MLP Coursework 4: Self-paced Learning

G17: s1702319, s1646851, s1752655

Abstract

Self-paced learning (SPL) is a recently proposed learning strategy which can effectively avoid bad local optimum in neural networks. Inspired by the learning process of humans and animals, the technique literately incorporates easy to more complex samples into training. One of the advantages of SPL is that it can build different self-paced regularization term for diverse applications in the field of computer version and patter recognition, which can be used to define easy examples. For instance, traditional SPL regards easy examples as those which have smaller training loss or higher likelihood. Therefore, in this project, we aim at applying self-paced learning technique to image classification task on the CIFAR-10 dataset and find out which self-paced regularization terms outperform others for this particular task. Experimental results have shown that the use of SPL technique evidently improves the performance of our 2-layer convolutional neural network, with the classification accuracy on CIFAR-10 climbing up to 84.1% , more than 2 percent higher than that of the baseline. Moreover, it has been demonstrated that SPL with Linear Soft Weighting (LinSW) outperforms other regularization schemes on this CIFAR-10 image classification task.

1. Introduction

Humans and animals start learning from simple and basic concepts and then gradually learn complex ones in an organized way. Many researchers focus on developing machine learning strategies according to human study models and intend to figure out whether this kind of strategy is able to perform better in neural networks. The idea that training a learning machine using similar strategy could go back to Elman (1993), which emphasized the importance of starting small. He states that training should start with an easier task, then gradually improve the difficulty and complexity, just similar to human trained with a curriculum. Then, Bengio et al. (2009) coined the term *Curriculum learning* which aimed at studying further on starting small strategy in 2009. Furthermore, curriculum learning is committed to breaking through the limitations of deep neural networks: non-convex optimization problems (help to find a better local minimum and speed up the convergence of training towards the global minimum).

One of the key questions in this strategy is how to identify easy tasks and complex tasks. Faced with this challenge, Kumar et al. (2010) presented an idea called *self-paced learning*(SPL) in which the introduction of the term regularization helps the model learn both easy and complex samples. Generally speaking, traditional SPL regards easy examples as those which have smaller training loss or higher likelihood. Compared with curriculum learning, self-paced learning has two advantages. Firstly, it combines the learning objectives with the curriculum so that the curriculum and the learned model are united in the same optimization problem. Secondly, it separates the regularization term from the loss of functions of specific problems. It has been demonstrated that the self-paced learning technique outperforms the state of the art methods in object localization, noun phrase coreference, motif finding and handwritten digit recognition (Kumar et al., 2010).

In this project we aim at exploring self-paced learning technique on the CIFAR-10 dataset and find out which self-paced regularization terms that are used to define easy examples outperform others for this particular dataset. Therefore, we'd like to run experiments using four different regularization terms and compare the classification accuracy.

2. Methodology

To address the issue of CL which we have discussed in coursework 3, Koller's group (Kumar et al., 2010) designed a new formulation, called self-paced learning (SPL).

2.1. Self-Paced Learning

The basic idea of the SPL is to design curriculum as a regularization term and add it to loss function. Therefore, the method of SPL has two main benefits: first, this method both optimizes the model and the curriculum at the same time; second, the regularization term cannot affect the optimization of weights in this model, because the self-paced regularization term is independent with the loss function.

The authors claim that given training set $\mathcal{D} = (\mathbf{x}_i, y_i), ..., (\mathbf{x}_n, y_n)$, the number of samples is n. Each $\mathbf{x}_i \in R^m$ represents a sample and y_i represents the classification label for the corresponding sample. The parameters $\mathbf{w}$ represent the optimization parameters of the corresponding model and its regularization item is $r(\mathbf{w})$. Then, They denote loss function as $L(y_i, g(x_i, \mathbf{w}))$. So, an objective function of traditional machine learning algorithm can be

denoted as:

$$\mathbf{w}_{t+1} = argmin(r(\mathbf{w}) + \sum_{i=1}^{n} L(y_i, g(x_i, \mathbf{w}))) \tag{1}$$

