Non Linear Support Vector Machine (SVM)*

1st t Evrard Leuteu Feukeu
Hochschule Hamm- Lippstadt
Electrical Engineering
Hamm, Germany
evrard.leuteu-feukeu@stud.hshl.de

***Abstract*—This document is a Overview and description for Non Linear Support Vector Machine(SVM) used in Deep learning. Nonlinear SVM opens endless opportunities to develop computer systems that can learn and adapt without continues assertion of support , explicit instructions of data patterns with the help of algorithm that process, analyse and interprets data into the computer system. A brief tour on how the SVM algorithm functions and will be explain and seen in detail in this document.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Support-vector machines (SVMs, also known as support-vector networks) are supervised learning models that examine data for classification and regression analysis in machine learning. SVMs were developed by Vladimir Vapnik and colleagues at ATT Bell Laboratories. They are based on statistical learning frameworks, an SVM training algorithm produces a model that assigns new examples to one of two categories, making it a non-probabilistic binary linear classifier, given a series of training examples, each marked as belonging to one of two categories (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). SVM maps training examples to points in space in order to widen the distance between the two categories as much as possible. New examples are then mapped into the same space and classified according to which side of the gap they fall on.SVMs can perform non-linear classification as well as linear classification by applying the kernel trick, which involves implicitly translating their inputs into high-dimensional feature spaces.

When data is unlabeled, supervised learning is impossible, hence an unsupervised learning strategy is necessary, in which the data is clustered naturally into groups and new data is mapped to these groups. Hava Siegelmann and Vladimir Vapnik invented the support-vector clustering algorithm, which uses support vector statistics obtained in the support vector machines technique to categorize unlabeled data.

What is the purpose of Support Vector Machines? For classification and regression tasks, Support Vector Machines are supervised learning models. They can solve both linear and nonlinear problems and are useful in a variety of situations. The concept behind Support Vector Machines is straightforward: In a classification problem, for example, the method draws a line between the classes. The line's purpose is to maximize the distance between points on each side of the so-called decision line. After the split, the model can readily guess the target classes (labels) for new examples, which is a big plus.

Maybe you think this just applies to low-dimensional problems, like a data collection with only two features, but you're mistaken! In higher-dimensional spaces, Support Vector Machines are actually quite effective. It works well even when the number of dimensions exceeds the number of samples in a data collection. This is primarily due to the kernel trick, which we will discuss later. In compared to alternative classification algorithms such as k-nearest neighbor or deep neural networks, Support Vector Machines have more memory efficiency, speed, and general accuracy. Of course, they are not always better than deep neural networks, but they do outperform deep neural networks on occasion.

II. Types and Differences

Linear and Non-Linear Data: What's the Difference? To clear things up, I'll quickly describe what linear and non-linear data is all about. When we can classify data with a linear classifier, we refer to it as linear data. A linear combination of attributes is used by the linear classifier to determine his classification choice. In machine learning, these traits are referred to as features. The diagram below will help to clarify things.

III. Implimentation

Solving the optimization yields the parameters of the maximum-margin hyperplane. There are a variety of specialized methods for swiftly handling the quadratic programming (QP) problem that arises from SVMs, with the majority of them relying on heuristics to break the problem down into smaller, more manageable chunks.

Another option is to utilize an interior-point method to solve the Karush–Kuhn–Tucker conditions of the primal and dual issues using Newton-like iterations. Rather than tackling a series of broken-down problems, this method answers the problem in its whole. A low-rank approximation to the matrix is frequently utilized in the kernel technique to avoid solving a linear problem containing the huge kernel matrix.

Another popular method is Platt's sequential minimal optimization (SMO) algorithm, which divides the problem into

two-dimensional sub-problems that may be solved analytically, obviating the requirement for a numerical optimization methodology and matrix storage. For tough SVM tasks, this approach is conceptually simple, straightforward to implement, and has improved scaling features.

The same strategies used to optimize its close relative, logistic regression, can be used to solve the particular case of linear support-vector machines more effectively; this class of algorithms includes sub-gradient descent (e.g., PEGASOS) and coordinate descent (e.g., LIBLINEAR). LIBLINEAR has several promising training-time characteristics. Each iteration of the convergence process is linear in the time it takes to read the train data, and the iterations also have a Q-linear convergence property, making the algorithm incredibly fast.

