Bhupender saini Follow
Oct 28, 2020 · 4 min read · Listen

Machine Learning: Understanding Mean Teacher Model

Recently, we got many intuitive machine learning models but one model that gained attention was the **Mean teacher model** by curious AI company[1]. The mean teacher mechanism of learning was intuitive and had shown the state of art results in the computer visualization domain. But, observing the performance of this model in the Natural Language Processing domain with different noise strategies was quite a journey of learning machine learning. Here, I will be sharing my understanding of the mean teacher model.

In the mean teacher model, two identical models are trained with two different strategies called the student and teacher model. In which, only the student model is trained. And, a very minimal amount of weights of student model is assigned to the teacher model at every step called exponential moving average weights that's why we call it as **Mean teacher.** The ability to utilize abundant unlabeled data called semi-supervised learning is one of the major advantages of the mean teacher.

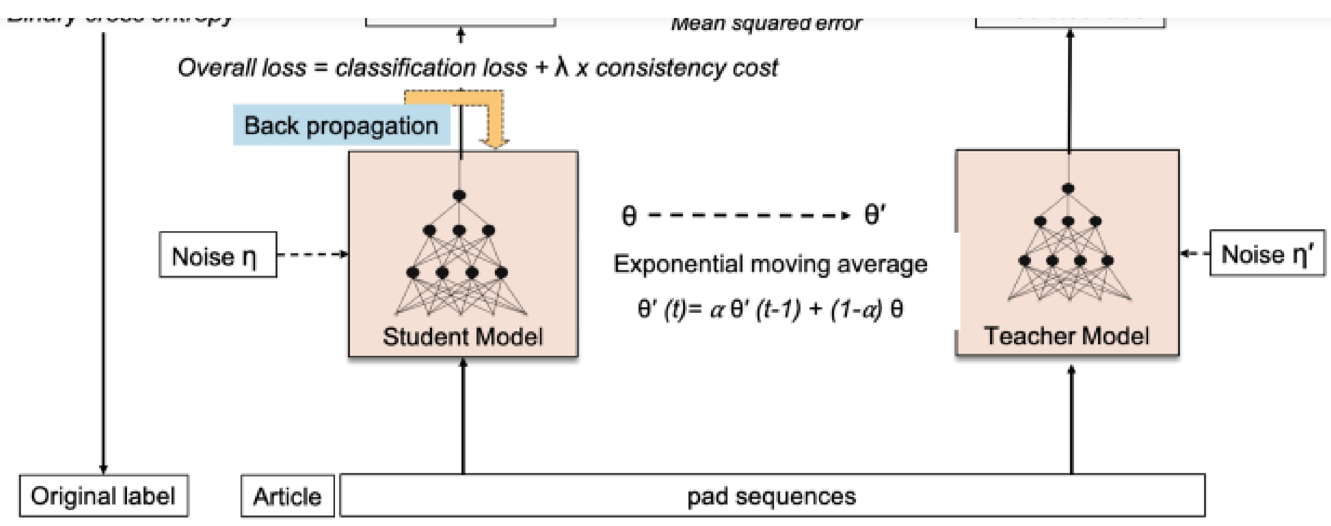


Figure 1: Mean teacher model working diagram.

As shown in Figure 1, two cost function plays an important role while backpropagating i.e. classification cost and consistency cost. Here, classification cost(C(θ)) is calculated as binary cross-entropy between label predicted by student model and original label. Consistency cost(J(θ)) is the mean squared difference between the predicted outputs of the student (weights θ and noise η) and teacher model (weights θ′ and noise η′).