The core idea of self-paced learning is to select samples with small training errors and high likelihood values from all samples in each iteration, and then, update the model parameters(Kumar et al., 2010). In each iteration, choosing the number of samples is determined by weighting parameters which can import more samples by decreasing itself. At last, it stops the iteration when all samples have been selected, or when the loss function is unable to decrease. Therefore, self-paced learning imports binary variable $\mathbf{v} = [v_1, ..., v_n]^T$ into traditional machine learning objective functions to represent whether each sample is selected, or whether it is a simple sample. For example, a simple sample is a sample that is high in likelihood or far away from the classification boundary. So, only the samples with $v_i = 1$ are included in the objective function calculation, and the objective function can be changed to:

$$(\mathbf{w}_{t+1}, \mathbf{v}_{t+1}) = argmin(r(\mathbf{w}) + \sum_{i=1}^{n} v_i L(y_i, g(x_i, \mathbf{w})) - \lambda \sum_{i=1}^{n} v_i) \tag{2}$$

In this equation, λ represents the parameter of self-paced learning pace which is used to decide which samples can be selected for training. If the λ value is small, the objective function optimization process tends to choose a sample with a small loss value. As the number of iterations increases, they gradually increase the value of λ. As λ values approaches large, more samples are selected. At this point, the self-paced learning training process is degenerated into the traditional machine learning training process. Therefore, the process of self-paced learning can be understood as starting with a small number of "simple" samples and gradually introducing more samples until all samples are selected into the model.

The Eq.2 indicates the loss of a sample is discounted by a weight. The objective of SPL is to minimize the weighted training loss together with the negative l1-norm regularizer $-\|\mathbf{v}\|_1 = -\sum_{i=1}^{n} v_i$(Jiang et al., 2015).

This loss function, however, has two kinds of weights ($\mathbf{w}$ and $\mathbf{v}$) which are dependent between each other. Therefore, the EM Algorithm is used to help minimize loss function. We briefly describe this well-known algorithms below:

EM Algorithm for Likelihood Maximization. The objective is to maxminize likelihood:

$$\max_{\mathbf{w}} \sum_{i} \log Pr(\mathbf{x}_i, \mathbf{y}_i; \mathbf{w}) = \max_{\mathbf{w}} (\sum_{i} \log Pr(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}_i; \mathbf{w}) - \sum_{i} \log Pr(\mathbf{h}_i | \mathbf{x}_i, \mathbf{y}_i; \mathbf{w})) \tag{3}$$

A common method to maximize this likelihood is to use the EM algorithm. Introduced in algorithm 1, EM iterates between searching the optimized value of the latent variables $\mathbf{h}$ and maximizing the objective 3 subject to this expectation in each epoch (Kumar et al., 2010).

Algorithm 1 The EM algorithm for parameter estimation by likelihood maximization.

input $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_n, \mathbf{y}_n)\}, \mathbf{w}_0, \epsilon$

$t \leftarrow 0$
repeat
  Obtain the expectation of objective 3 under the distribution $Pr(\mathbf{h}_i | \mathbf{x}_i, \mathbf{y}_i; \mathbf{w})$
  Update $\mathbf{w}_{t+1}$ by maximizing he expectation of objective 3. Specifically, $\mathbf{w}_{t+1} = argmax_w \sum_i Pr(\mathbf{h}_i | \mathbf{x}_i, \mathbf{y}_i; \mathbf{w}) \log Pr(\mathbf{x}_i, \mathbf{y}_i, (\mathbf{h}_i; \mathbf{w})$
  $t \leftarrow t + 1$
until Objective function cannot be increased above tolerance ϵ.

Applying EM algorithm on objective function of SPL 2, the parameters are optimized iteratively by two steps in each epoch. EM updates firstly $\mathbf{v}$ with fixed $\mathbf{w}$ and then optimizes the weight $\mathbf{w}$ with expected $\mathbf{v}$.

Algorithm 2 Self-spaced learning

input training sets $\mathcal{D}$, $paceparameter\mu > 1$

initialize w^*, λ_0
while not converged **do**
  while not converged **do**
    Update $\mathbf{v}^* = argmin_{\mathbf{v}} E(\mathbf{w}^*, \mathbf{v})$
    Update $\mathbf{w}^* = argmin_{\mathbf{w}} E(\mathbf{w}, \mathbf{v}^*)$
  end while
  $\lambda \leftarrow \mu\lambda$
end while
return $w = w^*$

As described above, the algorithm 2 introduces pseudocode of self-spaced learning. In this algorithm, we need to initialize model parameters $\mathbf{w}$, and then update latent weights variable $\mathbf{v}$ as well as model parameters $\mathbf{w}$. To update $\mathbf{v}$, we can rewrite the Eq.2 as Eq.6 because of the fixed $\mathbf{w}$:

$$\mathbf{v}_{t+1} = argmin(\sum_{i=1}^{n} v_i L_i - \lambda \sum_{i=1}^{n} v_i) \tag{4}$$

because the Eq.6 is convex, the global optimum $\mathbf{v}^* = [v_1^*, , v_n^*]$ is able to be easily calculated by:

$$v_i^* = \begin{cases} 1, & \text{if } L(y_i, g(\mathbf{x}_i, \mathbf{w})) < \lambda \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