Sub-gradient descent can also be used to solve general kernel SVMs (for example, P-packSVM), particularly when parallelization is possible. LIBSVM, MATLAB, SAS, SVMlight, kernlab, scikit-learn, Shogun, Weka, Shark, JKernelMachines, OpenCV, and other machine-learning toolkits support kernel SVMs. To improve classification accuracy, data preprocessing (standardization) is highly suggested. Standardization techniques include min-max, normalization by decimal scaling, and Z-score. SVM is commonly performed by subtracting the mean and dividing the variance of each feature.

A. Properties

IV. Error minimization

Solving the optimization yields the parameters of the maximum-margin hyperplane. There are a variety of specialized methods for swiftly handling the quadratic programming (QP) problem that arises from SVMs, with the majority of them relying on heuristics to break the problem down into smaller, more manageable chunks. Another option is to utilize an interior-point method to solve the Karush–Kuhn–Tucker conditions of the primal and dual issues using Newton-like iterations. [41] Rather than tackling a series of broken-down problems, this method answers the problem in its whole. A low-rank approximation to the matrix is frequently utilized in the kernel technique to avoid solving a linear problem containing the huge kernel matrix.

Another option is to use Platt's sequential minimal optimization (SMO) approach, which divides the problem into 2-dimensional sub-problems that can be solved analytically, removing the requirement for a numerical optimization algorithm and matrix storage. For tough SVM tasks, this approach is conceptually simple, straightforward to implement, and has improved scaling features. The same strategies used to optimize its close cousin, logistic regression, can be used to solve the particular case of linear support-vector machines more effectively; this class of algorithms includes sub-gradient descent (e.g., PEGASOS) and coordinate descent (e.g., LIBLINEAR). It has some appealing training-time characteristics. Each repetition of the convergence process takes time linear in the time it takes to read the train data, and the iterations have a Q-linear convergence feature, making the algorithm quick.

V. Extension

VI. Comparison

A. Advantages

B. Disadvantages

VII. Application

References

[1] A. Geron, Hands-on machine learning with Scikit- ´ Learn, Keras, and TensorFlow: concepts, tools, and techniques to build intelligent systems, Second edition. Sebastopol, CA: O'Reilly Media, Inc, 2019. [2] A. C. Muller and ¨ S. Guido, Introduction to machine learning with Python: a guide for data scientists, First edition. Sebastopol, CA: O'Reilly Media, Inc, 2016. [3] V. Lakshmanan, S. Robinson, M. Munn, and an O. M. C. Safari, Machine Learning Design Patterns. 2021. [4] G. Bonaccorso, Machine Learning Algorithms Popular Algorithms for Data Science and Machine Learning, 2nd Edition. Birmingham: Packt Publishing Ltd, 2018. [5] L. Massaron and A. Boschetti, Regression analysis with Python: learn the art of regression analysis with Python. Birmingham Mumbai: Packt Publishing, 2016. [6] https://hands-on.cloud/implementation-ofsupport-vector-machine-svm-using-python/: :text=Nonlinea

Bennett, Kristin P.; Campbell, Colin (2000). "Support Vector Machines: Hype or Hallelujah?" (PDF). SIGKDD Explorations. 2 (2): 1–13. doi:10.1145/380995.380999. S2CID 207753020. Cristianini, Nello; Shawe-Taylor, John (2000). An Introduction to Support Vector Machines and other kernel-based learning methods. Cambridge University Press. ISBN 0-521-78019-5. Fradkin, Dmitriy; Muchnik, Ilya (2006). "Support Vector Machines for Classification" (PDF). In Abello, J.; Carmode, G. (eds.). Discrete Methods in Epidemiology. DIMACS Series in Discrete Mathematics and Theoretical Computer Science. Vol. 70. pp. 13–20. Ivanciuc, Ovidiu (2007). "Applications of Support Vector Machines in Chemistry" (PDF). Reviews in Computational Chemistry. 23: 291–400. doi:10.1002/9780470116449.ch6. ISBN 9780470116449. James, Gareth; Witten, Daniela; Hastie, Trevor; Tibshirani, Robert (2013). "Support Vector Machines" (PDF). An Introduction to Statistical Learning : with Applications in R. New York: Springer. pp. 337–372. ISBN 978-1-4614-7137-0. Schölkopf, Bernhard; Smola, Alexander J. (2002). Learning with Kernels. Cambridge, MA: MIT Press. ISBN 0-262-19475-9. Steinwart, Ingo; Christmann, Andreas (2008). Support Vector Machines. New York: Springer. ISBN 978-0-387-77241-7. Theodoridis, Sergios; Koutroumbas, Konstantinos (2009). Pattern Recognition (4th ed.). Academic Press. ISBN 978-1-59749-272-0.