Training Tricks : Batch Normalization

Alvaro Soto

Computer Science Department, PUC

So far, we have already discussed some training tricks:

- Preprocessing: ex. mean substraction, normalization, etc.
- Dropout.
- Data augmentation.

Now, we will discuss a useful trick: Batch normalization.

From AI Class

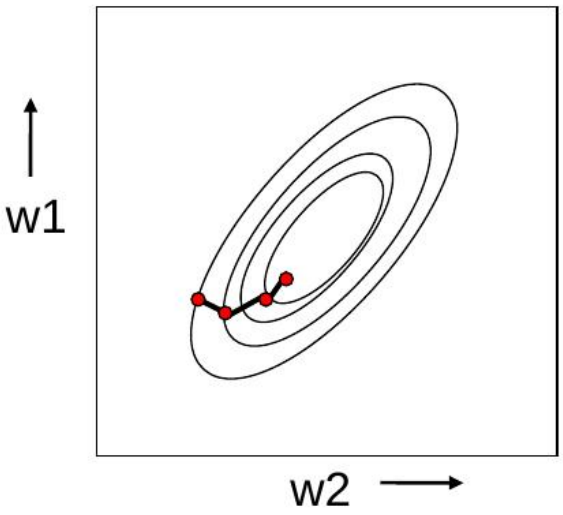
Entrenamiento de un Perceptrón

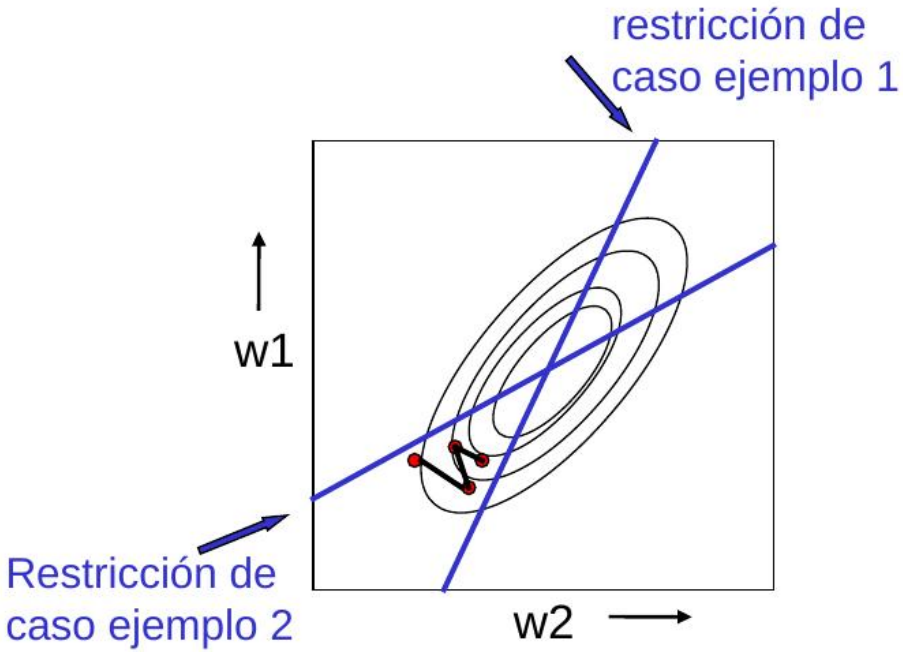
- Modo Batch

  sigue gradiente en superficie de error en set de training

- Modo incremental

  zig-zags por qué?





Stochastic gradient descent (SGD)

- Gradient of the loss over a mini-batch is an estimate of the gradient over the training set.
- Estimation quality improves as the batch size increases. Then, computation over a mini-batch is more robust than computations over individual examples.
- Also, computation over a mini-batch can be much more efficient than m computations over individual examples. Ex. due to parallelism.
- So mini-batch based training is great.
- We will see next that a smart normalization over each mini batch can have a drastic impact on several aspect of the training process.
- Let's first understand the covariate shift problem.

