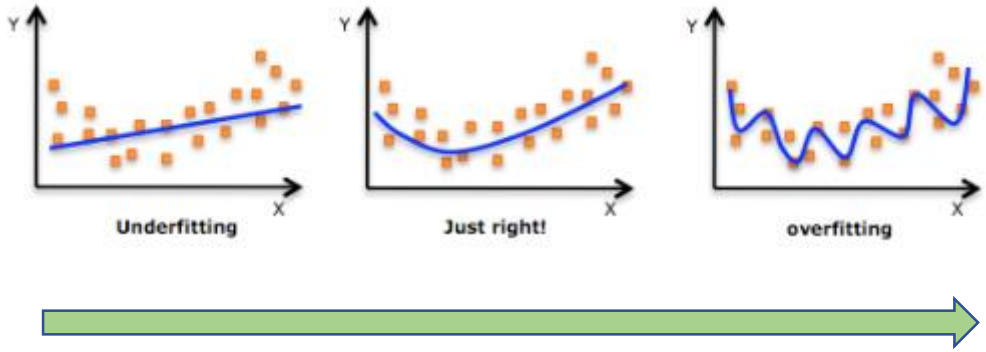
Chapter 6. 학습 관련 기술들

Overfitting

박종혁

Overfitting

Problem



Complexity가 높을수록 좋은 모델이 아니다?

Overfitting

Problem

그렇다면 Model Complexity를 낮추기 위한 방법은 무엇이 있을까?

Overfitting

Regularization

Regularization :

모델의 설명도를 유지하면서, 모델의 복잡도를 낮추는 방식으로 Overfitting을 해결하려는 방법 중 하나.

Tools:

Early Stopping / Noise Input / Drop Out / Pruning / Feature Selection / Ensemble / …

Overfitting

Problem

Overfitting을 억제하는 방법 1. 가중치 감소 (Weight Decay)

Overfitting

Weight Decay

가중치 감소(Weight Decay)란, 가중치 매개변수의 값이 작아지도록 학습하는 방법.

Overfitting

Weight Decay

가중치의 초기값이 왜 작아져야하지?

1. 가중치의 초기값은 올바른 학습을 하는데 매우 중요한 작용을 함.
2. Neural Net에서는 가중치가 클수록 Overfitting이 일어날 가능성이 큼. (경험적 근거)
3. 그렇다고 지나치게 0으로 다 맞춰버리는 경우, 학습이 진행됨에 따라 모두 같은 값으로 바뀌므로 가중치를 여러 개 갖는 의미를 사라지게 함.

Random하게 해야 할 필요성 제시.

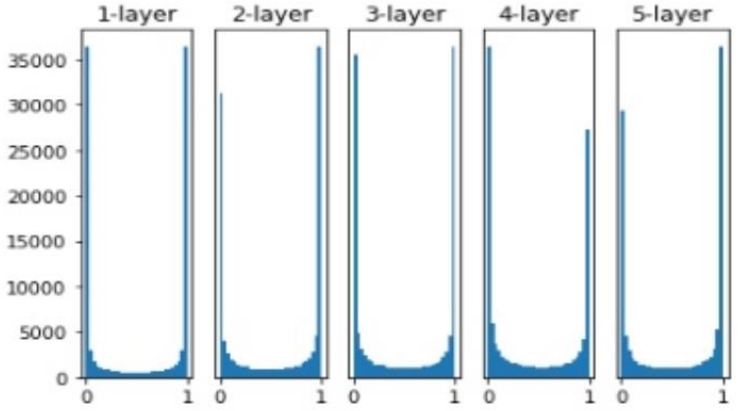
Initial Value

Problem

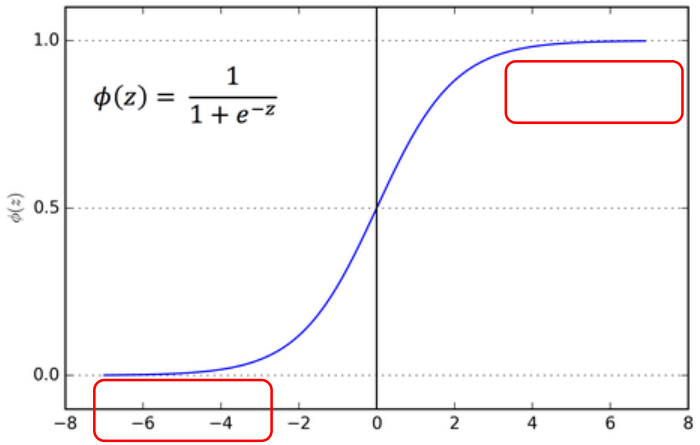
초기값이 달라짐에 따라 모델 내부에서는 값이
어떻게 바뀔까?

Overfitting

Case 1



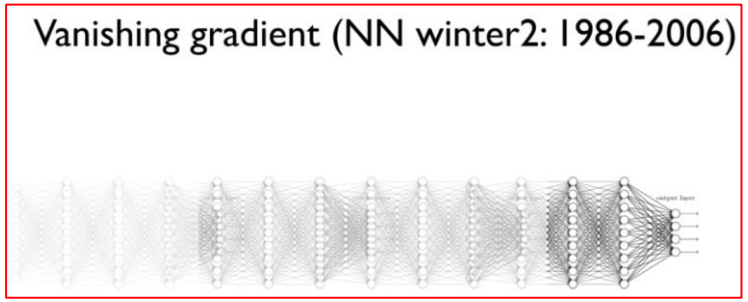
표준편차가 1인 정규분포로 초기화 한 경우.