In this process of update, we increase iteratively λ by multiplying the pace parameter μ to add more training samples for training. we can consider learning pace λ as "age" of the model, because it controls the number of samples to be considered in training: if the model is young (λ is small), it tends to select only "easy" samples with small value loss $L(y_i, g(x_i, \mathbf{w}))$ ($v_i^* = 1$), or otherwise unselect "hard" samples ($v_i^* = 0$). As λ grows, more samples with larger losses

will be gradually appended to train a more 'mature' model (Jiang et al., 2015). In all our experiments, we set the initial λ_0 such that in the first epoch more than half the samples will be selected (so that there are more easy samples than hard ones).

summarily, the basic self-paced learning optimization process is simple. $v_i = 1$ when $L(y_i, g(\mathbf{x}_i, \mathbf{w})) \leq \lambda$. $v_i = 0$ when $L(y_i, g(\mathbf{x}_i, \mathbf{w})) > \lambda$. In particular, the optimization problem of the above self-paced learning can be transformed into biconvex optimization problem when $f()$ and $r()$ are convex functions. The problem of biconvex optimization is that the parameter set z can be divided into two mutually exclusive sets z_1 and z_2. If any parameter set z_i fixed a set of parameter values, another set of parameter optimization problems can be seen as convex optimization problems. This kind of problem can be considered as a biconvex optimization problem. For example, in the above expressions, there are two sets of parameters $\mathbf{w}$ and $\mathbf{v}$. We can alternately fix $\mathbf{w}$ to obtain the optimal solution of $\mathbf{v}$, and then fix $\mathbf{v}$ to find the optimal solution of $\mathbf{w}$. This optimization method can be called alternative convex search (ACS) method, which can guarantee the local optimal solution of the function(Gorski et al., 2007).

However, SPL also has a limitation that basic SPL cannot incorporate prior knowledge into training, resulting to overfitting. It is obvious that ignoring prior knowledge is less reasonable when reliable prior information is available(Gorski et al., 2007).

2.2. Self-paced Curriculum Learning

To alleviate the problem of SPL, Gorski et al. (2007) introduce a new framework named Self-paced Curriculum Leaning (SPCL), which could combine both prior information known before training and knowledge learned during training.

Similar in CL, SPL consider that the model is given a curriculum. Following the notation defined above, the objective of SPCL can be defined:

$$\min_{w,v\in[0,1]^n} E(\mathbf{w}, \mathbf{v}, \lambda) = \sum_{i=1}^{n} v_i L(y_i, g(x_i, \mathbf{w})) + f(\mathbf{v}; \lambda) \quad (6)$$

Where f is named the self-paced function that determines the learning scheme. The self-paced function is convex respect with to $\mathbf{v} \in [0, 1]^n$.

Algorithm 3 Self-paced Curriculum Learning.

input training sets $\mathcal{D}$, pace parameter $\mu > 1$, self-paced function f

initialize v^*, λ_0
while not converged **do**
  Update $\mathbf{w}^* = argmin_{\mathbf{w}} E(\mathbf{w}, \mathbf{v}^*)$
  Update $\mathbf{v}^* = argmin_{\mathbf{v}} E(\mathbf{w}^*, \mathbf{v})$
  if λ is small then increase λ by the μ
end while
return $w = w^*$

self-paced function: The self-paced function can be considered as a regularization term that is independent of original loss functions, and can be easily used to solve various problems. As discussed above, we fixed the weight $\mathbf{w}$ to find expected value of $\mathbf{v}$ by optimizing the objective:

$$\mathbf{v}^* = arg \min_{\mathbf{v}\in[0,1]^n} E_w = arg \min_{\mathbf{v}\in[0,1]^n} \sum_{i=1}^{n} v_i l_i + f(\mathbf{v}, \lambda) \quad (7)$$

We will briefly introduce three kinds of regularization terms here and implement them in our experiments:

