Table of Contents

[1. Introduction 1](#_Toc37959977)

[2. Methods 2](#_Toc37959978)

[3. Results 3](#_Toc37959979)

[3.1. Support Vector Machine 3](#_Toc37959980)

[3.1.1. MNIST Dataset 3](#_Toc37959981)

[3.1.2. CIFAR10 Dataset 5](#_Toc37959982)

[3.2. Artificial Neural Networks 5](#_Toc37959983)

[3.3. Convolutional Neural Networks 6](#_Toc37959984)

[4. Discussion 7](#_Toc37959985)

[5. References 7](#_Toc37959986)

# Introduction

In neuroscience the main question is how the brain acquires complex behaviours, such motor control, objects recognition, speaking, and etc. In computational neuroscience, in particular, we are interested in *models* which execute the kind of behaviours we see in the brain. This may come down, to some extend, to understanding two basic pieces, namely; (1) underlying network architectures and (2) learning rules, which are formally formulated as optimization problems.

In this project we take classification of MNIST and CFAR10 objects as the main task to show that how moving towards biologically plausible architecture helps solve this AI task with better performance, and vice versa, how backpropagation learning algorithm helps us understand the function of some feed back paths, which are in total ten times more than feedforward paths in the brain. This is an example of virtuous cycle, as opposed to vicious cycle, between neuroscience and artificial intelligence as a subset of computer science.

In this project we start to solve object classification task with support vector machine (SVM) algorithm as one of the best conventional machine learning (ML) algorithms. Then, we continue with modern ML approaches, i.e., artificial neural networks (ANNs). In this part, we implement shallow and deep neural networks with biologically plausible version of backpropagation, called feedback alignment. Finally, we end our investigation with applying convolutional neural networks (CNN) architecture, which is inspired by the biological visual system.

# Methods

Support Vector Machines (SVMs) are discriminative classifiers that can be formally defined as a separating hyperplane, meaning that given some labeled training data, the algorithm calculates an optimal hyperplane that categorizes new examples. In Python, we used scikit-learn to implement the SVM algorithm using the linear and Radial Basis Function (RBF) kernels by object creation, model fitting and prediction to measure its accuracy. We then optimized for the values of the hyperparameters C & gamma using GridSearchCV and created another optimized model with the optimal hyperparameters.

In artificial neural network, backpropagation is the latest learning algorithm that works in practice and shows up in virtually all of the state-of-the-art supervised, unsupervised, and reinforcement learning. In summary, the following formula holds for the updates of the upper layer’s weights.

There are the number of reasons why the backprop, in the form described above, is not biologically plausible.

1. It requires error term, or to put it in other words, it needs some predefined targets. This is not clear which part of the brain generates these target points.
2. The main problematic issue is about how the upper layers have access to the transpose of downstream weight matrix.
3. We also need the derivative of the activation function. This is important because the biological networks generate discrete spikes, but the ones we usually run on the computer are usually continuous.
4. And also, there is the issue of separate forward and backward passes. In the brain the forward and backward passes are pretty entangled, such that their activities influence one another. However, in backprop the backward path, where the weight updates are computed, has nothing to do with forward path.

Among the above-mentioned concerns, the one which particularly seems not to be biologically plausible is the second one. In this project, we will focus on an algorithm, called feedback alignment, which tries to deal with this issue which is known, in the literature, as weight transport problem.

A potential synaptic circuitry underlying feedback alignment is shown for a single hidden unit in Figure 1. As illustrated in this figure, the error is fed back through a second network (yellow) . In this case, a hidden unit with activity *hj* will update its incoming synaptic weights by three-factor formula , where is a the *j*-th element of modulator vector , is the pre-synaptic activity, is the a simple function of post-synaptic activity for the hidden unit with sigmoid nonlinearity.