Sigmoid Function

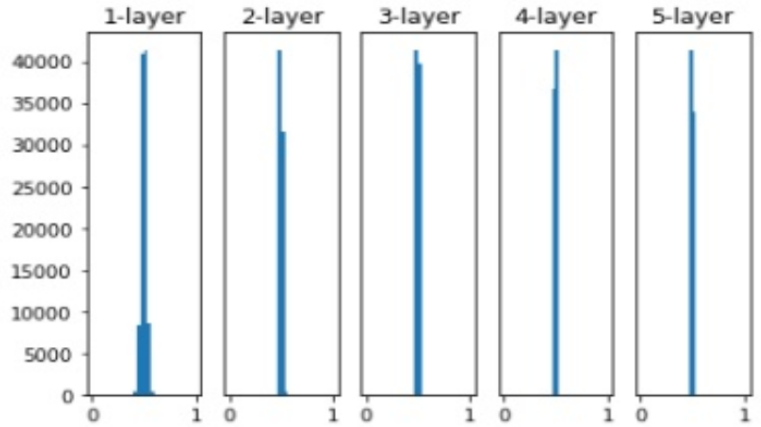
1. 각 층의 활성화값들이 0과 1에 치우쳐져 있음.
2. Sigmoid함수는 값이 0과 1로 가면 갈수록 평평해지기 때문에 기울기값(미분값)이 0에 가까워짐.
3. Back Propagation진행시 Sigmoid의 미분값이 곱해지는데 매우 작은 값이 곱해지게 됨.
4. Back Propagation 진행을 하면 할수록 (뒤로 갈수록) 기울기가 0에 수렴 → 학습이 잘 안됨.

Gradient vanishing!
(기울기 소실)



Overfitting

Case 2



1. 다수의 값들이 가운데에 포진하고 있음.
2. 이전 경우와 달리 Gradient Vanishing 문제는 일어나지 않음.
3. 하지만, 다수의 가중치들이 비슷한 값들로 되어있기 때문에, 여러 개의 뉴런을 둔 이유가 없어짐. → 표현력 제한!

Initial Value

Problem

그렇다면 어떤 초기값을 사용해야할까?

Overfitting

Xavier

AISTATS 2010

Understanding the difficulty of training deep feedforward neural networks

Xavier Glorot **Yoshua Bengio**
DIRO, Université de Montréal, Montréal, Québec, Canada

Abstract

Whereas before 2006 it appears that deep multi-layer neural networks were not successfully trained, since then several algorithms have been shown to successfully train them, with experimental results showing the superiority of deeper vs less deep architectures. All these experimental results were obtained with new initialization or training mechanisms. Our objective here is to understand better why standard gradient descent from random initialization is doing so poorly with deep neural networks, to better understand these recent relative successes and help design better algorithms in the future. We first observe the influence of the non-linear activations functions. We find that the logistic sigmoid activation is unsuited for deep networks with random initialization because of its mean value, which can

learning methods for a wide array of *deep architectures*, including neural networks with many hidden layers (Vincent et al., 2008) and graphical models with many levels of hidden variables (Hinton et al., 2006), among others (Zhu et al., 2009; Weston et al., 2008). Much attention has recently been devoted to them (see (Bengio, 2009) for a review), because of their theoretical appeal, inspiration from biology and human cognition, and because of empirical success in vision (Ranzato et al., 2007; Larochelle et al., 2007; Vincent et al., 2008) and natural language processing (NLP) (Collobert & Weston, 2008; Mnih & Hinton, 2009). Theoretical results reviewed and discussed by Bengio (2009), suggest that in order to learn the kind of complicated functions that can represent high-level abstractions (e.g. in vision, language, and other AI-level tasks), one may need *deep architectures*.

Most of the recent experimental results with deep architecture are obtained with models that can be turned into

