Comparison between ANN and SVM in Image Classification

김예지(2017010583) 김가영(2017020605) 김서현(2016020110)

Korea University

**Abstract** – In this paper the performance of the artificial neural network (ANN) is compared to the performance of a statistical method such as the support vector machine (SVM). This comparison is made with respect to an image classification application. The artificial neural network and the support vector machine are trained to classify images with feature vectors. Analysis was done by using many different settings for each learning method. For each classification technique, accuracy was positively related with the value of parameters for the SVM and the number of hidden units and training sets for the ANN. In addition, both classifiers are also relative to the preprocessing. In general, the most accurate classifications were derived from the ANN approach. ANN classification was more accurate (54%) than that derived from the SVM (47.2%). A Comparative analysis is made between the ANN and SVM which differed in the ability to correctly label individual images by their experiment settings.

1. Introduction

To get an answer of the question "what is this?" from the computer, ​computers should be taught so that they can judge themselves. Pattern recognition is the way to solve this problem. ​There are various methodologies for classification. ​The most representative examples are ANN and SVM. ​Therefore, we implemented ANN and SVM algorithms to classify about 10 different categories of images. And then We are working on a project to measure the performance of these three algorithms. ​This project aims to understand the basic classification methods of pattern recognition.

An artificial neural network is a paralled distributed processor [1] that has a natural tendency for storing experiential knowledge. They can provide solutions for problems, which are generally characterized by non-linear ties, high dimensionality noisy, complex, imprecise, and imperfect or error prone sensor data, and lack of a clearly stated mathematical solution or algorithm [2]. So, it has been useful for decades to the development of image classification algorithms [3].

Support vector Machines have been recently introduced in the statistical learning theory domain [9] for regression and classification problems, and applied to the classification of multispectral and hyperspectral mages [10]. The technique consists in finding the optimal surface between classes thanks to the identification of the most representative training samples of the side of the class. This kind of classifier reveals very interesting properties for hyperspectral image processing and it may perform class separation even with means very closed to each other with a small number of training samples.

However, many different kinds of neural networks exist currently and it is obvious those networks show better performances than the ANN or the SVM does [11]. In this paper, we are aim to maximize the ability of the ANN with single layer and the SVM with different kernels. Since calculation of deep neural network takes so long due to high computational complexity, we are to make a discussion on single layer ANN and the SVM which have a relatively low computational work.

1. Model

We used two machine learning algorithms ANN and SVM to classify 10 categories of images. Here are the basic theories of ANN and SVM

* 1. ANN

Neural Networks are modeled as collections of neurons that are connected in an acyclic graph. In other words, the outputs of some neurons can become inputs to other neurons. But cycles are not allowed since that would imply an infinite loop in the forward pass of a network. Neural Network models are often organized into distinct layers of neurons. For regular neural networks, the most common layer type is the fully-connected layer in which neurons between two adjacent layers are fully pairwise connected, but neurons within a single layer share no connections.

**[Figure 1]** are two example neural network topologies that use a stack of fully-connected layers. A left neural network has one hidden layer of 4 neurons and one output layer with 2 neurons. A right network has two hidden layer of 4 neurons and one output layer with 1 neuron. In this project, we are using only single hidden layer artificial neural network architecture like left to exam limitation of it. It has input layer which has 3072 neurons and output layer which has 10 neurons because of CIFAR-10 dataset. So, we have [3072K] + [K10] learnable parameter (K: the number of hidden layer neurons). These weights are commonly adjusted by backpropagation.

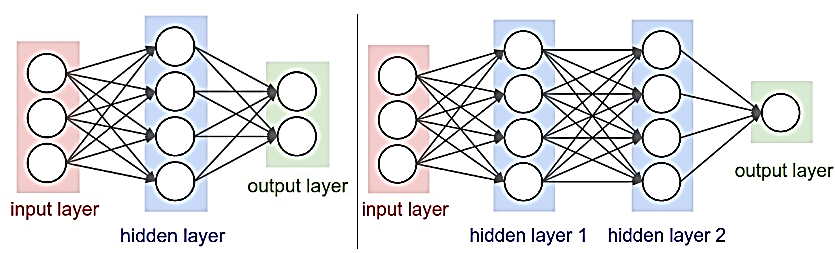


Figure 1 Neural network single-layer (left)/multi-layer (right)

In fact,the area of Neural Networks has originally been primarily inspired by the goal of modeling biological neural systems, but has since diverged and become a matter of engineering and achieving good results in Machine Learning tasks. The [**Figure 2]** shows a cartoon drawing of a biological neuron and a common mathematical model. The signals () that travel along the axons interact multiplicatively () with the dendrites of the other neuron based on the synaptic strength at that synapse (). The idea is that the synaptic strengths (the weights ) are learnable and control the strength of influence.

