Paper summary - 'When and how epochwise double descent happens'

Ilanit sobol, Emmy Abitbul

March 2022

Course: 096275-ML2

Introduction

- The paper describes analytical models which demonstrate when Epochwise Double Descent occurs during training RESNET18 Neural Network on CIFAR10 data set, where the independent variable is number of epochs.
- In practice the number of training iterations is often estimated by monitoring generalization performance on validation data under the assumption that if the generalization performance no longer improves, or improves at a very slow rate then further training is unlikely to be worthwhile.
- With that been said, this is not always the case, owing the phenomena called 'Epochwise Double Descent' (EDD) in which the generalization performance initially improves, then begins to decrease before reversing course and improving again. If the generalization performance follows the EDD behavior, simple heuristics for determining the end of training may give sub-optimal results.
  - The paper shows, that EDD occurs only if the amount of label noise exceeds a critical threshold, and simple early stopping heuristics fail only for intermediate amounts of noise.
- The model developed in the paper, is based on the assumption that training data contains informative features that are difficult to learn and typically become more "relevant" in the end of the training.

Main Contributions

The paper emphasize two modifications to training procedure that both removes EDD:

1. Removing features: comes at a price of reduced generalization. In this method we choose only the k first principle components. That way we "get ride of" the problematic features that take awhile to train on.
2. Retrain the last layer: matches or exceeds the generalization of standard training. In this method we use a well trained Neural Networkand only retrain the last layer with a small modification, for finetuning.

In the end, we want to reduce Epochwise Double Descent, that is in order reduce the computational complexity and use simple heuristics for early stopping.

Epochwise Double Descent Model

Analytical and Practical

Goal: Building a toy example of epochwise double descent, where we explore the model's properties.

Introduction: During training, it seems that the large eigenvalues (i.e large scale features) are learned first by the model. Later in the training (at long training times), we conclude that the small eigenvalue are actually more dominant to the model learning.

Simple Model description: We assume a training data-set $\boldsymbol{X} \in^{N\times D}$, meaning we have N data points with D features such that $[\boldsymbol{X}]_{ij} \sim \mathcal{N}(0,1)$. Each point is assigned a label $\boldsymbol{y} \in^{C}$, where in our probabilistic world we take $[\boldsymbol{y}]_i \in \{0,1,...,C\}$. In our words : we have C possible classes and we take the class with the highest probability

The matrix $\boldsymbol{\Phi} \in \mathbb{R}^{F\times N}$ given by the result of an F dimensional embedding of x by $[\boldsymbol{\Phi}]_{ij} = \phi_i(x_j)$ where each $\phi_i(\cdot)$ is a map $\phi_i : \mathbb{R}^D \to \mathbb{R}$. In our case: we use $\boldsymbol{\Phi}$ as the features output of the last dense layer.

Assumptions:

- When the training error is to decrease in this regime, then the small eigenvalues must be relatively noise free.
- When the training error is to increase in this regime, then the larger eigenvalues must be noisy.
- In our regime the researchers construct a model where the noise effects the larger eigenvalues.

  The above assumptions will suffice to demonstrate the Epochwise Double Descent phenomena during training.

Cost function: The paper uses MSE cost, for some ϵ noise (which effects only the larger eigenvalues), where the cost is a function of the σ[1] and λ[2], also means if the model is over/underparameterized)

Classes of behavior in regards to model dynamics

1. NDD-NES : No Epochwise Double Descent, No early stopping
2. NDD-ES : No Epochwise Double Descent, with early stopping
3. EDD-NES : Epochwise Double Descent, No early stopping
4. EDD-ES : Epochwise Double Descent, with early stopping

[1] $\sigma = standarddeviation$
[2] $\lambda = \frac{\#features}{datasetsize} = \frac{D}{N}$

In regards to the above behaviour classes, There are 4 practical properties:

***P*1**: For clean detests (low variance), early stopping is not necessary for good generalization

***P*2**: Epochwise Double Descent occurs when $\lambda \approx 1$, i.e critical parameterization

***P*3**: Epochwise Double Descent requires that a critical amount of noise occurs in the dataset

***P*4**: When the noise amount is bigger enough, early stopping will give the best results in generalization performance, and we shouldn't train for too long expecting Epochwise Double Descent

Empirical challenges:

In theory, when the model is very big, Epochwise Double Descent shouldn't occur. The researchers didn't see the above claim in practice. In their experiments, they used RESNET18 architecture on CIFAR10, but because of computational limitation, they couldn't run the network with width bigger than 92. In addition, because of data augmentation and parameter redundancy, it is unclear at what width we shouldn't see Epochwise Double Descent.