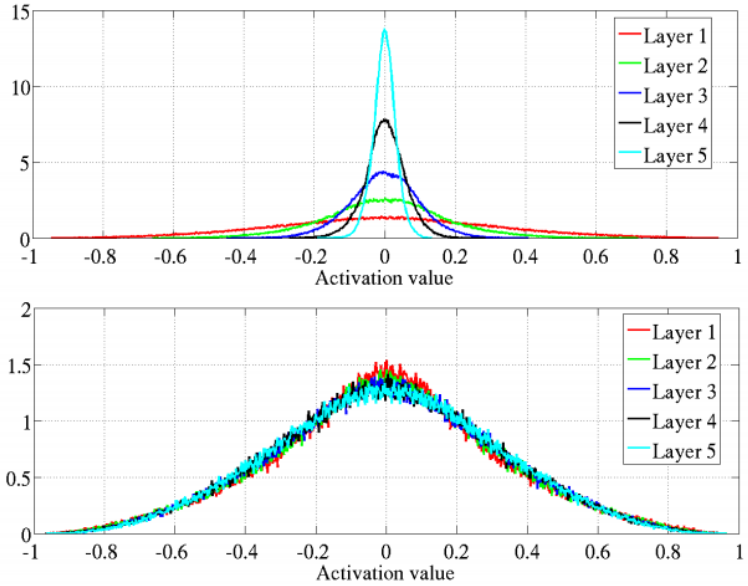


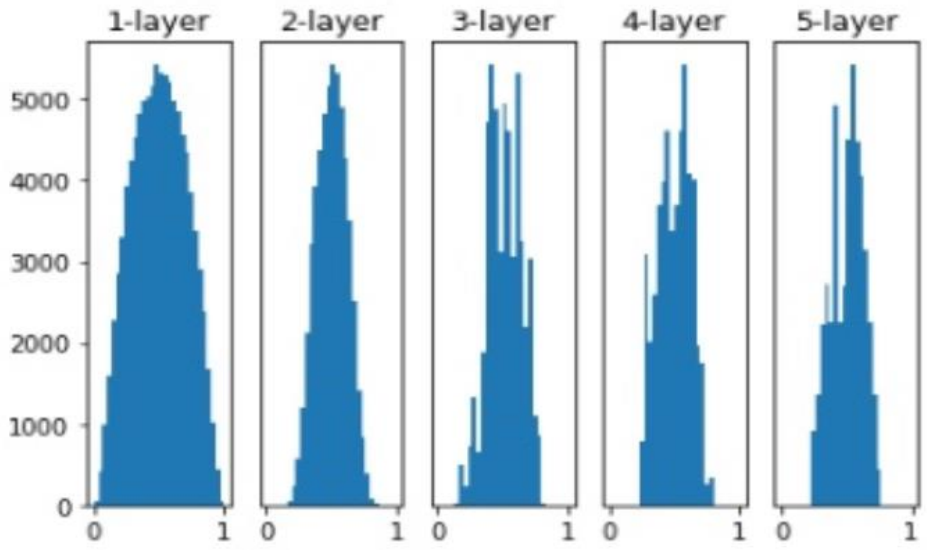
Figure 6: *Activation values normalized histograms with hyperbolic tangent activation, with standard (top) vs normalized initialization (bottom). Top: 0-peak increases for higher layers.*

전제 조건 : Activation Function은 linear Function.

[출처] http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf

Overfitting

Xavier



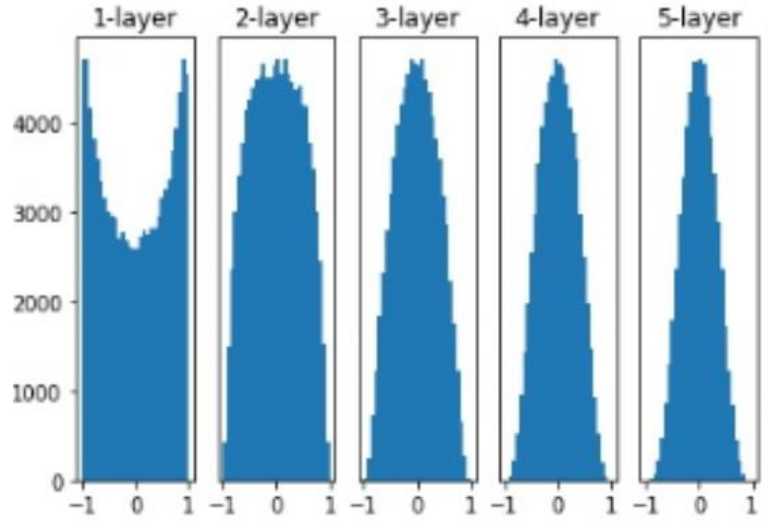
Xavier을 Sigmoid Function에 적용

1. 앞서 본 것과 달리 고르게 분포하고 있음을 볼 수 있다.

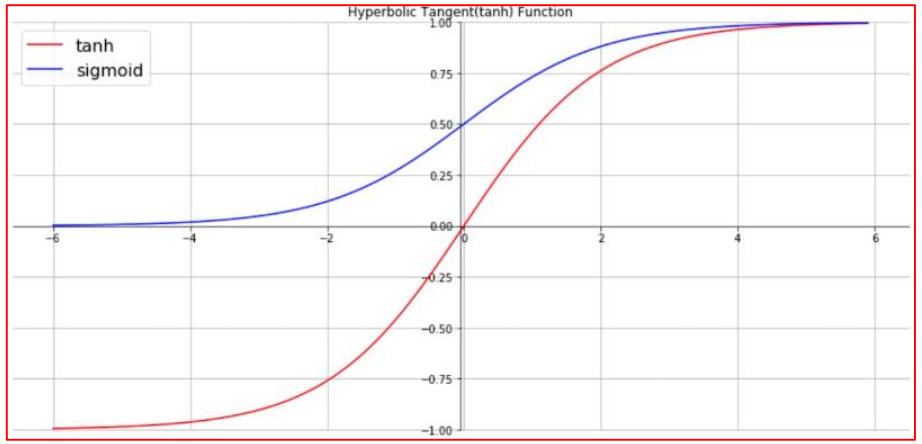
2. 각 층에 흐르는 데이터들이 적당히 퍼져있으며 표현력도 제한받지 않는다.

