https://machinelearningmastery.com/introduction-to-regularization-to-reduce-overfitting-and-improve-generalization-error/

How to Avoid Overfitting in Deep Learning Neural Networks

by **Jason Brownlee** on December 17, 2018 in **Deep Learning Performance**

Tweet**Share**

Last Updated on August 6, 2019

Training a deep neural network that can generalize well to new data is a challenging problem.

A model with too little capacity cannot learn the problem, whereas a model with too much capacity can learn it too well and overfit the training dataset. Both cases result in a model that does not generalize well.

A modern approach to reducing generalization error is to use a larger model that may be required to use regularization during training that keeps the weights of the model small. These techniques not only reduce overfitting, but they can also lead to faster optimization of the model and better overall performance.

In this post, you will discover the problem of overfitting when training neural networks and how it can be addressed with regularization methods.

After reading this post, you will know:

- Underfitting can easily be addressed by increasing the capacity of the network, but overfitting requires the use of specialized techniques.
- Regularization methods like weight decay provide an easy way to control overfitting for large neural network models.
- A modern recommendation for regularization is to use early stopping with dropout and a weight constraint.

Kick-start your project with my new book Better Deep Learning, including *step-by-step tutorials* and the *Python source code* files for all examples.

Let’s get started.

A Gentle Introduction to Regularization to Reduce Overfitting and Improve Generalization Error

Photo by jaimilee.beale, some rights reserved.

Overview

This tutorial is divided into four parts; they are:

1. The Problem of Model Generalization and Overfitting
2. Reduce Overfitting by Constraining Model Complexity
3. Methods for Regularization
4. Regularization Recommendations

The Problem of Model Generalization and Overfitting

The objective of a neural network is to have a final model that performs well both on the data that we used to train it (e.g. the training dataset) and the new data on which the model will be used to make predictions.

The central challenge in machine learning is that we must perform well on new, previously unseen inputs — not just those on which our model was trained. The ability to perform well on previously unobserved inputs is called generalization.

— Page 110, Deep Learning, 2016.

We require that the model learn from known examples and generalize from those known examples to new examples in the future. We use methods like a train/test split or k-fold cross-validation only to estimate the ability of the model to generalize to new data.

Learning and also generalizing to new cases is hard.

Too little learning and the model will perform poorly on the training dataset and on new data. The model will underfit the problem. Too much learning and the model will perform well on the training dataset and poorly on new data, the model will overfit the problem. In both cases, the model has not generalized.

- **Underfit Model**. A model that fails to sufficiently learn the problem and performs poorly on a training dataset and does not perform well on a holdout sample.
- **Overfit Model**. A model that learns the training dataset too well, performing well on the training dataset but does not perform well on a hold out sample.
- **Good Fit Model**. A model that suitably learns the training dataset and generalizes well to the old out dataset.

A model fit can be considered in the context of the bias-variance trade-off.

An underfit model has high bias and low variance. Regardless of the specific samples in the training data, it cannot learn the problem. An overfit model has low bias and high variance. The model learns the training data too well and performance varies widely with new unseen examples or even statistical noise added to examples in the training dataset.

In order to generalize well, a system needs to be sufficiently powerful to approximate the target function. If it is too simple to fit even the training data then generalization to new data is also likely to be poor. […] An overly complex system, however, may be able to approximate the data in many different ways that give similar errors and is unlikely to choose the one that will generalize best …

— Page 241, Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks, 1999.

We can address underfitting by increasing the capacity of the model. Capacity refers to the ability of a model to fit a variety of functions; more capacity, means that a model can fit more types of functions for mapping inputs to outputs. Increasing the capacity of a model is easily achieved by changing the structure of the model, such as adding more layers and/or more nodes to layers.

Because an underfit model is so easily addressed, it is more common to have an overfit model.

An overfit model is easily diagnosed by monitoring the performance of the model during training by evaluating it on both a training dataset and on a holdout validation dataset. Graphing line plots of the performance of the model during training, called learning curves, will show a familiar pattern.

For example, line plots of the loss (that we seek to minimize) of the model on train and validation datasets will show a line for the training dataset that drops and may plateau and a line for the validation dataset that drops at first, then at some point begins to rise again.

As training progresses, the generalization error may decrease to a minimum and then increase again as the network adapts to idiosyncrasies of the training data.

— Page 250, Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks, 1999.
A learning curve plot tells the story of the model learning the problem until a point at which it begins overfitting and its ability to generalize to the unseen validation dataset begins to get worse.

Reduce Overfitting by Constraining Model Complexity

There are two ways to approach an overfit model:

1. Reduce overfitting by training the network on more examples.
2. Reduce overfitting by changing the complexity of the network.

A benefit of very deep neural networks is that their performance continues to improve as they are fed larger and larger datasets. A model with a near-infinite number of examples will eventually plateau in terms of what the capacity of the network is capable of learning.

A model can overfit a training dataset because it has sufficient capacity to do so. Reducing the capacity of the model reduces the likelihood of the model overfitting the training dataset, to a point where it no longer overfits.

The capacity of a neural network model, it’s complexity, is defined by both it’s structure in terms of nodes and layers and the parameters in terms of its weights. Therefore, we can reduce the complexity of a neural network to reduce overfitting in one of two ways:

1. Change network complexity by changing the network structure (number of weights).
2. Change network complexity by changing the network parameters (values of weights).