1. **Linear Soft Weighting**: A common self-paced function is the Linear soft weighting which is widely applied:

$$f(\mathbf{v}, \lambda) = \lambda(\frac{1}{2}\|\mathbf{v}\|^2 - \sum_{i=1}^{n} v_i) \quad (8)$$

in which $\lambda > 0$. As the objective 7 is convex, we can compute the expected value of $\mathbf{v}^*$ by setting its partial derivatives $\frac{\partial E_w}{\partial v_i}$ equal zero:

$$v^*(\lambda, l) = \begin{cases} -\frac{l}{\lambda} + 1, & l < \lambda \\ 0, & l \geq \lambda \end{cases} \quad (9)$$

2. **Logarithmic Soft Weighting**: A more conservative method is to optimize the loss logarithmically:

$$f(\mathbf{v}, \lambda) = \sum_{i=1}^{n} (1 - \lambda) v_i - \frac{(1 - \lambda)^{v_i}}{\log(1 - \lambda)} \quad (10)$$

where $0 < \lambda < 1$. Similarly, we can compute the expected value of $\mathbf{v}$ by the following function:

$$v^*(\lambda, l) = \begin{cases} \frac{\log(l+1-\lambda)}{1-\lambda}, & l < \lambda \\ 0, & l \geq \lambda \end{cases} \quad (11)$$

3. **Mixture Weighting**: Mixture scheme is a combination of the "soft" and the "hard" scheme (Jiang et al., 2014). This function introduces a new parameter, i.e. $\lambda = [\lambda_1, \lambda_2]^T$. It can be implemented by the following function:

$$f(\mathbf{v}, \lambda) = -\zeta \sum_{i=1}^{n} \log(v_i + \frac{1}{\lambda_1}\zeta) \quad (12)$$

in which $\zeta = \frac{\lambda_1\lambda_2}{\lambda_1 - \lambda_2}$, and $\lambda_1 > \lambda_2 > 0$. its expected value of $\mathbf{v}$ is:

$$v^*(\lambda, l) = \begin{cases} \frac{1}{\log\zeta} \log(l + \zeta), & l < \lambda \\ 0, & l \geq \lambda \end{cases} \quad (13)$$

3. Experiments

3.1. Motivation

We implement the methods of basic SPL as well as three kinds of SPCL-regularization terms (Linear Soft Weighting, Logarithmic Soft Weighting and Mixture Weighting). Our

experiments aim to evaluate the effect of each regularization term and compare their performance.

Additionally, another task is to investigate how the hyper-parameters (λ and μ) affect the performance of each regularization term and to find the best configuration of the two hyper-parameters to get the highest validation accuracy on our CIFAR-10 classifier.

3.2. Description

During the experiments, we use CIFAR-10 and CIFAR-100 datasets which are well-known datasets for object recognition in images. CIFAR-10 comprises 60,000 colour images (3x32x32 pixels), with 10 object classes and it has been split into a training set of 40,000 images, a validation set of 10,000 images and a test set of 10,000 images. CIFAR-100 is similar to CIFAR-10, but has 100 classes; the training set size is again 40,000 images with validation and test sets each containing 10,000 images.

In this coursework, we focus on CIFAR-10 to do our experiments.

In previous experiments, we implemented and tested different activation functions (leaky-ReLU Maas et al. (2013), sigmoid, tanh), different learning rules, different network architectures, different regularization methods (L1 penalty, L2 penalty and dropout) and batch normalization. These experiments help us to construct a baseline model for further exploration. This model has two fully connected layers of neural network and each layer has 1024 leaky-ReLU units. We use Adam learning method Kingma & Ba (2014) with the start learning rate of 0.001. For preventing over-fitting, we apply dropout to each layer with the dropout ratio of 0.5. The final accuracy we achieve through this model is 0.820 on CIFAR-10 test set. The error evolution and its detailed statistics are shown below.

training loss	training accuracy	validation loss	validation accuracy	test accuracy
0.148	0.946	0.647	0.838	0.820

Table 1. The result of baseline

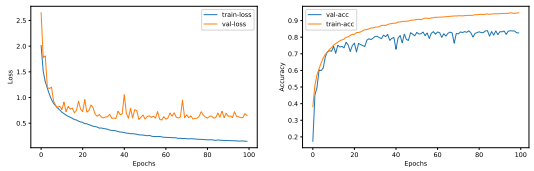

Figure 1. Training baseline system

As we can see, the baseline system is good enough, so that our further experiment will be built on the basis of this model. In addition, we set the initial λ_0 such that in the first epoch more than half the samples will be selected (so that there are more easy samples than hard ones).

