RESEARCH ARTICLE OPEN ACCESS

Performance Evaluation Of Support Vector Machines (Svms)And Convolutional Neural Network (Cnn) On Binary Classification Problem

Abdelaziz Botalb*
**(Department of Electrical and Computer Engineering, King Abdulaziz University, Saudi Arabia*

ABSTRACT
Support vector machines (SVMs) have been around for decades, they have been used for a number of classification tasks. They actually have a very strong theory behind them, which make it relatively easy to choose the best hyper-parameters. The kernel trick makes it easier for SVMs to implicitly classify in higher dimensional space, making it possible to work with nonlinearly separable datasets. On the other hand, Convolutional Neural Networks (CNNs) have gained important attention in recent years for their high performance in image classification problems with high number of categories. The automatic feature extraction of convolutional layer and the dimensionality reduction of the pooling layer make CNN gain high predictive power on testing data. In this work both models are briefly discussed and implemented on a binary classification problem from the EMIST character dataset. The CNN outperformed SVM achieving a misclassification error rate on test data of 1.7 % against 2.32 % for SVM.

Keywords – Support Vector Machine, Convolutional Neural Network, Cost Function, Optimization, Predictive Power, Hyper-parameters, Kernel trick.

Date Of Submission:13-09-2018 Date Of Acceptance: 28-09-2018

I. INTRODUCTION

In machine learning domain and more especially in the supervised learning, it is not possible to state that there is a certain algorithm that works perfect for every task at hand. As an example, we cannot say that Support Vector Machines (SVMs) are always better than Convolutional Neural Networks (CNNs) or vice-versa, and this is because there are many factors to consider such as size, dimensionality, variance, and structure of the dataset used for training and testing the model. Therefore, various appropriate algorithms should be tried for a specific problem and a hold-out method should be used to pick up the best one with the highest predictive power. In this paper, two machine learning models SVM and CNN will be evaluated for a binary image classification problem. Our target is to find out how the very well-known old SVM will perform in front of the recently pioneered CNN, knowing that CNN has proved to perform very well in multiple class problems compared to other algorithms including SVM. In this work, a binary classification problem will be given to CNN and SVM to find out whether SVM can win CNN for problems with less number of classes.

There are a number of related works to this paper but none of them was applied on a binary classification problem on two easily recognized letters such as the ones we are using “O” and “R”. The main purpose of these chosen two letters is the huge difference in their shape, meaning that they can be easily distinguished and they are not hard to classify. This will allow us to find out whether an SVM can still make a higher classification rate compared to CNN which is known to work well in complex problems.

Related Work: In [1] Niu et al. presented a hybrid model by using CNN for feature extraction and SVM as a classifier on the MNIST dataset. Their model achieved an accuracy of 94.40%. In [2], Hong et al. proposed an online visual tracking algorithm using a pre-trained CNN and an online SVM. They tested their approach on a challenging benchmark and they claimed that their approach outperformed state-of-the-art tracking algorithm. In [3], Elleuch et al. also proposed a hybrid model using SVM and CNN on Arabic handwriting recognition. 2.56% error rate was achieved on 24 class dataset. In [4], Toth, et al. evaluated SVM and CNN separately on a plant image recognition problem, and they concluded that CNN outperformed SVM.

II. CONVOLUTIONAL NEURAL NETWORK (CNN)

CNN has gained important attention in recent years especially in image classification problems with multiple classes. The concept of CNN

was actually pioneered by Yann LeCuns paper in 1998 [5], but it was not until 2012 when Alex Krizhevsky,et a.l [6] proposed their CNN architecture in the ILSVRC (ImageNet Large-Scale Visual Recognition Challenge), and their model architecture consisted of 5 conv layers, max-pooling layers, dropout layers, and 3 fully connected layers. The network competed for the classification of 1000 possible classes. The network was the winner of the competition outperforming all other Machin learning approaches, with an outstanding top 5 error rate of 15.4 %, and this result surprised the computer vison community. Since then CNNs have evolved with new techniques and gained more importance and focus. In this work a very brief overview of its internal blocks and operation will be reviewed.
A CNN fundamentally consist of three important layers, convolution, pooling and fully connected.