Overfitting

Xavier



Xavier을 Tanh Function에 적용

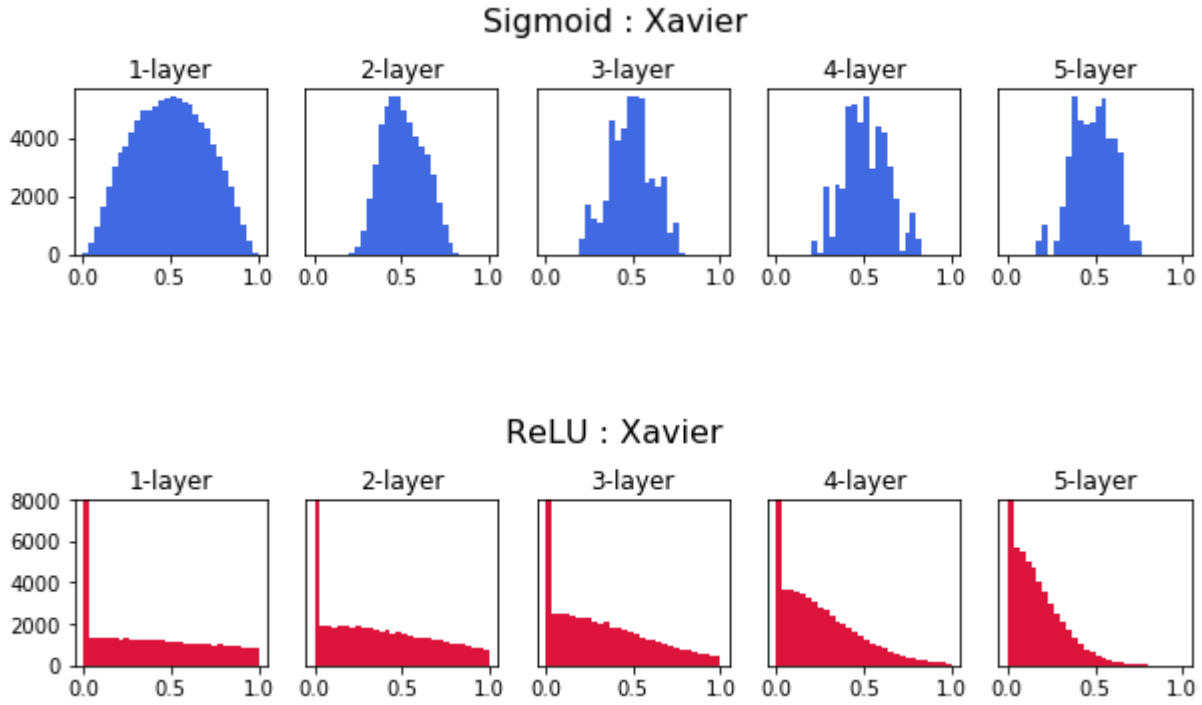


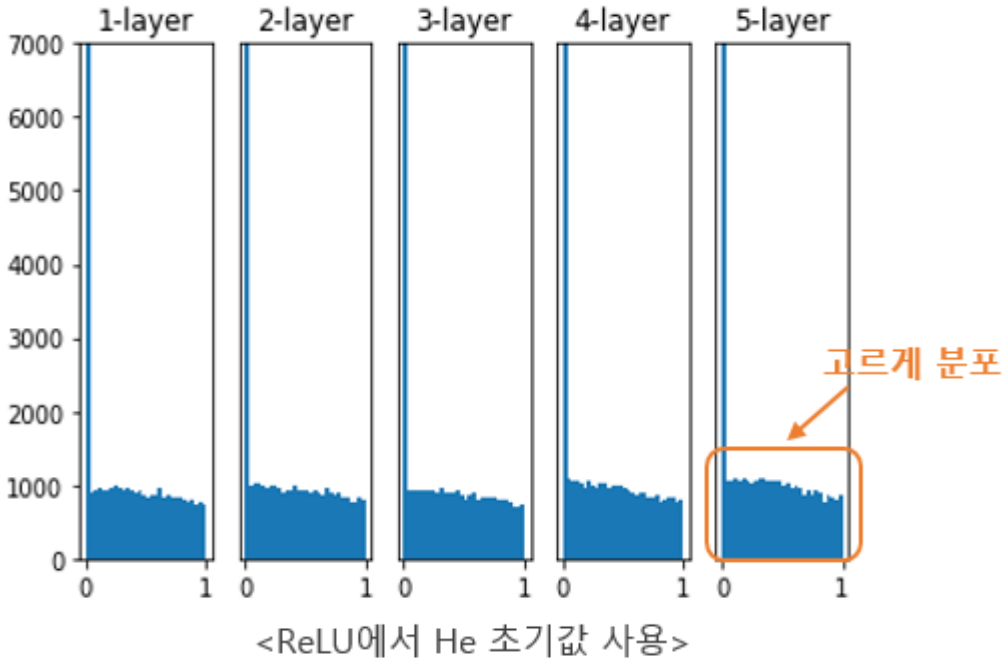
1. 정규분포(?) 형태를 띄고 있다.

2. 1-layer에서 여전히 -1과 1에 치우쳐진 상황을 관찰할 수 있다. → Gradient Vanishing 을 완전히 해결했다고 보기는 어려움.

Overfitting

He





<ReLU에서 He 초기값 사용>

Note :

$ReLU(x) = \max(0, x)$

Overfitting

Problem

다른 방법은 없는걸까?

Batch Normalization

Paper

ICML 2015

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

Sergey Ioffe
Google Inc., *sioffe@google.com*

Christian Szegedy
Google Inc., *szegedy@google.com*

Abstract

Training Deep Neural Networks is complicated by the fact that the distribution of each layer's inputs changes during training, as the parameters of the previous layers change. This slows down the training by requiring lower learning rates and careful parameter initialization, and makes it notoriously hard to train models with saturating nonlinearities. We refer to this phenomenon as *internal covariate shift*, and address the problem by normalizing layer inputs. Our method draws its strength from making normalization a part of the model architecture and performing the normalization *for each training mini-batch*. Batch Normalization allows us to use much higher learning rates and be less careful about initialization. It also acts as a regularizer, in some cases eliminating the need for Dropout. Applied to a state-of-the-art image classification model, Batch Normalization achieves the same accuracy with 14 times fewer training steps, and beats the original model by a significant margin. Using an ensemble of batch-normalized networks, we improve upon the best published result on ImageNet classification: reaching 4.9% top-5 validation error (and 4.8% test error), exceeding the accuracy of human raters.