Consistency cost is actually the distribution difference between two predictions (student and teacher prediction) and the original label is not required. During training, the model tries to minimize the distribution difference between the student and teacher model. So, instead of labeled data, we may utilize unlabelled data here. The mathematical declaration of consistency cost is as follows.

$$J(\theta) = \mathbb{E}_{x,\eta,\eta'}[\| f(x,\theta,\eta) - f(x,\theta',\eta')\|^2]$$

Consistency cost

One of the important factors that play a crucial role in adding robustness to the model is the introduction of noise during training. It's like fooling models with noise data so the model will not be biased towards a particular target and also can perform well while predicting unseen data.

$$O(\theta) = \lambda C(\theta) + (1 - \lambda) J(\theta)$$

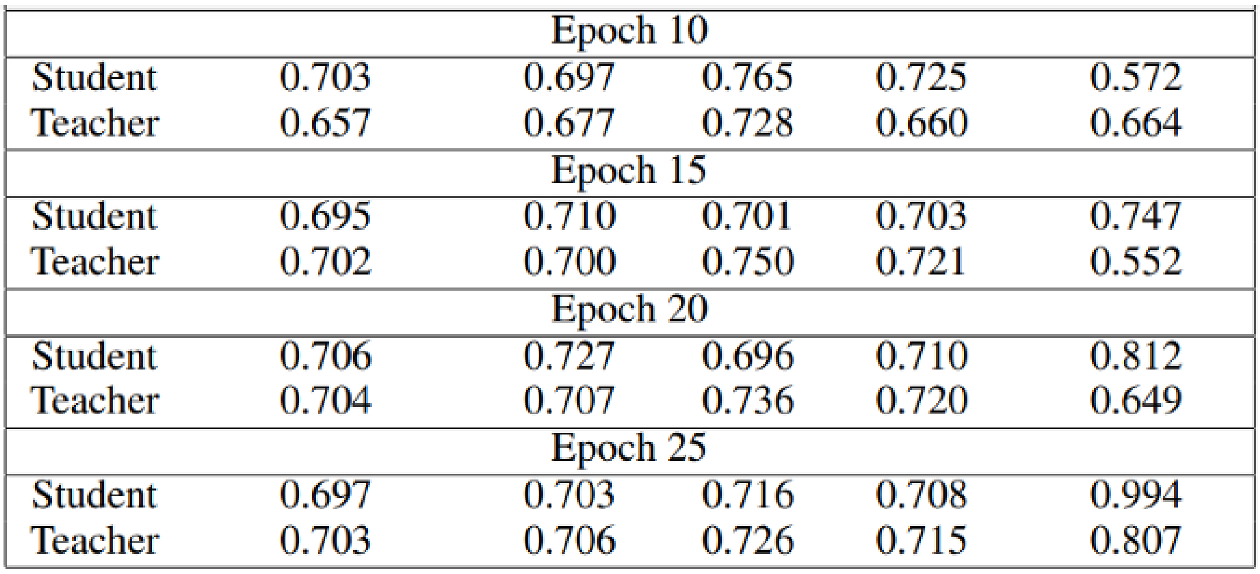
Overall Cost

During training, the exponential moving average(EMA) weights of the student model are assigned to the teacher model at every step and the proportion of weights assigned is controlled by parameter alpha(α). As mentioned in equation 3, while assigning weights, the teacher model holds its previous weights in alpha(α) proportion and (1–α) portion of student weights.

$$\theta'_t = \alpha \theta'_{t-1} + (1 - \alpha)\theta_t$$

Exponential moving average

We ran the mean teacher model for fake news detection and as per our observations, the teacher model starts performing better than the student model after particular epochs, in our case at epoch 15, where the teacher model starts overtaking the student model. However, the convergence of the teacher model depends on epoch, batch size, train data size, and alpha α. The tuning of parameters is required to get better results.