In the basic model, the dendrites carry the signal to the cell body where they all get summed. If the combined incoming signals are strong enough, the neuron can fire, sending a spike along its axon. Based on rate code interpretation, we can model the firing rate of neuron with an activation function.

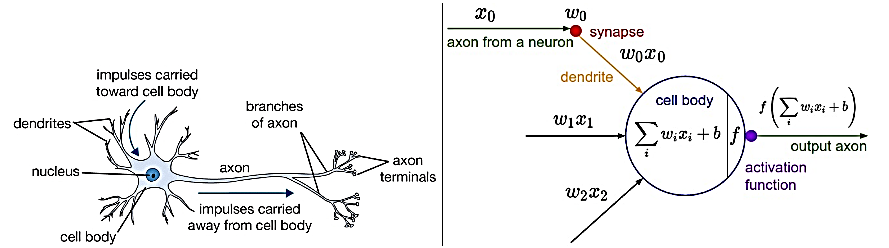


Figure 2 Drawing of biological neural (left) / mathematical model

**2.1.1. Backpropagation**

The goal of supervised learning algorithms is to find a function that best maps a set of inputs to its correct output. The Backpropagation can learn the appropriate internal representations to allow neural network to learn any arbitrary mapping of input to output. So the backpropagation is commonly used as a part of algorithms that optimize the performance of the network by adjusting the weights. It is a method to calculate the gradient of the loss function with respect to the weights in an artificial neural network.

When an input vector is presented to the network, it is propagated forward through the network, layer by layer. The output of the network is then compared to the desired output and an error value is calculated using loss function. Backpropagation uses these error values to calculate the gradient of the loss function. This gradient is fed to the optimization method, which in turn uses it to update the weights, in an attempt to minimize the loss function.

**2.1.2 Activation function**

There are several commonly used activation functions such as sigmoid, tanh, ReLU, and softmax. The **sigmoid** has the mathematical form . Sigmoid takes a real-valued number and squashes it into range between 0 and 1. In particular, large negative numbers become – and large positive numbers become 1. But it saturates and kills gradients and outputs are not zero-centered.

The **tanh** has mathematical form . It squashes a real-valued number to the range [-1,1]. Like the sigmoid neuron, it activations saturate, but unlike the sigmoid neuron its output is zero-centered.

The **ReLU** computes the function . In other words, the activation is simply thresholded at zero. The ReLU was found to greatly accelerate the convergence of stochastic gradient descent compared to sigmoid/tanh functions. Also it can be implemented by simply thresholding a matrix of activation at zero when compared to tanh and sigmoid that involve expensive operations. Unfortunately, ReLU units can be fragile during training and can die. For example, a large gradient flowing through a ReLU neuron could cause the weights to update in such a way that the neuron will never activate on any data point again. With a proper setting of the learning rate this is less frequently an issue.

The **softmax** has the mathematical form for j=1,…,k. The softmax squashes a K-dimensional vector z of arbitrary real values to K-dimensional vector of real values in the range (0,1). So, the output of softmax function can be used to represent a probability distribution over K different possible outcomes. The softmax function used in various multiclass classification methods, such as multinomial logistic regression and artificial neural network

* 1. SVM

SVMs are very attractive for the classification of remotely sensed data. This approach seeks to find the optimal separating hyperplane between classed by focusing on the training cases that lie at the edge of the class distributions, the support vectors, with the other training cases effectively discarded [5].