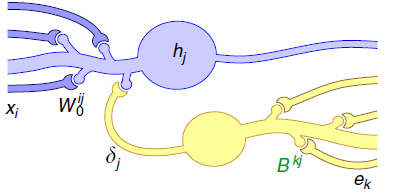


Figure 1: Feedback Alignment possible circuitry(Lillicrap, Cownden, Tweed, & Akerman, 2016)

The vector **,** is modulatory signal, as it does not impact forward path post-synaptic activity; however, it changes the plasticity of forward synapses. For the case of backpropagation, , where *e* is the error in the network output. Whereas, in feedback alignment algorithm, . The essence of feedback alignment algorithm is that feedback weights do not need to be exactly . In fact, any fixed random matrix is enough, as long as, on average, . Geometrically, this means that the teaching signal sent by the B matrix, , lies within 900 of the signal used by backprop, . That is, B pushes the network in roughly the same direction as backprop would.

For convolutional neural network we have used TensorFlow to implement the architecture. In order to avoid overfitting, we have employed dropout regularization method for convolution and dense layers. In order to improve the algorithm in terms of accuracy and speed we have taken advantage of batch normalization technique.

# Results

## Support Vector Machine

### MNIST Dataset

***Linear Kernel- MNIST***

Accuracy: 0.9404

Confusion Matrix:

[[ 957 0 4 1 1 6 9 1 0 1]

[ 0 1122 3 2 0 1 2 1 4 0]

[ 8 6 967 11 3 3 7 8 17 2]

[ 4 3 16 947 1 16 0 9 12 2]

[ 1 1 10 1 942 2 4 2 3 16]

[ 10 4 3 36 6 803 13 1 14 2]

[ 9 2 13 1 5 16 910 1 1 0]

[ 1 8 21 10 8 1 0 957 3 19]

[ 8 4 6 25 7 26 6 7 877 8]

[ 7 7 2 11 33 4 0 18 5 922]]

***RBF Kernel- MNIST***

Accuracy: 0.9792

Confusion Matrix:

[[ 973 0 1 0 0 2 1 1 2 0]

[ 0 1126 3 1 0 1 1 1 2 0]

[ 6 1 1006 2 1 0 2 7 6 1]

[ 0 0 2 995 0 2 0 5 5 1]

[ 0 0 5 0 961 0 3 0 2 11]

[ 2 0 0 9 0 871 4 1 4 1]

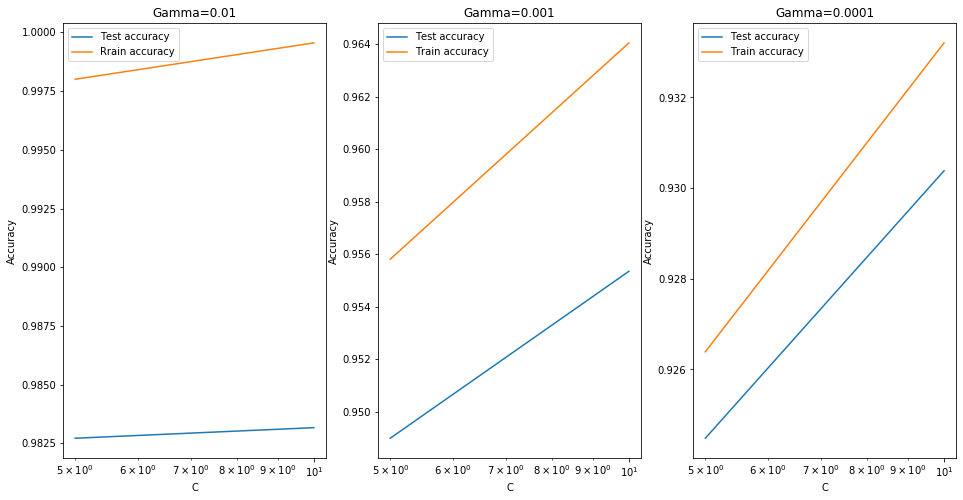
[ 6 2 0 0 2 3 944 0 1 0]

[ 0 6 11 1 1 0 0 996 2 11]

[ 3 0 2 6 3 2 2 3 950 3]

[ 3 4 1 7 10 2 1 7 4 970]]