2.1 Convolutional Layer

This layer is where the features are automatically extracted. The input images are fed as 2-d matrices, unlike SVM and other approaches where images have to be converted into vectors in 1-d format. The input volume at any layer is a 4-d tensor A×B×C×D, where A and B are rows and columns of the image, C is the number of channels, and D is the number of the mini-batch images, in case stochastic gradient descent is used. The weights that have to be learnt are called filters (also called kernels and feature detectors) and are also organized in 4-d tensor E×F×G×H where E and F are the rows and columns of the filters, G is the number of channels of the filters which must be equal the number of channels of the input volume to that layer where C = G, and H is the number of filters. Every filters looks for one feature within the input volume, so the number of filters corresponds to the number of features we want to extract. The operation of the convolution is actually nothing but an algebraic dot product (as a similarity function) of the filters and a receptive field in the input volume, and the output 3-d tensor (which from the input to the next layer) K×T×R where K and T are the rows and columns of the feature map, and R is the number of channels of it, and it must be equal the number of filters used, where R = H.

The hyper-parameters to choose for this layer are: number and size of filters, stride which is how many steps the convolution is moving, and zero-padding, which is actually used if we want to preserve the spatial resolution of the input volume, as convolution is inherently a lossy operation.

2.2 Pooling Layer

This layer has no weights to learn, its primary role is to reduce the size of the input feature maps, and it is a down-sampling layer. There are two common types of pooling, max and average. In max pooling, the maximum value of a certain receptive field is taken and the rest is ignored, where in average pooling, we take the average of that receptive field values, and these values are what represent their receptive field in the next layer. This operation will result in less parameter to be learnt in next layer, which will reduce the possibility of model overfitting and also reduce computation complexity and time. It also makes the model equi-variant to translations of the input image. Some research papers used CNNs without pooling layer, and they used larger filters in convolution layer so as to simulate the operation of pooling.

The hyper-parameters to choose for this layer are: size of filter window, stride, and type of pooling.

2.3 Fully Connected Layer

This is actually the conventional neural network i.e. the Multilayer Perceptron (MLP) network. It is usually used as a classifier at the end of the architecture (MLP is not necessarily, other classifiers could also be used).

The type of activation function affects the convergence and the performance of the algorithm. The most used ones are, **sigmoid**, **hyperbolic tangent**, and **Relu,** each with pros and cons. Activation can be at the end of any layer. Some other hyper-parameters include, learning rate, momentum, regularization factor, number of each layer type, the sequential order of layers, and type of cost function.

2.4 Learning Rule

CNN uses back-propagation algorithm using gradient descent approach so as to learn the weights and biases. The update rule is also derived using optimization by minimizing a cost function with respect to weights and biases. The most common used cost functions in CNN are:
Quadratic cost, also known as mean squared error, maximum likelihood, and sum squared error:

$$J(\boldsymbol{w},\boldsymbol{b},\boldsymbol{a}) = \frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{d}_i - \boldsymbol{a}_i)^2 \qquad (1)$$

Cross-entropy cost, also known as Bernoulli negative log-likelihood and Binary Cross-Entropy:

$$J(\boldsymbol{w},\boldsymbol{b},\boldsymbol{a}) = \sum_{i=1}^{N}[\boldsymbol{a}_i ln(\boldsymbol{d}_i) + (1-\boldsymbol{a}_i)\ln(1-\boldsymbol{d}_i)] \qquad (2)$$

Where:
w: weight
b: biases
$\mathbf{a_i}$: activated output of CNN
$\mathbf{d_i}$: desired output

Cross-entropy cost function is the most commonly used one for image classification

problems and is usually used with **softmax function** at the output of the fully connected layer so as to map the output vector to a probability distribution summing to 1 with the highest probability corresponding to the network prediction.

Parameter update rule is derived by the same way as multilayer perceptron, and this by taking the partial derivative of the cost function with respect to weights and biases.

New weight update rule:

$$w^{t+1} \leftarrow w^t - \mu \frac{\partial J(w,b,a)}{\partial w} \qquad (3)$$

New bias update rule:

$$b^{t+1} \leftarrow b^t - \mu \frac{\partial J(w,b,a)}{\partial b} \qquad (4)$$

Where **μ** is the learning rate.

III. SUPPORT VECTOR MACHINES (SVMS)

SVMs have been known as a very useful machine learning model used for a variety of pattern recognition tasks, including text categorization [7], image classification [8] , protein and Cancer classification [9], and hand-written character recognition [10]. In this work we are not going to give a thorough explanation of how SVMs work but an informative and to-the-point brief about its theory will be highlighted; [11] gives more deeper insight.