SVM differ radically from Neural Networks in that SVM training always finds a global minimum, the main difference between ANN and SVM is the principal of risk minimization. In case of SVM, structural risk minimization principle is applied by minimizing an upper bound on the expected risk whereas in ANN, traditional empirical risk minimization is used minimizing the error on the training data. The difference in risk minimization is to improve the generalization performance of SVM compared to ANN.

There are two main limitations in ANN which are solved by SVM. For ANN, there is no theoretic relation between the ANN structure and the classification task. However, SVM have a relationship between the structure (the support vectors) and the classification tasks. Also, SVM optimize the separation surfaces between two classes when ANN derive hyperplanes separation surfaces, in feature representation space, which are not optimal in terms of margin between examples of two different classes.

The complete mathematical formulation of SVM can be found in [6]. We just present a brief description of the classification process.

**2.2.1 SVM basis**

The basic nature of classification with an SVM can be illustrated most easily for the simple situation in which there are two linearly separable classes in D-dimensional space. Using L training points, where each input has D attributes (i.e. is of dimensionality D) and is one of two classes =-1 or +1.

,

Many hyperplanes could be fitted to separate the classes, but there is only one optimal separating hyperplane, which is expected to generalize well in comparison to other hyperplanes. This optimal hyperplane should run between two classes, with all cases of a class located to one side of the separating hyperplane, which is itself located such that the distance to the closest training data points in both of the classes is as large as possible. This hyperplane can be described by **w∙x +** b = 0 where **w** is normal to hyperplane and is the perpendicular distance from the hyperplane to the origin.

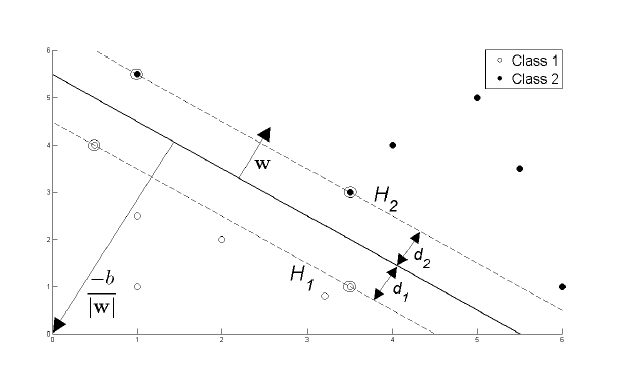


Figure 3 Hyperplane through two linearly separable classes

The support vectors lie on two hyperplanes, which are parallel to the optimal hyper plane, of equation:

**w∙x +** b

**w∙x +** b

The maximization of the margin with the equation of the two support vector hyperplanes leads to the following constrained optimization problem.

**2.2.2 Non-linear separability**

Commonly the classes are not linearly separable, and the constraint of cannot be satisfied. To deal with such cases using only linear separating boundaries, a new set of variables, sometimes referred to as slack variables that indicate the distance the case is from the optimal hyperplane and so the amount of violation of the constraints may be introduced.

The constraint then becomes

The above constraint, in case of outliers, can always be met by making very large, so a penalty term is added to penalize solution for which are very large. The constant *C* controls the magnitude of the penalty associated with training samples that lie on the wrong side of the decision boundary. With a low value of *C*, an inappropriately large fraction of support vectors may be derived, while with a large value of *C*, there is a danger of the SVM overfitting to training data and so having low generalization ability. With the addition of the penalty term, the optimization problem becomes

under the constraint of . In this, the first part of the term aims to maximize the margin, while the second part seeks to penalize the cases located on the incorrect side of the decision boundary with *C* controlling the relative balance of these two competing objectives [7]. If the classes overlap considerably in the feature space, then can be very large, and the hyperplane may not generalize well.

**2.2.3 Kernel-based SVM**

The basic approach outlined above may be extended to allow for nonlinear decision surfaces. For example, the input data may be mapped into a high-dimensional space through some nonlinear mapping that has the effect of spreading the distribution of the data points in a way that facilitates the fitting of a linear hyperplane. Specifically, the training data may be projected into a high-dimensional space **H**.

=

is an example of a family of function called *Kernel Functions*(= being known as a Linear Kernel). The set of kernel functions is composed of variants based on calculating inner products of two vectors. This means that if the functions can be recast into a higher dimensionality space by some potentially non-linear feature mapping function **x**, only inner products of the mapped inputs in the feature space need be determined without us needing to explicitly calculate

The reason that this *Kernel Trick* is useful is that there are many classification problems that are not linearly separable in the space of the inputs **x**, which might be in a higher dimensionality feature space given a suitable mapping **x**.