1 Introduction

Using mini-batches of examples, as opposed to one example at a time, is helpful in several ways. First, the gradient of the loss over a mini-batch is an estimate of the gradient over the training set, whose quality improves as the batch size increases. Second, computation over a batch can be much more efficient than m computations for individual examples, due to the parallelism afforded by the modern computing platforms.

While stochastic gradient is simple and effective, it requires careful tuning of the model hyper-parameters, specifically the learning rate used in optimization, as well as the initial values for the model parameters. The training is complicated by the fact that the inputs to each layer are affected by the parameters of all preceding layers – so that small changes to the network parameters amplify as the network becomes deeper.

The change in the distributions of layers' inputs presents a problem because the layers need to continuously adapt to the new distribution. When the input distribution to a learning system changes, it is said to experience *covariate shift* (Shimodaira, 2000). This is typically handled via domain adaptation (Jiang, 2008). However, the notion of covariate shift can be extended beyond the learning system as a whole, to apply to its parts, such as a sub-network or a layer. Consider a network computing

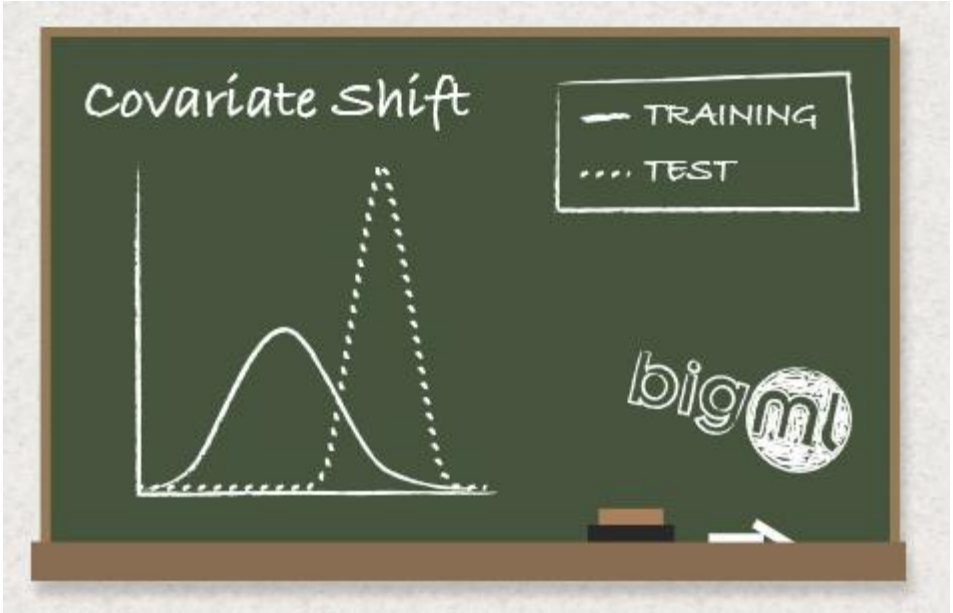
$$\ell = F_2(F_1(\mathrm{u}, \Theta_1), \Theta_2)$$

Batch Normalization

Problem

2 Towards Reducing Internal Covariate Shift

We define *Internal Covariate Shift* as the change in the distribution of network activations due to the change in network parameters during training. To improve the training, we seek to reduce the internal covariate shift. By



Machine Learning에서 Covariate shift 라고 하면 일반적으로 Train data와 Test data 의 분포가 다른 현상을 말한다.

Batch Normalization

ISSUE

How Does Batch Normalization Help Optimization?

Shibani Santurkar* MIT shibani@mit.edu
Dimitris Tsipras* MIT tsipras@mit.edu
Andrew Ilyas* MIT ailyas@mit.edu
Aleksander Mądry MIT madry@mit.edu

Abstract

Batch Normalization (BatchNorm) is a widely adopted technique that enables faster and more stable training of deep neural networks (DNNs). Despite its pervasiveness, the exact reasons for BatchNorm's effectiveness are still poorly understood. The popular belief is that this effectiveness stems from controlling the change of the layers' input distributions during training to reduce the so-called "internal covariate shift". In this work, we demonstrate that such distributional stability of layer inputs has little to do with the success of BatchNorm. Instead, we uncover a more fundamental impact of BatchNorm on the training process: it makes the optimization landscape significantly smoother. This smoothness induces a more predictive and stable behavior of the gradients, allowing for faster training.

1 Introduction

Currently, the most widely accepted explanation of BatchNorm's success, as well as its original motivation, relates to so-called *internal covariate shift* (ICS). Informally, ICS refers to the change in the distribution of layer inputs caused by updates to the preceding layers. It is conjectured that such continual change negatively impacts training. The goal of BatchNorm was to reduce ICS and thus remedy this effect.

Batch Norm이 잘 되는 이유는 Internal covariate shift(ICS)를 줄였기 때문에 잘 작동한다라는 사실이 현재 다들 알고 있는 사실임.

