other library you want :)

ENSTA IP PARIS

sklearn.decomposition.PCA

3. Task 4: SVM

scikit-learn Machine Learning in Python

Scikit-learn: a free software machine learning library for the Python programming language.

SVC, NuSVC and LinearSVC are classes capable of performing binary and multi-class classification on a dataset.

Similar, More kernels **Only linear kernel**

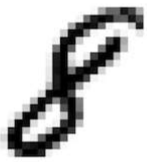


Image input size:

28*28=784 (so big)

Dimensionality Reduction e.g. PCA → **output size: e.g.150** → **SVM Classifier** → **?**

sklearn.decomposition.PCA

https://scikit-learn.org/stable/modules/svm.html

Principal component analysis

ENSTA IP PARIS

Rule
--2 persons in one group
--Deadline: Before Monday
Submit:
--Report paper + code
--Github or ENSTA gitlab

GitHub or

End!
Question?