If we define our kernel to be :

Then a data set that is not linearly separable in the two dimensional data space **x** is separable in the nonlinear feature space defined implicitly by this nonlinear kernel function known as a *Radial Basis Kernel*.

Other popular kernels for classification are the *Polynomial Kernel*

and the *Sigmoidal Kernel*

where and b are parameter defining the kernel’s behavior.

3. Experiment

**3.1 Dataset**



Figure 4 CIFAR-10

The CIFAR-10 dataset consists of 60000 images in 10 classes. As seen above, 10 random images are in the dataset for each class. Classes consist of ‘plane’, ‘car’, ‘bird’, ‘cat’, ‘deer’, ‘dog’, ‘frog’, ‘horse’, ‘ship’, ‘truck’. The dataset is divided into five training batches and one test batch, each with 10000 images. Each image has 32 We used single training batch and 1000 samples from the test batch. Also, if the individual features do not more or less look like standard normally distributed data, learning algorithm might behave badly. So preprocessing was done that all features are centered around 0 and have invariance in the same order.

**3.2 Experiment Settings**

We tested our algorithms to examine the performances of the ANN and the SVM using CIFAR-10 dataset.

**3.2.1 ANN settings**

In our project, we have two ANN python codes (Our code and Reference code). We not only initialized weight vectors arrays and network architecture but also updated weight using activation function and derivation of loss function. We did not use machine learning python library such as scikit-learn [8] and Chainer [13] except math library like log.

In our code, we used network that has one input layer which has 32323 neurons, one hidden layer, and one output layer which has 10 neurons because of CIFAR-10 10 class. So we made two-dimensional arrays for weight vectors and these were randomly initialized, range [0,1]. Also, We fixed learning rate 0.001 and initialized bias randomly. And we experimented 100, 200, and 300 hidden units and made several activation function, softmax and sigmoid to compare each accuracy. In order to overcome very large dataset, we used mini-batch (=200) and 20 epoch. But we could not get high performance and accuracy in multi-class classification using our code.

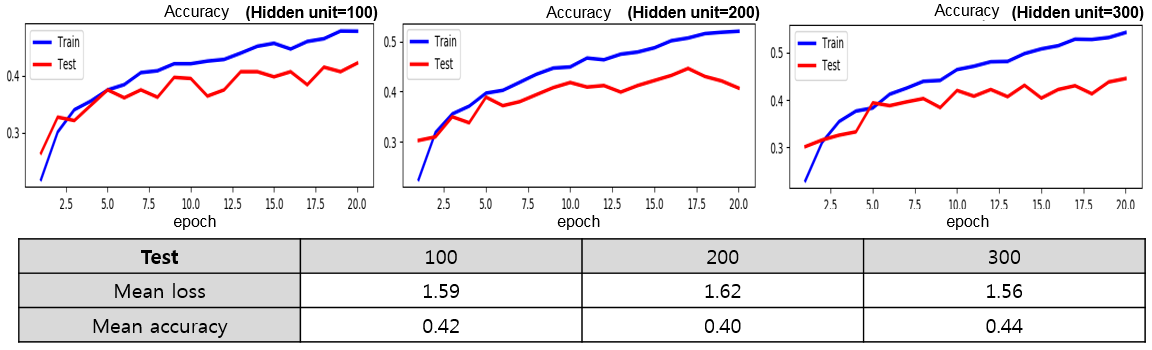


Figure 5 ANN result using 10000 train images (without data normalization)

So, we test a reference code which is using Chainer python library to compare our code with reference code. In reference code, we used neural network which has one input layer, one hidden layer, and one output layer and only use ReLU activation function. We experimented 10000 or 50000 train images and 1000 test images using 100, 200, and 300 hidden units as same as our code. Also we used mini-batch (=200) and 20 epoch to overcome the problem of very large dataset. But they use data mean normalization and data augmentation. Also they initialized network weights with i.i.d Gaussian samples, each of which has zero mean and deviation.