Even though this explanation is widely accepted, we seem to have little concrete evidence supporting it. In particular, we still do not understand the link between ICS and training performance. The chief goal of this paper is to address all these shortcomings. Our exploration lead to somewhat startling discoveries.

하지만 우리는 ICS와 훈련 performance와의 관계가 잘 이해가 안된다.(= 무슨 상관인지 잘 모르겠다.)
따라서 우리가 생각하는 Batch Norm이 잘 되는 이유를 설명해보겠다.

Batch Normalization

Advantage

1. 학습을 빨리 진행할 수 있다. (학습 속도 개선)

2. 초깃값에 크게 의존하지 않는다. (골치 아픈 초깃값 선택 장애여 안녕!)

3. 오버피팅을 억제한다. (드롭아웃 등의 필요성 감소)

Batch Normalization

Idea

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

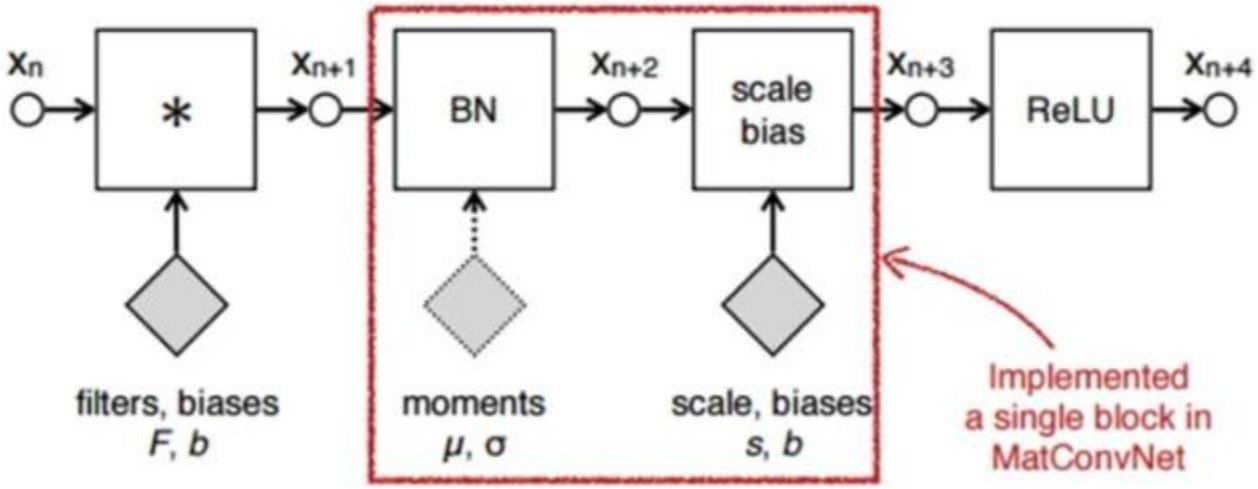
$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

1. 입력값(Mini-batch)의 평균을 구한다.
2. 입력값(Mini-batch)의 분산을 구한다.
3. 각 입력데이터에 대해 Normalize 작업을 수행한다 (평균이 0 분산이 1이 되도록)
4. Scale and shift를 수행한다.

   >> To Non-linearity

Batch Normalization

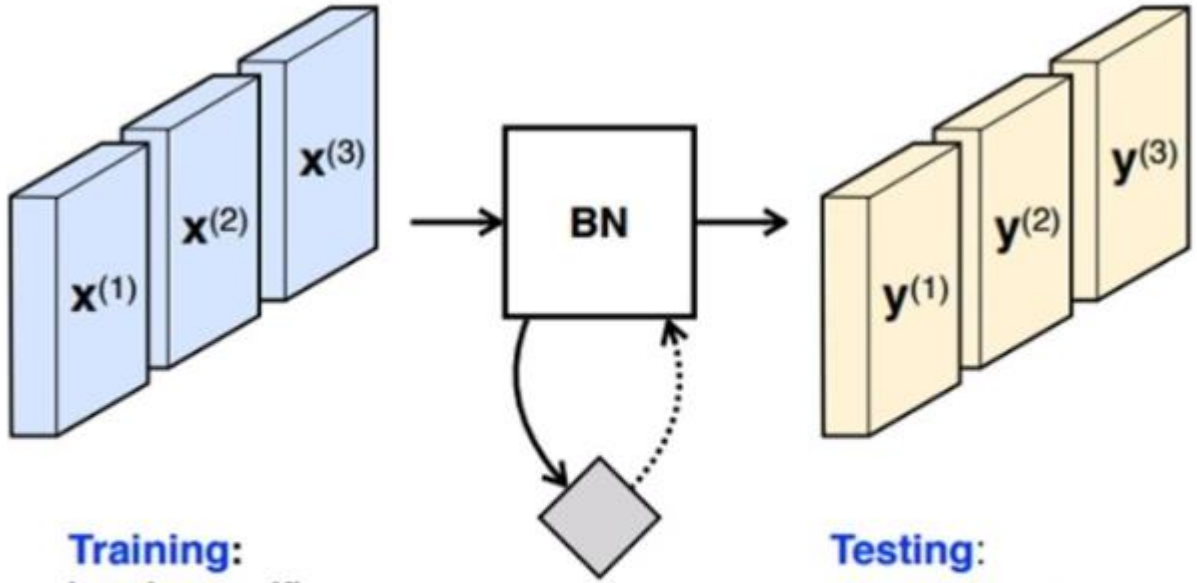
어떻게 적용시킬까?



