-UNITE!-
University Network for Innovation, Technology and Engineering

---

Aalto University

School of Science

unite!

Human-Centered Machine Learning Course

Support Vector Machines

Professor:
Cecilio Angulo

Authors:
Pablo De Ramon
Giang Le
Vasco Pearson
Amir Ingher
Sergi Garcia
Iván Quirante

Academic Course 2022-2023

2. Practice

The main aim in this practice work is to understand how hyperparameters

- kernel, $k(\cdot, \cdot)$, especially polynomial and Gaussian
- parameter associated to kernel, p
- regularizaton term C

work on a kernel machine when classifying a dataset.
It is an open practice. Try to work with concepts like high bias, high variance, weights, size, local/general solutions, linear/nonlinear, classification, multiclassification, number of support vectors, margin,. . .
The process should be similar to:

- Select / build a dataset with, at least, 200 training vectors, but less that 1000, in dimension less than five.
- Select only two dimensions (using PCA, feature selection, ...) to facilitate the visualization of the results.
- Play time. At least, five experiments should be completed and analysed.

Solution
Support vector machines are supervised learning models that can be used for regression and classification problems. In this assignment we will focus on using support vector machines for classification (particularly multi-class classification). In a simple classification problem where the classes are linearly separable, what the support vector machine does is to fit a **hyperplane** to an N-dimensional space (N is the number of features) and classifies the data points depending on which side of the hyperplane they lie. However, data is not always linearly separable and some **transformations** need to be applied, which map the data from the original space into a **higher dimensional feature space**. The goal is that after the transformation the classes are now linearly separable and one can then fit a hyperplane with the training data to later make accurate predictions.
However, this leads to a problem in real applications: there may be too many features and applying transformations that involve many mathematical combinations of them could lead to extremely high and impractical computational costs. The solution to this problem is the **kernel trick**. Kernel methods are used as a way of measuring the similarity/difference between pairs of datapoints instead of having to explicitly apply the transformation $\phi(x)$ and representing the data by these transformed coordinates in the higher dimensional feature space.

In this practice we decided to use the Iris dataset - common in the Machine Learning literature - to demonstrate how some classifier models work. The dataset contains 4 features (sepal length, sepal width, petal length, petal width) all measured in centimetres. Since the dataset only has 150 data points, more have been manually created by duplicating each datapoint and adding some random noise to it, finally creating a dataset with 300 datapoints.

In order to project data to a lower dimensional space of only two components and help visualize the results, a **Principal Component Analysis (PCA)** was applied. To do so, data was first standarized because in this way the important information is extracted from it and then reduced to a set of summary indices known as **principal components**. After applying PCA we now have a dataset with 2 features and 300 datapoints where we can try different suport vector machine models. Bellow there is a table with the 6 models tried and the results achieved:

Model	Kernel param.	Regularizaton term	Train accuracy	Val accuracy
SVC with linear kernel	none	3	0.9667	0.95
LinearSVC	none	0.1	0.9167	0.95
SVC with rbf kernel	$\gamma = 0.5$	3	0.9667	0.95
SVC with rbf kernel	$\gamma = 1.5$	1	0.9667	0.95
SVC with polinomial kernel	degree=3	1	0.9625	0.9
SVC with polinomial kernel	degree=3	3	0.9667	0.9167

The first two are simple suport vector machines with a linear kernel. The difference between the two python classes is that *SVC* takes a one-vs-one approach to multi-class classification, while *LinearSVC* opts for one-vs-all approach. Moreover, *LinearSVC* has more flexibility in the choice of penalties and loss functions and should scale better to large numbers of samples. After a few experiments we concluded that the best values for the regularization term where the ones in the table above. This parameter tells the SVM optimization algorithm how much we want to avoid misclassifying each training example. If it is high we are giving a lot of importance to the loss term that makes sure that the weights classify the training data points correctly, whereas a low value means that simpler weights are being prioritized..

We also tried two SVMs with a radial basis function (rbf) kernel. The mathematical expression for this kernel is

$$K(x, x') = \exp\left(-\frac{||x - x'||^2}{2\sigma^2}\right)$$

We now have a $\gamma = \frac{1}{2\sigma^2}$ parameter that needs to be tuned. It can be thought of as the 'spread' of the kernel and therefore the size of the decision region. When γ is too small, the model is too constrained and cannot capture the complexity or "shape" of the data. Contrastingly, when gamma is too large, the radius of the area of incluence surrounding the data point is smaller, which can then create islands of decision-boundaries around each observation. The values for the regularization term and γ that achieved the highest validation error are shown in the table above.

Finally we decided to try two SVMs with a polynomial kernel. For degree d, the mathematical expression for this kernel is

$$K(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^{\mathrm{T}}\mathbf{x}' + c)^d$$

These models slightly underperformed the previous ones, having achieved lower accuracy on the validation set. However the degree that achieved the best results was degree 3, and later several regularization terms were tried with this option before concluding that the best value was also 3.

Bellow we can see the visualization of the results for these 6 models.