In the case of neural networks, the complexity can be varied by changing the number of adaptive parameters in the network. This is called structural stabilization. […] The second principal approach to controlling the complexity of a model is through the use of regularization which involves the addition of a penalty term to the error function.

— Page 332, Neural Networks for Pattern Recognition, 1995.

For example, the structure could be tuned such as via grid search until a suitable number of nodes and/or layers is found to reduce or remove overfitting for the problem. Alternately, the model could be overfit and pruned by removing nodes until it achieves suitable performance on a validation dataset.

It is more common to instead constrain the complexity of the model by ensuring the parameters (weights) of the model remain small. Small parameters suggest a less complex and, in turn, more stable model that is less sensitive to statistical fluctuations in the input data.

Large weighs tend to cause sharp transitions in the [activation] functions and thus large changes in output for small changes in inputs.

— Page 269, Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks, 1999.

It is more common to focus on methods that constrain the size of the weights in a neural network because a single network structure can be defined that is under-constrained, e.g. has a much larger capacity than is required for the problem, and regularization can be used during training to ensure that the model does not overfit. In such cases, performance can even be better as the additional capacity can be focused on better learning generalizable concepts in the problem.

Techniques that seek to reduce overfitting (reduce generalization error) by keeping network weights small are referred to as regularization methods. More specifically, regularization refers to a class of approaches that add additional information to transform an ill-posed problem into a more stable well-posed problem.

A problem is said to be ill-posed if small changes in the given information cause large changes in the solution. This instability with respect to the data makes solutions unreliable *because small measurement errors or uncertainties in parameters may be greatly magnified and lead to wildly different responses. […] The idea behind regularization is to use supplementary information to restate an ill-posed problem in a stable form.*

— Page 266, Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks, 1999.

Regularization methods are so widely used to reduce overfitting that the term "*regularization*" may be used for any method that improves the generalization error of a neural network model.

Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error. Regularization is one of the central concerns of the field of machine learning, rivaled in its importance only by optimization.

— Page 120, Deep Learning, 2016.

Regularization Methods for Neural Networks

The simplest and perhaps most common regularization method is to add a penalty to the loss function in proportion to the size of the weights in the model.

1. **Weight Regularization (weight decay)**: Penalize the model during training based on the magnitude of the weights.

This will encourage the model to map the inputs to the outputs of the training dataset in such a way that the weights of the model are kept small. This approach is called weight regularization or weight decay and has proven very effective for decades for both simpler linear models and neural networks.

A simple alternative to gathering more data is to reduce the size of the model or improve regularization, by adjusting hyperparameters such as weight decay coefficients …

— Page 427, Deep Learning, 2016.

Below is a list of five of the most common additional regularization methods.

1. **Activity Regularization**: Penalize the model during training base on the magnitude of the activations.
2. **Weight Constraint**: Constrain the magnitude of weights to be within a range or below a limit.
3. **Dropout**: Probabilistically remove inputs during training.
4. **Noise**: Add statistical noise to inputs during training.
5. **Early Stopping**: Monitor model performance on a validation set and stop training when performance degrades.

Most of these methods have been demonstrated (or proven) to approximate the effect of adding a penalty to the loss function.

Each method approaches the problem differently, offering benefits in terms of a mixture of generalization performance, configurability, and/or computational complexity.

Regularization Recommendations

This section outlines some recommendations for using regularization methods for deep learning neural networks.

You should always consider using regularization, unless you have a very large dataset, e.g. big-data scale.

Unless your training set contains tens of millions of examples or more, you should include some mild forms of regularization from the start.

— Page 426, Deep Learning, 2016.
A good general recommendation is to design a neural network structure that is under-constrained and to use regularization to reduce the likelihood of overfitting.

… controlling the complexity of the model is not a simple matter of finding the model of the right size, with the right number of parameters. Instead, … in practical deep learning scenarios, we almost always do find—that the best fitting model (in the sense of minimizing generalization error) is a large model that has been regularized appropriately.

— Page 229, Deep Learning, 2016.
Early stopping should almost universally be used in addition to a method to keep weights small during training.

Early stopping should be used almost universally.

— Page 426, Deep Learning, 2016.
Some more specific recommendations include:

- **Classical**: use early stopping and weight decay (L2 weight regularization).
- **Alternate**: use early stopping and added noise with a weight constraint.
- **Modern**: use early stopping and dropout, in addition to a weight constraint.

These recommendations would suit Multilayer Perceptrons and Convolutional Neural Networks.

Some recommendations for recurrent neural nets include:

- **Classical**: use early stopping with added weight noise and a weight constraint such as maximum norm.
- **Modern**: use early stopping with a backpropagation-through-time-aware version of dropout and a weight constraint.

There are no silver bullets when it comes to regularization and systematic experimentation is strongly encouraged.

Summary

In this post, you discovered the problem of overfitting when training neural networks and how it can be addressed with regularization methods.

Specifically, you learned:

- Underfitting can easily be addressed by increasing the capacity of the network, but overfitting requires the use of specialized techniques.
- Regularization methods like weight decay provide an easy way to control overfitting for large neural network models.
- A modern recommendation for regularization is to use early stopping with dropout and a weight constraint.