Epochwise Double Descent Main results

1. Method 1 - Remove features

In this experiment, the researchers proposed a method to avoid Epochwise Double Descent. They discarded the least significant features (i.e the smaller scale features), and trained the network (via standard training procedure) on the remaining components. This is also called Rank K approximation, and it is implemented using PCA.

Empirical results:

1. In the control experiment (model trained on all features), Epochwise Double Descentoccurs on the test set.
2. Epochwise Double Descent-phenomena disappears as the number of smaller scale features decreases, i.e the remaining features consists of 90% of the variance.
3. We can see that 100 important components are the larger scale features, and without them we couldn't understand the images (figure C in the paper).
4. Removing Informative features, who correspond to smaller eigenvalues, although they are not the most important (in their contribution to the overall variance), results in better generalization. In our words, the smaller scale features are responsible for fine tuning the model, which helps in making the model more precise.

Our results:

We implemented PCA in a smaller scale with a subset of 4 classes of CIFAR10 using RESNET18. Results and conclusions are in attached 'EDD results Analysis.pdf'

2. Method 2 - Retrain the last layer

The Challenge: In method 1 we dropped features and focused on the main components (who corresponds to the large eigenvalues) but this lead to a drop in generalization performance.

The Proposed Solution:

1. Training Deep Neural Networkwith SGD soft max cross entropy loss during t epochs - standard training
2. Replace the weights of the final classification layer with 'converged' weights calculated using analytical solution with Gradient descent, using the penultimate layer[3] activations as features. The result is a network which has the final layer trained to convergence, and the earlier layers trained for t epochs with a standard training procedure.

Empirical Results:

This method removes Epochwise Double Descent all completely and got better results than standard training. Analytical analysis on the subject of the 'converged weights' can by found the following Section.

Our results:

We implemented the method on a synthetic data set (very similar to toy example setup described in the paper), and replaced the layer weights' with the weights defined by equation (4) in the following section. Analysis, results and conclusions are in attached 'EDD results Analysis.pdf'

[3]Penultimate layer is a fully-connected hidden layer

Analytical analysis on the subject of the 'converged weights'

In the paper, there is a comparison between training dynamics with MSE, and with Cross Entropy Loss (XENT):

$$\mathcal{L}_{MSE} = \frac{1}{2N} \operatorname{Tr}\left[(\mathbf{w}\mathbf{\Phi} - \mathbf{y})^{\intercal} (\mathbf{w}\mathbf{\Phi} - \mathbf{y})\right] \tag{1}$$

$$\mathcal{L}_{XENT} = -\frac{1}{N} \operatorname{Tr}\left[\tilde{\mathbf{P}}_{\mathbf{L}}^{\intercal} \log\left(\mathbf{P}_{\mathbf{M}}\right)\right] \tag{2}$$

where:

$$\mathbf{P}_{\mathbf{M}} = \frac{e^{\beta \mathbf{w}_{XENT}\mathbf{\Phi}}}{1_C^{\intercal} e^{\beta \mathbf{w}_{XENT}\mathbf{\Phi}}} \quad \text{and} \quad \tilde{\mathbf{P}}_{\mathbf{L}} = \alpha \mathbf{P}_{\mathbf{L}} + (1-\alpha)\frac{1_{CN}}{C} \tag{3}$$

such that $\mathbf{P_L}$ is matrix of one-hot vectors defining the labels given by the y_js, $\tilde{\mathbf{P}}_{\mathbf{L}}$ is the label-smoothed matrix [4] and $\mathbf{P_M}$ is a softmax activation function.