***Optimizing for C & Gamma using RBF Kernel- MNIST***



The best test score is 0.9831666666666667 corresponding to hyperparameters {'C': 10, 'gamma': 0.01}

***Optimized model with the best hyperparameters on MNIST***

Accuracy 0.9576

Confusion Matrix:

[[ 970 0 1 1 0 4 1 1 2 0]

[ 0 1124 3 1 0 1 3 1 2 0]

[ 7 1 984 7 4 0 9 7 11 2]

[ 0 0 11 970 0 7 0 10 10 2]

[ 1 1 8 0 944 0 3 2 2 21]

[ 6 2 2 24 3 831 8 1 12 3]

[ 7 2 3 0 3 12 929 0 2 0]

[ 2 9 21 5 7 0 0 971 1 12]

[ 5 3 5 16 6 14 6 4 914 1]

[ 5 7 2 12 25 3 1 11 4 939]]

### CIFAR10 Dataset

***Linear Kernel- CFAR10***

Accuracy: 0.3753

Confusion Matrix:

[[480 40 75 40 28 34 24 43 169 67]

[ 81 433 45 48 27 46 37 37 66 180]

[103 48 306 99 107 89 104 78 40 26]

[ 57 65 130 257 71 177 127 44 34 38]

[ 59 35 176 88 273 96 135 97 24 17]

[ 57 52 128 195 96 291 63 60 34 24]

[ 31 38 103 137 113 90 432 31 10 15]

[ 73 62 101 80 114 87 34 373 24 52]

[174 74 35 36 18 42 13 16 509 83]

[103 208 26 34 25 25 44 59 77 399]]

***RBF Kernel- CFAR10***

Accuracy: 0.5436

Confusion Matrix:

[[622 23 57 17 21 15 20 28 144 53]

[ 32 647 21 42 6 17 16 20 49 150]

[ 85 24 407 89 133 62 112 50 23 15]

[ 32 30 91 386 55 167 121 42 25 51]

[ 47 14 161 68 435 44 125 68 22 16]

[ 22 14 87 197 71 431 84 49 21 24]

[ 11 18 64 84 95 46 641 14 13 14]

[ 36 22 50 76 74 67 34 567 16 58]

[ 83 66 16 22 20 19 15 18 688 53]

[ 44 151 11 40 10 14 28 38 52 612]]

***Optimized model on CFAR10***

Accuracy 0.53

Confusion Matrix:

[[604 23 61 22 26 12 23 33 146 50]

[ 44 624 21 34 9 20 21 26 53 148]

[ 86 26 420 82 127 60 111 58 18 12]

[ 32 32 106 379 60 153 125 41 26 46]

[ 44 11 158 61 444 53 121 70 25 13]

[ 28 19 107 195 80 404 77 50 21 19]

[ 10 16 82 93 107 38 614 17 14 9]

[ 45 20 61 76 87 70 29 551 16 45]

[ 83 58 19 27 24 23 10 12 689 55]

[ 48 172 10 34 17 17 31 41 59 571]]

## Artificial Neural Networks

First, we implemented backprop algorithm to find proper hyperparameters (Table 1). For CFAR10 dataset, we have changed learning rate several times till finding acceptable learning rate. Other parameters were found with less trial and error. The results of our search are shown in the code.

Table 1: ANN hyperparameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Learning Rate | Weight Decay | Epochs | Batch Size |
| MNIST | 0.5 | 0. | 400 | 128 |
| CFAR10 | 0.1 | 0. | 400 | 128 |

We used the same hyperparameters for feedback alignment as for backpropagation. Although the weights of feedback alignment matrices were just some random numbers, this algorithm could reach acceptable level of accuracy. However, as is it shown in the plots of the code file, the feedback alignment is slower than backpropagation and its test accuracy curve is noisy and has much fluctuations. The best accuracies achieved by ANNs are shown in Table 2.