사람마다 의견이 분분한데, Activation Function 앞쪽에 붙여야 한다는 사람과 Activation Function 뒤쪽에 붙여야 한다는 사람들이 있다.

Batch Normalization

Training & Test



Input: Network N with trainable parameters Θ;
subset of activations $\{x^{(k)}\}_{k=1}^{K}$

Output: Batch-normalized network for inference, $N_{\text{BN}}^{\text{inf}}$

1: $N_{\text{BN}}^{\text{tr}} \leftarrow N$ // Training BN network
2: **for** $k = 1 \ldots K$ **do**
3: Add transformation $y^{(k)} = \text{BN}_{\gamma^{(k)},\beta^{(k)}}(x^{(k)})$ to $N_{\text{BN}}^{\text{tr}}$ (Alg. 1)
4: Modify each layer in $N_{\text{BN}}^{\text{tr}}$ with input $x^{(k)}$ to take $y^{(k)}$ instead
5: **end for**
6: Train $N_{\text{BN}}^{\text{tr}}$ to optimize the parameters $\Theta \cup \{\gamma^{(k)}, \beta^{(k)}\}_{k=1}^{K}$
7: $N_{\text{BN}}^{\text{inf}} \leftarrow N_{\text{BN}}^{\text{tr}}$ // Inference BN network with frozen // parameters
8: **for** $k = 1 \ldots K$ **do**
9: // For clarity, $x \equiv x^{(k)}, \gamma \equiv \gamma^{(k)}, \mu_{\mathcal{B}} \equiv \mu_{\mathcal{B}}^{(k)}$, etc.
10: Process multiple training mini-batches $\mathcal{B}$, each of size m, and average over them:

$$\text{E}[x] \leftarrow \text{E}_{\mathcal{B}}[\mu_{\mathcal{B}}]$$
$$\text{Var}[x] \leftarrow \frac{m}{m-1}\text{E}_{\mathcal{B}}[\sigma_{\mathcal{B}}^2]$$

11: In $N_{\text{BN}}^{\text{inf}}$, replace the transform $y = \text{BN}_{\gamma,\beta}(x)$ with $y = \frac{\gamma}{\sqrt{\text{Var}[x]+\epsilon}} \cdot x + \left(\beta - \frac{\gamma\,\text{E}[x]}{\sqrt{\text{Var}[x]+\epsilon}}\right)$
12: **end for**

Overfitting

Weight decay

Simplest Way :

손실함수에 가중치의 제곱합만큼 더해주는 방법!

→ 가중치가 커지면 손실함수도 커지게 되는 효과!!

→ 손실함수는 작아지는 방향으로 움직이기 때문에 가중치가 함부로 커질수 없음!

Drop out

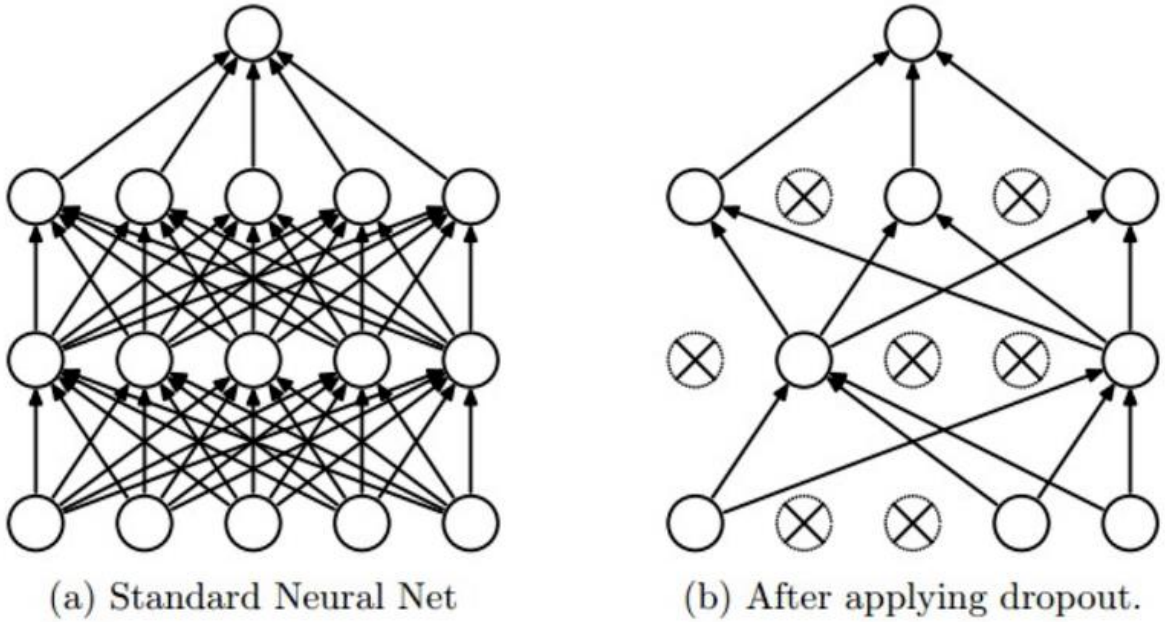
Regularization

Drop out :

1. 뉴런을 임의로 삭제하면서 학습하는 방법!

2. 은닉층의 뉴런을 무작위로 골라서 삭제!