**3.2.2 SVM settings**

4 different kernel functions were used to compare performance of SVM: linear, polynomial, RBF, sigmoid. Parameter *C* denotes penalty which was state in section 2.2.2. Gamma ( is coefficient of Kernel for ‘rbf’, ‘poly’ and ‘sigmoid’. 1/n\_features is normally used for gamma. We set C as 1.0 and gamma as 0.00033 and 0.01 (1/3072. Gamma was set differently in each kernel for the best performance. Before the algorithm, preprocessing was done. The tool used for SVM experiment including preprocessing, learning and accuracy was Scikit-learn.

**3.3 Result**

Fundamental to this work is the comparison of classification accuracy statements. The evaluation and comparison of classifications is plagued with problems. Classification accuracy is commonly expressed using a metric computed from the error or confusion matrix using the testing set and estimates for different classification compared to indicate the significance of differences in the classification outputs. This section describes the implemented ANN and SVM image classification systems respectively and presents their results. The two classifiers, differed markedly in their basis for class allocation and expected dependency on training set size.

**3.3.1 ANN result**

We estimated ANN performance and accuracy using reference code. **[Figure 5]** shows the ANN result using 10000 train images and 1000 test images due to time limitation. And, it did not use data normalization and data augmentation. As seen in below, the number of hidden units does not seem to have much effect.



Figure 6 ANN result using 50000 train images (without data normalization)

But when we increased the number of train images and the results were different. **[Figure 6]** shows the ANN result using 50000 train images and 1000 test images. As you can see, the number of train images effects the result. In this case, the larger the number of hidden units the better the results.

**[Figure 7]** shows the ANN result with data normalization. It also uses 50000 train images and 1000 test images. As seen in below, the results with data normalization are better.

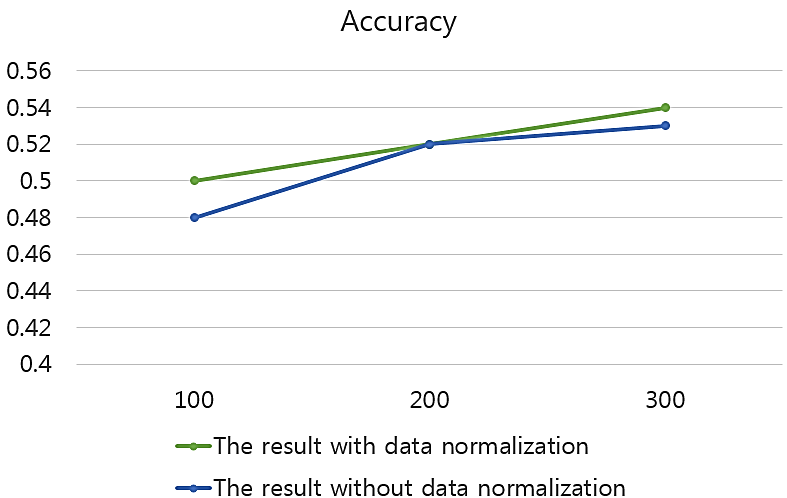
****

Figure 7 ANN result about data normalization

In our experiments, we can get the best performance of ANN with 300 hidden units and data normalization. As a result, we can get about 60% accuracy with 50000 train, 10000 test images, normalization dataset, and 6000 hidden units.

**3.3.2 SVM result**

The sensitivity of the accuracy of the SVM classifications to the parameters indicates the need for finding optimal values. The *C* parameter defines how far the influence of a single training example reaches, with low values meaning ‘far’ and high values meaning ‘close’. The *C* parameter trades off misclassification of training examples against simplicity of the decision surface. A low *C* makes the decision smooth, while a high *C* aims at classifying all training examples correctly by giving model freedom to select more samples as support vectors. We set *C* as 1.0 which is a default value to prevent overfitting. The behavior of the model is very sensitive to the gamma parameter. If gamma is too large, the radius of the area of influence of the support vectors only includes the support vector itself and no amount of regularization with *C* will be able to prevent overfitting. When gamma is very small, the model is too constrained and cannot capture the complexity or “shape” of the data. The region of influence of any selected support vector would include the whole training set. The resulting model will behave similarly to a linear model with a set of hyperplanes that separate the centers of high density of any pair of two classes. This can be seen explicitly in Poly kernel since it shows lower accuracy which was near to the result of linear SVM in low value of gamma.