Table 2: Backprop and Feedback Alignment accuracies

|  |  |  |
| --- | --- | --- |
|  | MNIST | CFAR10 |
| Back Propagation accuracy (%) | 98.2 | 54 |
| Feedback Alignment accuracy (%) | 97.8 | 50.9 |

As discussed in Methods section, the angle between and should be less than 900. However, we do not have control over this angle in the first place, because we define B matrix elements randomly. However, the interesting observation is that as the iterations go, the angle between these two vectors decreases and become less than 900. This can be seen in the plots labeled .

We have also implemented feedback alignment on deeper networks. Feedback alignment works quite well on deeper network for MNIST. However, the final accuracy (95.9%) in less than the one achieved by shallow network (97.8%) after 400 epochs. The justification may be that the amount of randomness increases as we make the network deeper. We would probably get the same amount of accuracy as shallow network with feedback alignment if we run the model for more epochs. For deep ANN with back prop, we expected to get higher accuracy on CFAR10. However, after 100 epochs the level of accuracy turned out to be the same in deep and shallow network.

## Convolutional Neural Networks

Finally, we implemented AlexNet-like CNN architecture on CFAR10 (Krizhevsky, Sutskever, & Hinton, 2012) . We have reached to 88.9% accuracy by employing this model. We had to define dropout regularization method, for every layer of the network. Otherwise, the model would overfit. The other key factor in our network is using batch normalization. In this technique, each batch is normalized such that it has zero mean and standard deviation of one (Ioffe, Szegedy, 2015). Surprisingly, it has accelerated network training and ended up increasing the accuracy by 10%. We have applied the same technique of normalization for input data. Although not affecting the training speed and accuracy, input normalization has decreased the fluctuations in the test accuracy curve vs. epochs and made it smooth.

# Discussion

For the case of SVM, for MNIST dataset, we notice that although the simpler linear kernel has a slightly lower accuracy than the RBF kernel, however both of them display a very high degree of accuracy overall of 94% & 97%. When optimizing for the hyperparameters of C and gamma, we note that the accuracy increases with increasing C and gamma. When it comes to the more difficult CIFAR10 dataset, we find a significant difference between the accuracy of the linear kernel of 37%, significantly lower than the RBF kernel that has managed 54%. Optimizing the hyperparameters has also yielded similar results.

Using artificial neural networks with backpropagation resulted in the same level of accuracy as SVM, i.e., 98% for MNIST and 54% for CFAR10. In this project we implemented the feedback alignment algorithm which relaxes the weight transport constraint of the backprop to make it more biologically sensible. Probably, in he brain a separate backward network have the responsibility of generating teaching modulatory signals. Assumption of independence of forward and backward network, does not seem biologically plausible. These two networks definitely interact with and regulate the function of one another. However, it is quite interesting that, even when we look at these networks separately and consider totally random synaptic weights for backward network, still we can get acceptable results for image classification.

Getting more inspiration from biology, i.e., vision system, led to proposing convolutional nets architecture. The result of implementing of this architecture on CFAR10 was outstanding (88% accuracy). As opposed to the previous part, in which thinking about biological plausibility of a mathematical learning algorithm, has delivered insights about the function of some backward paths in the brain, in this part, the biological vision system helped the scientists to come up with an architecture that has by far better performance than other conventional and modern machine learning algorithms.

In this project we try to show the bilateral relationship between neuroscience and computer science, which is, as far as we have perceived, the main domain of computational neuroscience. There are many interesting examples of this kind. For example, how computational models of reinforcement learning helped neuroscientist to unlock the mechanism of dopamine release in the brain by helping them understand that instead of reward, what is important in dopamine release is how unexpected and surprising that reward is. However, here we try to show this constructive relationship, in the context of visual system and object classification.

# References

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Paper presented at the *Advances in Neural Information Processing Systems 25,* 1097–1105.

Lillicrap, T. P., Cownden, D., Tweed, D. B., & Akerman, C. J. (2016). Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications, 7*(1), 13276. doi:10.1038/ncomms13276

Ioffe, S., Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.[arXiv:1502.03167](https://arxiv.org/abs/1502.03167)