3.3. Results

As what have been described before, we implement the basic SPL function with different pace parameter μ see algorithm 2 into baseline system and trained 100 epochs as well. In this experiment, we also try $\mu = 1.2$, but it is too big to take data in, so in Table 2, the experimental results for $\mu = 1.2$ are lost. Because of this, there is no need to try the value of $\mu = 1.2$ that higher than 1.2 any longer. The final results are shown in table 2 and figure 2 and figure 3.

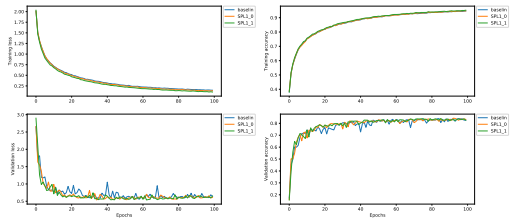

Figure 2. SPL with different pace parameter (μ=1.0, 1.1) compared with baseline

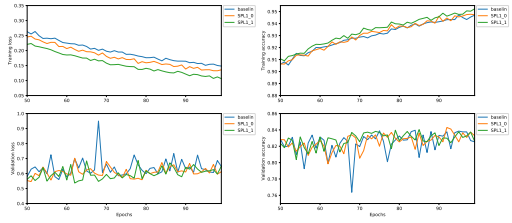

Figure 3. A zoom out version of figure 2(50 to 100 epochs).

experiment	test loss	test accuracy
baseline	0.634	0.820
SPL1.0	0.632	0.834
SPL1.1	0.627	0.835
SPL1.2	×	×

Table 2. The performance of basic SPL models with different pace parameter (μ=1.0, 1.1, 1.2) and baseline on test set.

It is obvious to see that using SPL functions can obtain a better and more stable performance which have a lower loss and a more than 1 percent higher accuracy on test set. Therefore, we can conclude that SPL does improve the performance of our classifier on CIFAR-10 dataset with respect to image classification accuracy. The reason why SPL outperforms the baseline might be SPL successful avoid local minimum and can abtain a better global minimum. Moreover, we can also see that when $\mu = 1.1$, the classifier performs best, with the classification accuracy climbing up to 83.5%.

Based on previous experiments, we decide to explore three kinds of SPCL-regularization terms and compare the performance of theses different regularization terms with stan-

dard SPL ($\mu = 1.1$). The three SPCL-regularization terms are Linear Soft Weighting, Logarithmic Soft Weighting and Mixture Weighting. As described before, the algorithm 3 shows that basic idea of SPCL. We add Linear Soft Weighting described in equations 8 and 9, Logarithmic Soft Weighting described in equations 10 and 11, and Mixture Weighting described in equations 12 and 13. And as before, we try different values of μ as well. The results are shown in figure 4, figure 5 and table 3.

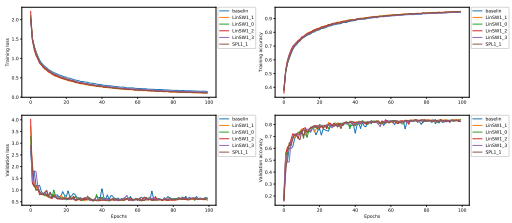

Figure 4. SPCL-Linear Soft Weighting with diffferent (μ=1.0, 1.1, 1.2, 1.3) compared with baseline

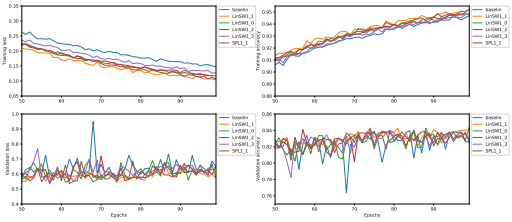

Figure 5. A zoom out version of figure 4(50 to 100 epochs).

experiment	test loss	test accuracy
baseline	0.634	0.820
LinSW1.0	0.623	0.831
LinSW1.1	0.603	0.841
LinSW1.2	0.622	0.833
LinSW1.3	0.625	0.832
SPL1.0	0.632	0.834
SPL1.1	0.627	0.835
SPL1.2	×	×

Table 3. The performance of SPCL-Linear Soft Weighting models with different pace parameter (μ=1.0, 1.1, 1.2, 1.3), basic SPL model ($\mu = 1.0, 1.1, 1.2$) and baseline on test set.