In addition it is shown that there is a correspondence between the training dynamics with MSE and XENT loss, which is illustrated in the following equations, which are derived from gradient descent analytical solution for each type, assuming a frozen subspace in which the parameters don't change in time (using $\mathbf{w}_{MSE}^{(\infty)}$ as this solution, and learning rate of γ) :

$$\mathbf{w}_{MSE}^{(t)} = \mathbf{w}_{MSE}^{(\infty)} + \left(\mathbf{w}_{MSE}^{(0)} - \mathbf{w}_{MSE}^{(\infty)}\right) \mathbf{U} \left[\mathbb{I}_F - \gamma \mathbf{\Lambda}\right]^t \mathbf{U}^{\intercal} \tag{4}$$

$$\mathbf{M}\mathbf{w}_{XENT}^{(t)} = \mathbf{M}\mathbf{w}_{XENT}^{(\infty)} + \mathbf{M}\left(\mathbf{w}_{XENT}^{(0)} - \mathbf{w}_{XENT}^{(\infty)}\right) \mathbf{U} \left[\mathbb{I}_F - \gamma \mathbf{\Lambda}\right]^t \mathbf{U}^{\intercal} \tag{5}$$

where:

$$\mathbf{M}\mathbf{w}_{XENT}^{(\infty)} = \mathbf{M}\left(C\mathbf{P}_{\mathbf{L}} - 1_{CN}\right) \mathbf{\Phi}^{\intercal} \left(\mathbf{\Phi}\mathbf{\Phi}^{\intercal}\right)^{+} + \mathbf{M}\mathbf{w}_{XENT}^{(0)} \left(\mathbb{I}_F - \mathbf{U}\mathbf{\Lambda}^{+}\mathbf{\Lambda}\mathbf{U}^{\intercal}\right) \tag{6}$$

$$\mathbf{w}_{MSE}^{(\infty)} = \mathbf{y}\mathbf{\Phi}^{\intercal} \left(\mathbf{\Phi}\mathbf{\Phi}^{\intercal}\right)^{+} + \mathbf{w}_{MSE}^{(0)} \left(\mathbb{I}_F - \mathbf{U}\mathbf{\Lambda}^{+}\mathbf{\Lambda}\mathbf{U}^{\intercal}\right) \tag{7}$$

and

$\mathbf{M} \equiv \mathbb{I}_C - \frac{1}{C} 1_{CC}$

$\mathbf{\Lambda}^{+}\mathbf{\Lambda}$ is a diagonal matrix with zeros on the diagonal corresponding to zero eigenvalues of $\mathbf{\Phi}\mathbf{\Phi}^{\intercal}$ and ones corresponding to nonzero eigenvalues.

[4] Label smoothing is a common practice to relax our confidence on the labels. In essence, label smoothing will help the model train around mislabeled data and consequently improve its robustness and performance.

Notes

1. Equation (4) is derived from a known property of Gradient descent, convergence equations, thus is not further elaborated in the above analysis.

2. If the assumption of frozen subspace don't hold for the MSE case, we can just use the following weights (instead of $\mathbf{w}_{MSE}^{(\infty)}$), which is a well known analytical result of Gradient Descent:

$$\mathbf{w}_{MSE}^{*} = \mathbf{y}\mathbf{\Phi}^{\intercal}\left(\mathbf{\Phi}\mathbf{\Phi}^{\intercal}\right)^{+} \tag{8}$$

   The above result can also be obtained if we set the right part of equation (7) to zero.

3. The correspondence between equations (4) and (5), and also (6) and (7) is intuitively obtained by replacing labels $\mathbf{y} = (C\mathbf{P_L} - 1_{CN})$ and by multiplying by $\mathbf{M}$.

Discussion

1. The results in the paper contradict the EMC theory[5]. In particular, some of the results show that a critical amount of noise is necessary for Epochwise Double Descent to occur, which is in contradiction to EMC theory which states that there are no conditions on the minimal amount of noise for Epochwise Double Descent.

2. One possible explanation is that Epochwise Double Descentoccurs as an interplay between the features in the data and the noise in the labels (as in our linear model) rather than as a result of the EMC. This result also correspond to our conclusions in method 2 analysis.

[5]EMC, effective model complexity hypothesis, defined as the maximum number of samples, given a dataset and training procedure that a model can fit with error $\leq const$