Covariate shift problem

Let's consider a 2 layer model:

$$NN_{2L} = f_2(f_1(x, \theta_1), \theta_2)$$

This is equivalent to a 2-step sequential process:

$$\bar{x} = f_1(x, \theta_1) \quad NN_{2L} = f_2(\bar{x}, \theta_2)$$

Using SGD, parameter θ_2 can be estimated (trained) iteratively by giving steps in the direction against the corresponding gradient, i.e.:

$$\theta_2 = \theta_2 - \frac{\eta}{m} \sum_{i}^{m} \frac{\partial f_2(\bar{x}_i, \theta_2)}{\partial \theta_2}$$

where:

η : learning rate.

m : mini batch size.

Any problem with the distribution of $\bar{x}$?

Covariate shift problem

- What is the most fundamental assumption of machine learning (ML)?
- What is the most relevant goal of a ML task?

We are traning θ_2 by using:

$$\theta_2 = \theta_2 - \frac{\eta}{m} \sum_i^m \frac{\partial f_2(\bar{x}_i, \theta_2)}{\partial \theta_2}$$

where:

$$\bar{x}_i = f_1(x_i, \theta_1)$$

Any problem with the distribution of $\bar{x}$?

- As a ML problem, it is desirable that the distribution of training and test data matches.
- However, the distribution of the input to each layer changes during training (why?).
- This slows down the training by requiring lower learning rates and careful parameter initialization.
- We would like to keep the distribution of $\bar{x}$ fixed over time.
- Then, θ_2 does not have to readjust to compensate for the change in the distribution of $\bar{x}$.

Batch Normalization (BN), Ioffe and Szegedy, 2015

- BN takes a step towards reducing the internal covariate shift problem between layers.
- Specifically, BN introduces a normalization step that fixes the means and variances of layer inputs.

$$\hat{x}^{(k)} = \frac{\bar{x}^{(k)} - E[\bar{x}^{(k)}]}{\sqrt{Var[\bar{x}^{(k)}]}} \tag{1}$$

- Expectation and variance are computed over the mini-batch for each dimension k. So each dimension is normalized independently.
- Therefore, BN normalizes each scalar feature independently trying to make them unit gaussian, i.e., each dimension has zero mean and unit variance.

Batch Normalization (BN)

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;

Output: $\{y_i = \text{BN}(x_i) \quad \}$

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

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

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

A small constant ϵ is added to the denominator to avoid an eventual division by zero.

Batch Normalization (BN)

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

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

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

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

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

- Simply normalizing each input of a layer might change the representational space of the layer.
- Ex. normalizing the inputs of a sigmoid would constrain them to lay in the linear part of the sigmoid.
- We need to add a mechanism to compensate for this effect.

Batch Normalization (BN)

- BN introduces a transformation that, if needed, allows the network to cancel the operation of the BN operator.
- In other words, the transformation provides the network with the flexibility to convert the BN operator in the identity function.
- Specifically, BN introduces for each activation $x^{(k)}$, parameters γ^k and β^k that scale and shift the normalized value.
- Using this transformaition, if needed, the network can recover the original activations by setting $\gamma^k = Var[\bar{x}^{(k)}]$ and $\beta^k = E[\bar{x}^{(k)}]$.

  Recall that: $\hat{x}^{(k)} = \frac{\bar{x}^{(k)} - E[\bar{x}^{(k)}]}{\sqrt{Var[\bar{x}^{(k)}]}}$
- Parameters γ^k and β^k are learned during training.

Batch Normalization (BN)