Compared with the baseline, SPCL-Linear Soft Weighting(LinSW) with $\mu = 1.1$ obtains a much better performance on test set, with the classification accuracy increasing by more than 1.5 percent on test set. It is also worth mentioning that SPCL with Linear Soft Weighting(LinSW) model ($\mu = 1.1$) generally works better than the standard SPL classifier ($\mu = 1.1$). Or that is to say, SPCL-Linear Soft Weighting(LinSW) is more robust than standard SPL, because the latter is sensible to choice of μ. In other words, choosing the proper pace (μ) is not an easy task for original SPL, because too large pace parameter will lead to bad performance due to the existence of outliers and noisy data.

Then we try Logarithmic Soft Weighting and Mixture Weighting based on the baseline, setting the value of μ to be 1.1, 1.2, 1.3 respectively. The results in figure 6 shows that the μ is too big to introduce data, which results in model unable to even begin training. Thus, again, it demonstrate that SPL based techniques are sensitive to parameter selection.

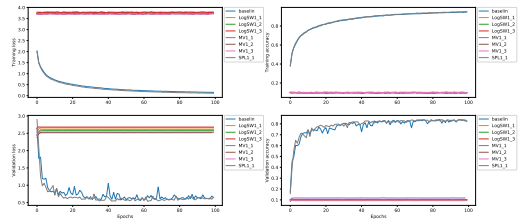

Figure 6. Logarithmic Soft Weighting and Mixture Weighting with different μ, basic SPL model($\mu = 1.1$) and baseline.

3.4. Interpretation and discussion

We tested the performance of SPL and three kinds of SPCL. It can be seen that Linear Soft Weighting is the self-paced function that is most suitable self-paced regularization term for the CIFAR-10 dataset. It can give higher accuracy both on validation and test sets. As discussed in the the Results 3.3, the use of SPL function is able to improve the performance of the model, with an significant improvement on classification accuracy (1.5%)

Furthermore, for the SPL function of Linear Soft Weighting, we investigated the effect of hyper-parameters (λ_0 and μ). As Kumar et al. (2010) discussed, the setting of λ_0 should control more than half samples to be selected. Therefore, we computed the likelihood of training data to set the value of λ_0 as 3.5. We also found the best value for μ is 1.1 (shown in table 3). The parameter λ means the "age" of the model so that the "young" model (small λ) consider less samples as "easy" while "mature" model will train more samples. The parameter μ controls the speed of model's growing so that higher μ cause that λ increases quickly and the model will train all samples after less iterations.

However, SPL is limited in the linear increase of λ. We update λ just by a fixed μ so that the model grow with a constant speed. We cannot stop the growth or slow it and unlimited growth of the model is meaningless. We thought that a possible idea to solve this issue is to keep the μ change through training time. For example, we can design the pace parameter as $exp(\frac{\gamma}{1+\alpha t})$, in which t is the time step (t^th iteration) and γ as well as α are new hyper-parameters. The γ controls the initial value of pace parameter and The α controls the speed of reduce of pace parameter. The $exp()$ keep the pace parameter always greater than one.

4. Related work

self-paced learning: By simulating the learning process of humans beings, self-paced learning technique has been widely used in modern machine learning tasks because it

can help to find a better local minimum and speed up the convergence of training towards the global minimum.

In Zhao et al. (2015), a self-paced matrix factorization (SPMF) structure which applies self paced technique to traditional matrix factorization was proposed in 2015. This really works because MF methods always have non-convex objectives and thus are vulnerable to local minimum, which can be avoided by using self-paced learning technique.

Another successful application that employs self-paced learning technique is long-term tracking Supancic & Ramanan (2013), which tackle object tracking problem in video processing. They employed self-paced learning strategy to label frames in the video, regarding images under partial occlusion as complex instances and the ones unoccluded as easy examples. It is proved that the model in this paper has dramatic better performance than the existing work.

A more recent study Li et al. (2016) applied self-paced learning strategy to multi-objective learning, which aims at addressing some weaknesses of SPL. First, researches have shown that SPL is vulnerable to initialization Jiang et al. (2014; 2015). Or, in other words, the strategy of initialization largely influences the performance of self-paced learning. Besides, self-paced learning is sensitive to outliers and noisy examples in the data set. Unlike traditional machine learning algorithms, SPL learns from easy to hard with the help of a wight parameter which is able to control the new instances learned by the model. Therefore, multi-objective self-paced learning (MOSPL) technique is able to train a more robust model even with bad initialization. In MOSPL, they divide the objectives into two parts, one is the standard cost function with regularization and the other is the regularization term in self-paced learning. As the name implies, when training the network, we take both objectives into consideration and compromise between loss and regularization term. Moreover, this variant of SPL has been proved to significantly outperform the original version on multiple applications including matrix factorization and action classification.