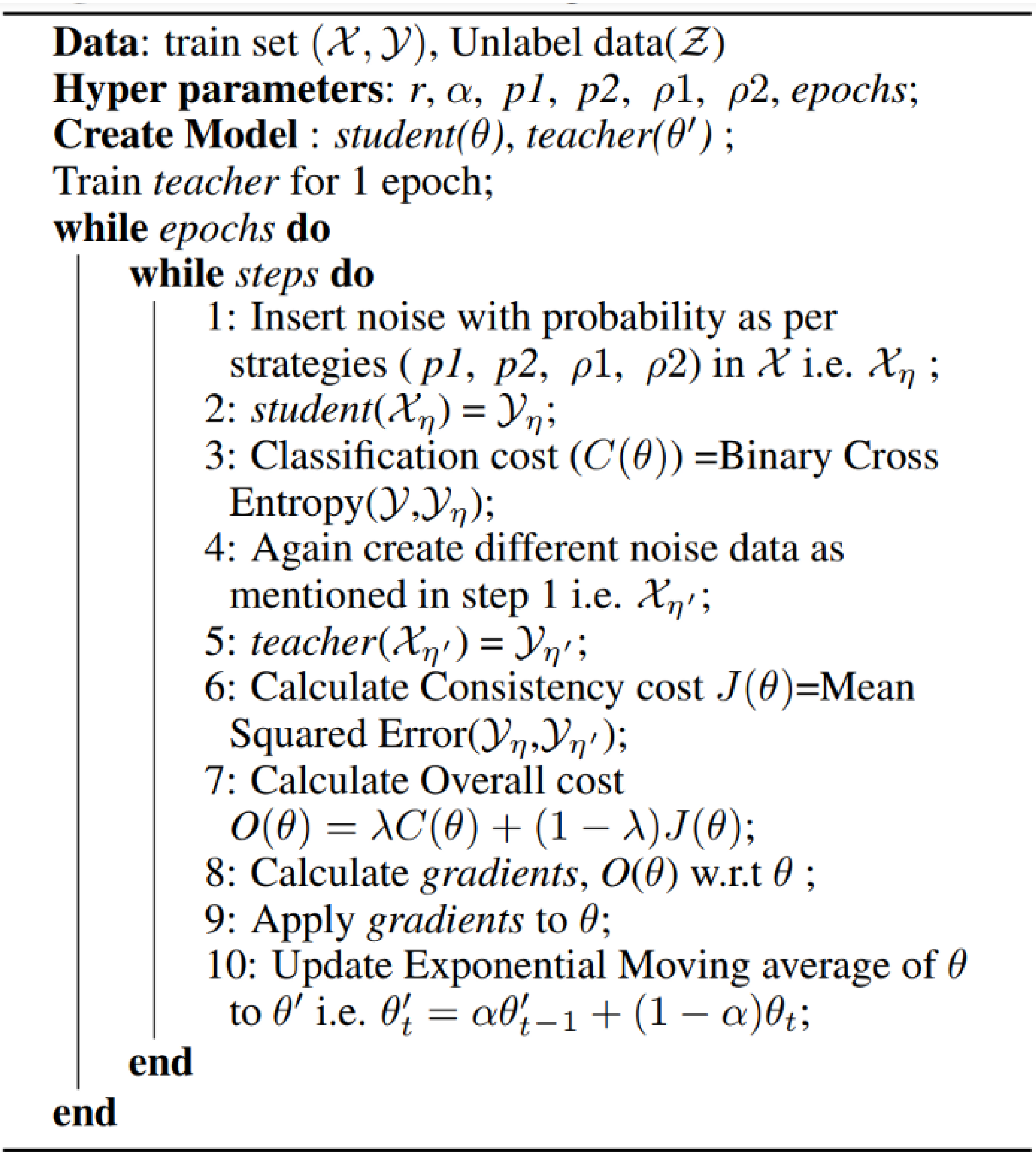


		Epoch 10			
Student	0.703	0.697	0.765	0.725	0.572
Teacher	0.657	0.677	0.728	0.660	0.664
		Epoch 15			
Student	0.695	0.710	0.701	0.703	0.747
Teacher	0.702	0.700	0.750	0.721	0.552
		Epoch 20			
Student	0.706	0.727	0.696	0.710	0.812
Teacher	0.704	0.707	0.736	0.720	0.649
		Epoch 25			
Student	0.697	0.703	0.716	0.708	0.994
Teacher	0.703	0.706	0.726	0.715	0.807

Table 1

Algorithm:

The complete algorithm of the mean teacher methodology is as follows:



```
Data: train set $(\mathcal{X}, \mathcal{Y})$, Unlabel data$(\mathcal{Z})$
Hyper parameters: $r, \alpha,\ p1,\ p2,\ \rho1,\ \rho2, epochs$;
Create Model : $student(\theta), teacher(\theta')$ ;
Train teacher for 1 epoch;
while epochs do
    while steps do
        1: Insert noise with probability as per
           strategies ( $p1,\ p2,\ \rho1,\ \rho2$) in $\mathcal{X}$ i.e. $\mathcal{X}_\eta$ ;
        2: $student(\mathcal{X}_\eta) = \mathcal{Y}_\eta$;
        3: Classification cost $(C(\theta))$ =Binary Cross
           Entropy$(\mathcal{Y},\mathcal{Y}_\eta)$;
        4: Again create different noise data as
           mentioned in step 1 i.e. $\mathcal{X}_{\eta'}$;
        5: $teacher(\mathcal{X}_{\eta'}) = \mathcal{Y}_{\eta'}$;
        6: Calculate Consistency cost $J(\theta)$=Mean
           Squared Error$(\mathcal{Y}_\eta,\mathcal{Y}_{\eta'})$;
        7: Calculate Overall cost
           $O(\theta) = \lambda C(\theta) + (1-\lambda) J(\theta)$;
        8: Calculate gradients, $O(\theta)$ w.r.t $\theta$ ;
        9: Apply gradients to $\theta$;
        10: Update Exponential Moving average of $\theta$
           to $\theta'$ i.e. $\theta'_t = \alpha\theta'_{t-1} + (1-\alpha)\theta_t$;
    end
end
```

Mean Teacher Algorithm

Code:

Code snippet of mean teacher algorithm is shown below and if interested you can find complete code for fake news detection using weakly supervised learning in the github.

```
        for step, (x_batch_train, y_batch_train) in enumerate(train_dataset):
            with tf.GradientTape() as tape:

                #introducting noise here
                x_batch_sn= synonym_noise(x_batch_train.numpy(),maxlen,tokenizer)
                # Run the forward pass of the layer
                logits= student(x_batch_sn, training= True)
                #calculating precision recall here
                train_metrics(y_batch_train,logits)
                #Calculating classification cost
                classification_cost = classification_costs(logits,y_batch_train)
                #Different noise but same input
                x_batch_sn1= synonym_noise(x_batch_train.numpy(),maxlen,tokenizer)
                # teacher is predicting
                tar_teacher = teacher(x_batch_sn1) #x_batch_train
                #calculating consistency cost
                consistency_cost= Consistency_Cost(tar_teacher,logits)
                # calculating overall cost
                overall_cost= Overall_Cost(classification_cost, consistency_cost, rati

            #adding loss to student model
            grads= tape.gradient(overall_cost, student.trainable_weights)
            i=i+1
            steps.append(i)

            # the value of the variables to minimize the loss.
            optimizer.apply_gradients(zip(grads, student.trainable_weights))
            teacher= ema(student, teacher, alpha=alpha)
        # Evaluate both the model here.

```

Note: Classification cost is binary cross-entropy in my case for two classes therefore not mentioning here.

```
#custom loss function
def Overall_Cost(classification_cost, consistency_cost, ratio=0.5):
    return (ratio * classification_cost) + ((1 - ratio)*consistency_cost)

#function for consistency cost
def Consistency_Cost(teacher_output, student_output):
    return sq_diff_layer

# function for Exponential moving average
def ema(student_model, teacher_model, alpha):
     #taking weights
    student_weights = student_model.get_weights()
    teacher_weights = teacher_model.get_weights()

    #length must be equal otherwise it will not work
    assert len(student_weights) == len(teacher_weights), 'length of student and teache
        len(student_weights), len(teacher_weights))

    new_layers = []
    # assigning weights
    for i, layers in enumerate(student_weights):
        new_layer = alpha*(teacher_weights[i]) + (1-alpha)*layers
        new_layers.append(new_layer)
    teacher_model.set_weights(new_layers)
    return teacher_model
```

Conclusion:

The mean teacher model is quite a simple and intuitive model to get better prediction and has the option of utilizing unlabeled data during training. The teacher model in the end performs well with the test data or unseen data then the student model. However, The convergence of the teacher model depends on epoch, batch size, train data size, and alpha α. So, tuning of the parameter is required to get better results. Furthermore, different noise strategies or attention modeling can be introduced in this model for better performance and robustness.

References:

1. https://thecuriousaicompany.com/

2. Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results.

3. https://github.com/bksaini078/fake_news_detection