SVMs are linear classifiers that predict a decision boundary in the form of a hyper-plane; and they work extremely well in linearly separable classification problems. The weight and bias update rule is obtained by the minimization of a constrained optimization cost function (using Lagrange multiplier) subject to linear inequalities, the cost function is described as the largest distance between the two classes of the dataset which is defined by vector points from the dataset called support vectors. So the aim of the optimization is to maximize the margin between the classes. The cost function is minimized with respect to the biases and weights (in the primal form) and also maximized with respect to the Lagrange multiplier (in the dual form). Hence we are looking for the **Saddle point** in the graph of the cost function. The resulted model is called **hard-margin SVM**, because it does not allow any error to occur and the decision boundary is a perfect fit into the training data. This causes the problem of overfitting which lead this type of SVM to not generalize very well on future unseen data. Also note that the cost function is quadratic and is strict convex allowing for one global minima, and this makes SVMs give excellent results in terms of classification accuracy; always give optimal solution.

Because of the existence of outliers in the dataset a new variable called **slack variable** is introduced to the cost function which penalizes the model for outliers, and allow for errors to occur this gives more generalization to the model which is then called **soft-margin SVM**.

For simplicity, we consider the optimization problem of the hard-margin SVM (without slack variable.).

- The optimization of the primal form of the cost function using Lagrange multiplier will be:

Minimize the cost with respect to **w** and **b**:

$$J(w,b) = \frac{1}{2} w^T w - \sum_{i=1}^{N} \alpha_i d_i (w^T x_i + b) + \sum_{i=1}^{N} \alpha_i \qquad (5)$$

Subject to:

$$\alpha_i \geq \mathbf{0} \quad , \quad \forall_i$$

Where all notations the same as before and **α** is the Lagrange multiplier and **x** is the input data.

This minimization will result in the expression of **w** and **b** in terms of **α**

- The optimization in the Dual form of the cost function will be:

Maximize the cost with respect to **α**:

$$J(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{i=1}^{N} \alpha_i \, \alpha_j \, d_i d_j \; x_i^T x_j \qquad (6)$$

Subject to:

$$\alpha_i \geq \mathbf{0} \quad \textbf{and} \quad \sum_{i=1}^{N} \alpha_i d_i = 0 \quad , \quad \forall_i$$

Using quadratic programing (QP) solver we will find that there are a bunch of α's are zero and only α's corresponding to support vectors which correspond to a certain examples of the dataset, will be non-zero, and using them we can find the values of **w** and **b** found in the primal form, and these are the variables that will form the hyperplane of the optimal decision boundary.

In order for the SVM to work in nonlinear classification problems, it implicitly maps the actual non-linear feature space into a higher dimensional space where the classification problem becomes linearly separable. Mapping from low to high dimensional space is done by the SVM using what is called the kernel trick. This mapping is done implicitly, which reduces the number of computations (hence time, and need for memory) drastically. To wrap this up, SVM is linear classifier which works for linearly separable problems, and in case the problem is non-linear, the **kernel trick** is employed to implicitly map the feature space to a higher dimension space where the same non-linearly separable problem will become linearly separable. Hyperbolic tangent, Polynomial, and Radial Basis function RBF (Gaussian) are very common used kernels.

The RBF allows for implicit mapping to infinite dimensional space (incredible property) and allow SVM to classify in that space. Here we are simulating a mapping to an infinite dimensional space, and the optimal SVP classifier is derived in that space, and the dimensionality of space does not matter, and it will not affect the performance of the model as long as the **VC dimension** [12][13] is less than the dimensionality of the space. Gaussian kernel allows us to work and classify in an infinite dimensional space where in fact we are in low dimensional space, and of course it is impossible to explicitly map to an infinite dimensional space but we can do it implicitly with the kernel trick, and this is the beauty of it.

Designing new kernels is possible, provided that the new kernel function satisfies Mercer's conditions which state that this function must be continuous, symmetric, and positive semi-definite.

IV. EXPERIMENTS

Both SVM and CNN were implemented in Matlab. The SVM was fully implemented with toolbox, where CNN was implemented with a light weight library that I have created from scratch. A binary dataset was derived from the **EMNIST** dataset of handwritten characters, and only letters "**O**" and "**K**" were used, giving a total number of 2246 images, 1123 each making it a balanced dataset to avoid misleading results. The data split was 61% for training and 39% for testing for both models.

4.1 SVM Implementation

The SVM Matlab built in function, was used with the following parameters:

- **Kernel Type:** Gaussian
- **Kernel Scale:** Auto
- **Standardize:** true

Everything else was left at default.