Ways of sample selection in machine learning: Inspired by the learning process of humans and animals, self-paced learning literately incorporates easy to more complex samples into training. In essence, multiple other techniques have been proposed to select samples in machine learning including active learning Settles (2012) and co-training Qiao et al. (2018).

However, self-paced learning is a supervised technique which means we can use the labels of examples to choose smaller training loss or higher likelihood samples during training, while active learning and co-training are always applied to semi-supervised structure.

Intuitively, active learning technique chooses the most useful unlabeled instances by some query algorithms and thus improves the performance the the model using human annotator data. Moreover, active learning strategy is super useful in large-scale machine learning tasks due to the fact that most real-world data are unlabeled, and always expensive, time or labor consuming to get the correct label. Besides, training with large amount of data is also time consuming, therefore active learning is a kind of technique that uses less training examples to train a classifier that has better performance. In the learning process of human beings, we always acquire new knowledge by exploiting current experiences and then accumulate experiences by knowledge that we already know. Inspired by the learning process described above, active learning improve the performance of the model by continuously revising the model according to the existing knowledge. As the name implies, active learning is able to acquire targeted knowledge instead of training passively.

The key of co-training is to choose most confidently unlabeled examples by more than two models using a few labeled data, which is widely used in applications where there is only small amounts of labeled data and large amounts of unlabeled data. By using two or more models to choose data, it is believed that each model applies different sets of features and thus we can acquire complementary information about the example.

Image recognition: Image recognition is a challenging and emerging task in computer vision that has been widely applied to multiple applications, including object recognition, visual positioning and tracking, face recognition. For us human beings, it is a super easy task to identify between different objects using our vision system and brain. However, learning to classify different objects is somewhat not an easy job for machines. Existing techniques for image recognition can be roughly divided into four classes: Neural network Krizhevsky et al. (2012), decision tree Zhang et al. (2017), support vector machine Peng et al. (2015) and fuzzy measure Korytkowski et al. (2016).

Recent advances in deep neural network, especially convolutional neural networks (CNN) Simonyan & Zisserman (2014), are proved to work well on image processing tasks.

According to Ciregan et al. (2012), CNNs have successfully reduced the error rate of digital character recognition task on MNIST database to 0.23. Moreover, the results on CIFAR10 data set also outperformed existing techniques, with a very low error rate of 17.41% at that time. Using the same image data set, Ciresan et al. (2011) reported the surprisingly rapid convergence speed using CNN.

More recent studies apply CNNs to face recognition/detection tasks Li et al. (2015); Farfade et al. (2015) and have been demonstrated as a successful in the field of computer version.

5. Conclusions

In this project, we implement Self-Paced Learning (SPL) and Self-Paced Curriculum Learning (SPCL) with three different regularization terms (Linear Soft Weighting, Logarithmic Soft Weighting and Mixture Weighting) base on a

2-layer convolutional neural network.

Overall, according to our experimental results we can come up with following conclusions with respect to the overall research questions and objectives.

Firstly, the use of SPL technique evidently improve the performance of our convolutional neural network, with the classification accuracy on CIFAR-10 climbing up to 84.1% on test set, more than 2 percent higher than that of the baseline. This significant improvement has demonstrated that SPL does help NNs to find a better local minimum as a regularizer and speed up the convergence of training towards the global minimum for convex objectives, which conforms to the theoretical understanding proposed in (Meng et al., 2015).

Secondly, we found that SPCL with Linear Soft Weighting (LinSW) outperforms other regularization schemes on this CIFAR-10 image classification task, with an increase over 0.05% on test set accuracy compared with standards SPL.

However, experimental results on other two regularization terms show that the model is sensitive to parameter initialization (μ, λ). The parameter λ means the "age" of the model so that the "young" model (small λ) consider less samples as "easy" while "mature" model will train more samples. The parameter μ controls the speed of model's growing so that higher μ cause that λ increases quickly and the model will train all samples after less iterations.

Empirically, we also found that it is super important for self-paced curriculum learning technique to have a development set under the same distribution to tune parameters (μ, λ).