Training results are also sensitive to some specific circumstances when it comes to preprocessing. Linear and Poly kernel were still able to make the results when not using preprocessing. However, Rbf and sigmoid kernel showed no meaningful accuracies which made no difference in random choice.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kernel | Linear  () | Poly  (  Degree = 3) | Rbf  () | Sigmoid  () |
| accuracy | 0.316 | 0.421 | 0.472 | 0.219 |

Table 1 Results of SVM

As seen in the Table 1, Rbf kernel showed the best performance as we have expected. Also, Poly kernel took the second place in the performances. Sigmoid kernel displayed result comparing to the linear SVM. For kernel function, the accuracy of the classification may vary with the parameter. Poly kernel performs well on the value of 0.001, whereas the other kernels were better on which is near to the value of 1/3072(3072 is the number of features).

Preprocessing also plays a big role in the results. Whether using preprocessing or not showed big differences in performances especially for Rbf and Sigmoid kernel. When preprocessing was not applied, accuracies of the both kernel were at most 0.1.

With all four classification methods, it was apparent that classification accuracy was positively related to the value of the parameters.

4. Discussion

Generally, the ANN presents better performance in image classification than SVM. Both classifiers showed different results in performances when preprocessing or the value of parameters were changed. However, for SVM showed no difference when used larger training sets.

For the SVM, the result of the algorithms got better when kernel function was applied. We expect this performance can get better when kernel function is more complicated or when we choose optimal parameter value for the kernel function. When it comes to four common kernel functions, however, its performance is not able to get better than neural network when used with single kernel function.

Although Single layer Neural network made almost over half of accuracy in labeling images of test data, its performance is still lower than other deep neural networks. The disadvantages of single layer ANN is in with large dimensional input data. Its size is vulnerable to overfitting, so Convolutional Neural Network is largely used to prevent overfitting by extracting features. This is presented in [12], using single layer CNN with accuracy of 79%. We made the conclusion that a low computational work has a limitation to overcome complicate multi-layer works.

References

1. Haykin, Simon. "Neural Networks: A Comprehensive Foundation: Macmillan College Publishing Company." *New York* (1994).
2. Seetha, M., et al. "ARTIFICIAL NEURAL NETWORKS AND OTHER METHODS OF IMAGE CLASSIFICATION." Journal of Theoretical & Applied Information Technology 4.11 (2008).
3. Mahmon, Nur Anis, and Norsuzila Ya'Acob. "A review on classification of satellite image using artificial neural network (ANN)." *Control and System Graduate Research Colloquium (ICSGRC), 2014 IEEE 5th*. IEEE, 2014.
4. CS231n, <http://cs231n.github.io/neural-networks-1/>
5. Mercier, Grégoire, and Marc Lennon. "Support vector machines for hyperspectral image classification with spectral-based kernels." *Geoscience and Remote Sensing Symposium, 2003. IGARSS'03. Proceedings. 2003 IEEE International*. Vol. 1. IEEE, 2003.
6. Burges, Christopher JC. "A tutorial on support vector machines for pattern recognition." *Data mining and knowledge discovery* 2.2 (1998): 121-167.
7. Belousov, A. I., S. A. Verzakov, and J. Von Frese. "A flexible classification approach with optimal generalisation performance: support vector machines." *Chemometrics and intelligent laboratory systems* 64.1 (2002): 15-25.
8. <http://scikit-learn.org/stable/>
9. Vapnik, V. "Statistical Learning Theory John Wiley and Sons Inc." *New York* (1998).
10. Melgani, Farid, and Lorenzo Bruzzone. "Support vector machines for classification of hyperspectral remote-sensing images." *Geoscience and Remote Sensing Symposium, 2002. IGARSS'02. 2002 IEEE International*. Vol. 1. IEEE, 2002.
11. <http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html#43494641522d3130>
12. Coates, Adam, Andrew Ng, and Honglak Lee. "An analysis of single-layer networks in unsupervised feature learning." *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 2011
13. https://chainer.org/