→ Ensemble의 효과! (Prevent co-adaptation)

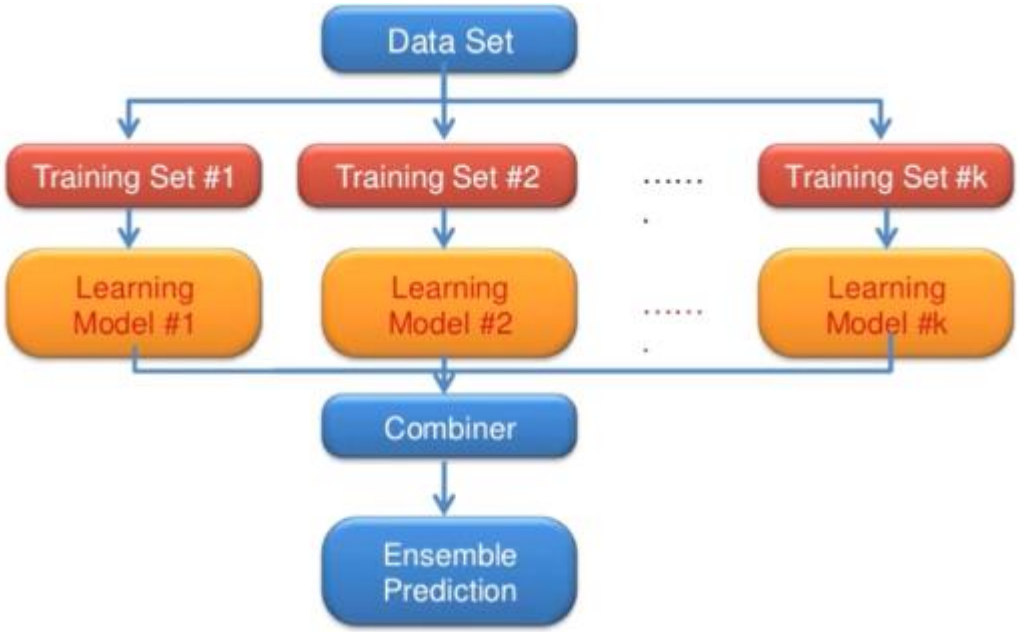


(a) Standard Neural Net

(b) After applying dropout.

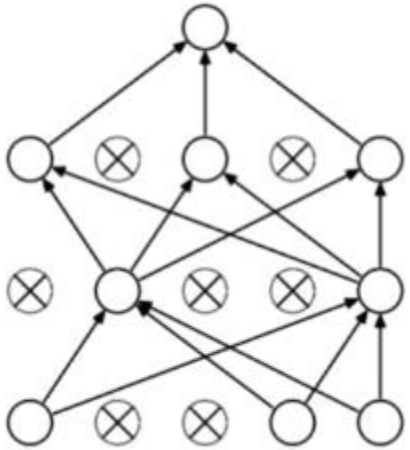
Note

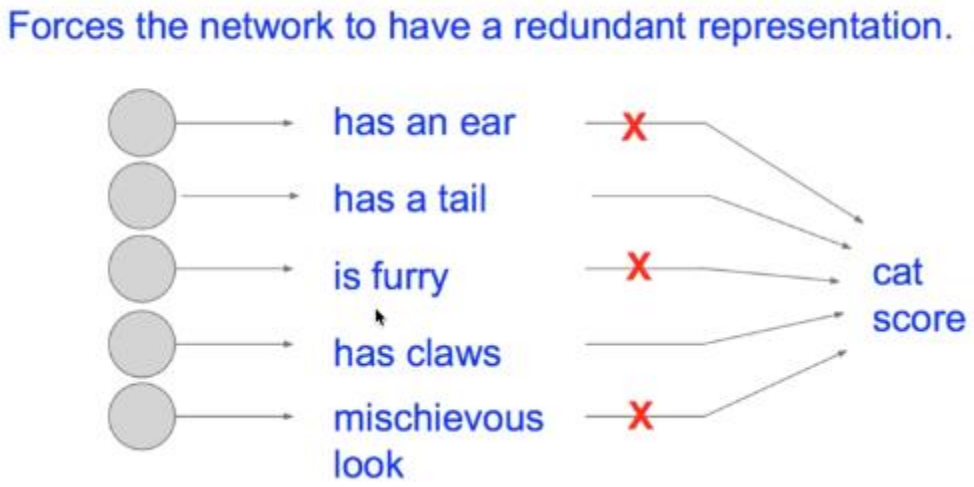
Ensemble



Drop out

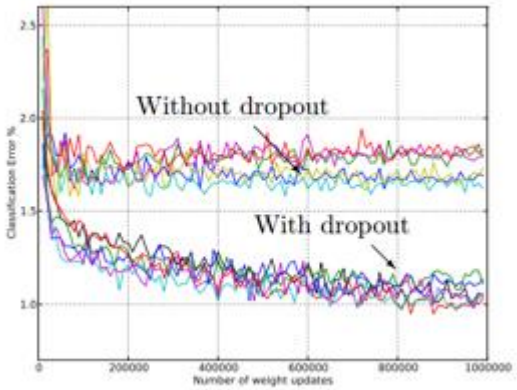
Regularization





Drop out

Effectiveness & Tips



MNIST

Tip)

Training 시에는 Drop out을 사용하지만,
Test할때는 Drop out을 사용하지 않는다.
즉, 모든 Node를 사용한다!

Any Question?

Thank You!