Finally, in standard SPL, we apply a binary variable (0 or 1) in objective to determine whether the instance is easy or hard to learn and define a pace parameter (μ) to determine whether or not to accept more examples. Moreover, a soft weighting scheme with three regularization terms is used in SPCL, the use of real-valued variables efficiently representing the easiness of examples in the particular dataset. From our experimental results, Logarithmic Soft Weighting and Mixture Weighting regularization terms in particular, SPL based techniques are vulnerable to the value of parameters (μ, λ). With improper value of parameters, the model is even unable to begin training. If we consider most of instances in dataset as difficult examples, there are not enough data for the model to learn what we want. Furthermore, it is hard to choose a proper end point for SPL based models which controls when to stop learning new instances. Or, in other words, it is a dilemma for SPL to select the proper value of the pace parameter which determines when the model stops accepting new data.

For future research, we'd like to run experiments on more variants of SPL based models including multi-objective self-paced learning (MOSPL) (Li et al., 2016) and try to solve the problem of pace parameter mentioned in conclusions.

References

Bengio, Yoshua, Louradour, Jérôme, Collobert, Ronan, and Weston, Jason. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48. ACM, 2009.

Ciregan, Dan, Meier, Ueli, and Schmidhuber, Jürgen. Multi-column deep neural networks for image classification. In *Computer vision and pattern recognition (CVPR), 2012 IEEE conference on*, pp. 3642–3649. IEEE, 2012.

Ciresan, Dan C, Meier, Ueli, Masci, Jonathan, Maria Gambardella, Luca, and Schmidhuber, Jürgen. Flexible, high performance convolutional neural networks for image classification. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, pp. 1237. Barcelona, Spain, 2011.

Elman, Jeffrey L. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99, 1993.

Farfade, Sachin Sudhakar, Saberian, Mohammad J, and Li, Li-Jia. Multi-view face detection using deep convolutional neural networks. In *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, pp. 643–650. ACM, 2015.

Gorski, Jochen, Pfeuffer, Frank, and Klamroth, Kathrin. Biconvex sets and optimization with biconvex functions: a survey and extensions. *Mathematical Methods of Operations Research*, 66(3):373–407, 2007.

Jiang, Lu, Meng, Deyu, Mitamura, Teruko, and Hauptmann, Alexander G. Easy samples first: Self-paced reranking for zero-example multimedia search. In *Proceedings of the 22nd ACM international conference on Multimedia*, pp. 547–556. ACM, 2014.

Jiang, Lu, Meng, Deyu, Zhao, Qian, Shan, Shiguang, and Hauptmann, Alexander G. Self-paced curriculum learning. In *AAAI*, volume 2, pp. 6, 2015.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Korytkowski, Marcin, Rutkowski, Leszek, and Scherer, Rafał. Fast image classification by boosting fuzzy classifiers. *Information Sciences*, 327:175–182, 2016.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Kumar, M Pawan, Packer, Benjamin, and Koller, Daphne. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, pp. 1189–1197, 2010.

Li, Hao, Gong, Maoguo, Meng, Deyu, and Miao, Qiguang. Multi-objective self-paced learning. In *AAAI*, pp. 1802–1808, 2016.

Li, Haoxiang, Lin, Zhe, Shen, Xiaohui, Brandt, Jonathan, and Hua, Gang. A convolutional neural network cascade for face detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5325–5334, 2015.

Maas, Andrew L, Hannun, Awni Y, and Ng, Andrew Y. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30, 2013.

Meng, Deyu, Zhao, Qian, and Jiang, Lu. What objective does self-paced learning indeed optimize? *arXiv preprint arXiv:1511.06049*, 2015.

Peng, Jiangtao, Zhou, Yicong, and Chen, CL Philip. Region-kernel-based support vector machines for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 53(9):4810–4824, 2015.

Qiao, Siyuan, Shen, Wei, Zhang, Zhishuai, Wang, Bo, and Yuille, Alan. Deep co-training for semi-supervised image recognition. *arXiv preprint arXiv:1803.05984*, 2018.

Settles, Burr. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, 2012.

Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Supancic, James S and Ramanan, Deva. Self-paced learning for long-term tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2379–2386, 2013.

Zhang, Rongting, Zhou, Guoqing, Huang, Jingjin, and Zhou, Xiang. Maximum variance unfolding based colocation decision tree for remote sensing image classification. In *Geoscience and Remote Sensing Symposium (IGARSS), 2017 IEEE International*, pp. 3704–3707. IEEE, 2017.

Zhao, Qian, Meng, Deyu, Jiang, Lu, Xie, Qi, Xu, Zongben, and Hauptmann, Alexander G. Self-paced learning for matrix factorization. In *AAAI*, pp. 3196–3202, 2015.