4.2 CNN Implementation

The layers sequence of the architecture was as follows:

conv1 > max pool1 > conv2 > max pool2 > conv3 > ReLu activation function > 1 fully connected layer > softmax > cross entropy loss

- Details of CNN hyper-parameters:

Layer Type	Conv 1	Pool 1	Conv 2	Pool 2	Conv 3
Size of Filters	5×5	2×2	5×5	2×2	4×4
Depth of Filters	1	20	20	50	50
Number of Filters	20	N/A	50	N/A	500
Stride	1	2	1	2	1
Zero Padding	0	0	0	0	0

Table 1: CNN Hyper-parameters

- **Mini-batch size** (Stochastic gradient descent was used): 100
- **Number of training epochs**: 15
- **Learning rate**: 0.001
- **Momentum**: 0.9
- **Regularization factor** (to avoid overfitting): 0.0005

V. RESULTS

The misclassification error rates of the test samples for both models are:

- ✓ SVM: **2.32 %**
- ✓ CNN: **1.7 %**

It is very clear from the results that CNN has outperformed SVM even for binary classification problems. This is another work result that proves the high performance of CNN in image classification problems that cannot be outperformed by SVM which is one of the oldest and most popular approaches in machine learning.

VI. CONCLUSION

In this paper we have gone through the theory of SVMs and CNN in a concise and informative way. We have empirically proved that for the given binary image classification problem CNN outperforms SVM even for binary image recognition problems, and this promises that CNN is more likely to be the best and first algorithm to try with any image classification problem at hand. The main problem of CNN and any neural network based approach is that there is no universal robust theory that supports the selection of hyper-parameters, and this makes it hard to find the optimal hyper-parameters which give the best predictive power, knowing that hyper-parameters space is very large. This is unlike SVM whose theory is clear and robust. CNN remains an important candidate for machine learning problems, and can give the best results if tuned properly, and I am not claiming that the chosen hyper-parameters in this paper are the best for the used dataset. The error rate of 1.7% could be lowered even more if more tuning was done, and also training for more epochs.

REFERENCES

[1]. X.-X. Niu and C. Y. Suen, "A novel hybrid cnn-svm classifier for recognizing handwritten digits," Pattern Recognition, vol. 45, no. 4, pp. 1318-1325,2012.

[2]. S. Hong, T. You, S. Kwak, and B. Han, "Online tracking by learning discriminative saliency map with convolutional neural network," in International Conference on Machine Learning, pp. 597-606, 2015.

[3]. M. Elleuch, R. Maalej, and M. Kherallah, "A new design based-svm of the cnn classifier

architecture with dropout for o ine arabic handwritten recognition," Procedia Computer Science, vol. 80, pp. 1712-1723, 2016.

[4]. B. P. Toth, M. J. Toth, D. Papp, and G. Szucs, "Deep learning and svm classifcation for plant recognition in content-based large scale image retrieval.," in CLEF (Working Notes), pp. 569-578, 2016.

[5]. Y. LeCun, L. Bottou, Y. Bengio, and P. Ha ner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, 1998.

[6]. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in neural information processing systems, pp. 1097-1105, 2012.

[7]. T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in European conference on machine learning, pp. 137-142, Springer, 1998.

[8]. Y. Lin, F. Lv, S. Zhu, M. Yang, T. Cour, K. Yu, L. Cao, and T. Huang, "Large-scale image classification: fast feature extraction and svm train-ing," in Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, pp. 1689-1696, IEEE, 2011.

[9]. I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," Machine learning, vol. 46, no. 1-3, pp. 389-422, 2002.

[10]. C. Bahlmann, B. Haasdonk, and H. Burkhardt, "Online handwriting recognition with support vector machines-a kernel approach," in Frontiers in handwriting recognition, 2002. proceedings. eighth international workshop on, pp. 49-54, IEEE, 2002.

[11]. M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," IEEE Intelligent Systems and their applications, vol. 13, no. 4, pp. 18-28, 1998.

[12]. V. N. Vapnik and A. Y. Chervonenkis, "On the uniform convergence of relative frequencies of events to their probabilities," in Measures of com-plexity, pp. 11-30, Springer, 2015.

[13]. V. N. Vapnik and S. Kotz, Estimation of dependences based on empirical data, vol. 40. Springer-Verlag New York, 1982.

Abdelaziz Botalb "Performance Evaluation Of Support Vector Machines (Svms)And Convolutional Neural Network (Cnn) On Binary Classification Problem "International Journal of Engineering Research and Applications (IJERA) , vol. 8, no.9, 2018, pp 44-48