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma,\beta}(x_i)\}$

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

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

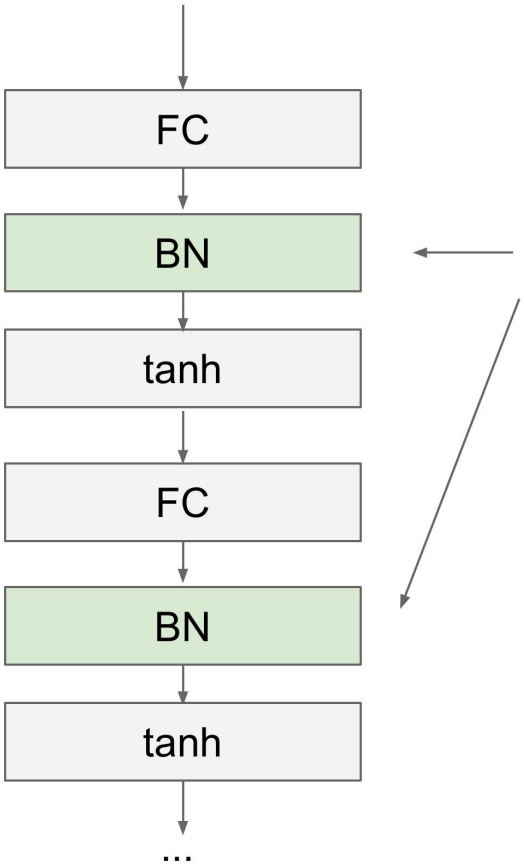
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

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

- What happens at test time?
- Do we have a mini batch?
- How mini-batch normalization operates at test time?

Mini-batches are normalized using all available data (or a suitable sample population) rather than using mini-batch statistics.

Batch Normalization (BN)



Usually inserted after Fully Connected or Convolutional layers, and before nonlinearity.

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

Batch Normalization (BN)

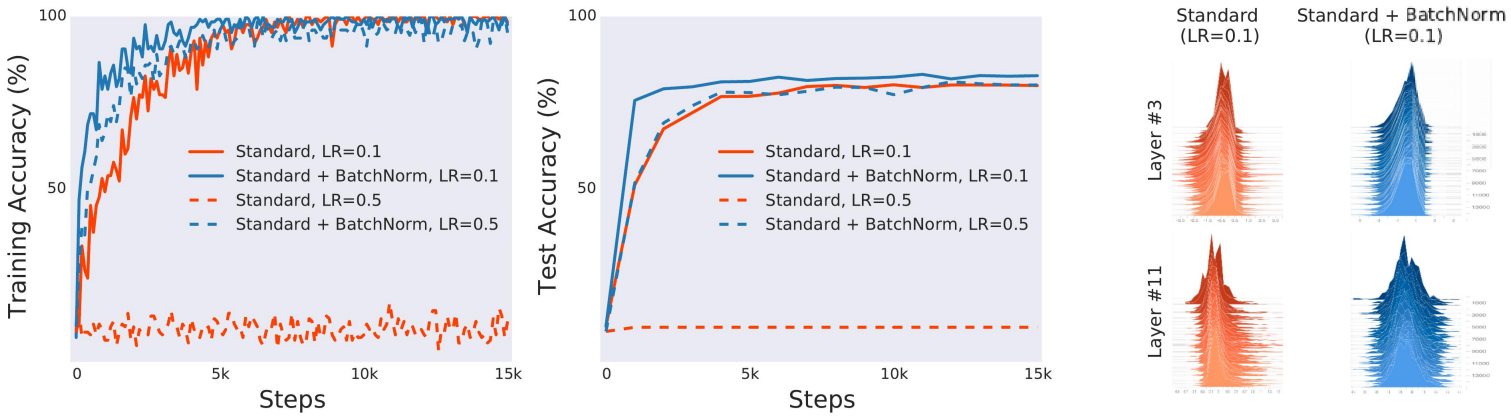


Figure 1: Comparison of (a) training (optimization) and (b) test (generalization) performance of a standard VGG network trained on CIFAR-10 with and without BatchNorm (details in Appendix A). There is a consistent gain in training speed in models with BatchNorm layers. (c) Even though the gap between the performance of the BatchNorm and non-BatchNorm networks is clear, the difference in the evolution of layer input distributions seems to be much less pronounced. (Here, we sampled activations of a given layer and visualized their distribution over training steps.)

How Does Batch Normalization Help Optimization? (No, It Is Not About Internal Covariate Shift)

Shibani Santurkar* MIT shibani@mit.edu

Dimitris Tsipras* MIT tsipras@mit.edu

Andrew Ilyas MIT ailyas@mit.edu

Aleksander Mądry MIT madry@mit.edu