**<https://machinelearningmastery.com/activation-regularization-for-reducing-generalization-error-in-deep-learning-neural-networks/>**

**A Gentle Introduction to Activation Regularization in Deep Learning**

by [**Jason Brownlee**](https://machinelearningmastery.com/author/jasonb/) on November 28, 2018 in [**Deep Learning Performance**](https://machinelearningmastery.com/category/better-deep-learning/)

Tweet Tweet  **Share** **Share**

Last Updated on August 6, 2019

Deep learning models are capable of automatically learning a rich internal representation from raw input data.

This is called feature or representation learning. Better learned representations, in turn, can lead to better insights into the domain, e.g. via visualization of learned features, and to better predictive models that make use of the learned features.

A problem with learned features is that they can be too specialized to the training data, or [overfit, and not generalize well to new examples](https://machinelearningmastery.com/introduction-to-regularization-to-reduce-overfitting-and-improve-generalization-error/). Large values in the learned representation can be a sign of the representation being overfit. Activity or representation regularization provides a technique to encourage the learned representations, the output or activation of the hidden layer or layers of the network, to stay small and sparse.

In this post, you will discover activation regularization as a technique to improve the generalization of learned features in neural networks.

After reading this post, you will know:

* Neural networks learn features from data and models, such as autoencoders and encoder-decoder models, explicitly seek effective learned representations.
* Similar to weights, large values in learned features, e.g. large activations, may indicate an overfit model.
* The addition of penalties to the loss function that penalize a model in proportion to the magnitude of the activations may result in more robust and generalized learned features.

**Kick-start your project** with my new book [Better Deep Learning](https://machinelearningmastery.com/better-deep-learning/), including *step-by-step tutorials* and the *Python source code* files for all examples.

Let’s get started.

.

**Overview**

This tutorial is divided into five parts; they are:

1. Problem With Learned Features
2. Encourage Small Activations
3. How to Encourage Small Activations
4. Examples of Activation Regularization
5. Tips for Using Activation Regularization

**Problem With Learned Features**

Deep learning models are able to perform feature learning.

That is, during the training of the network, the model will automatically extract the salient features from the input patterns or “*learn features*.” These features may be used in the network in order to predict a quantity for regression or predict a class value for classification.

These internal representations are tangible things. The output of a hidden layer within the network represent the learned features by the model at that point in the network.

There is a field of study focused on the efficient and effective automatic learning of features, often investigated by having a network reduce an input to a small learned feature before using a second network to reconstruct the original input from the learned feature. Models of this type are called auto-encoders, or encoder-decoders, and their learned features can be useful to learn more about the domain (e.g. via visualization) and in predictive models.

The learned features, or “*encoded inputs*,” must be large enough to capture the salient features of the input but also focused enough to not over-fit the specific examples in the training dataset. As such, there is a tension between the expressiveness and the generalization of the learned features.

*More importantly, when the dimension of the code in an encoder-decoder architecture is larger than the input, it is necessary to limit the amount of information carried by the code, lest the encoder-decoder may simply learn the identity function in a trivial way and produce uninteresting features.*

— [Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition](https://ieeexplore.ieee.org/document/4270182/), 2007.

In the same way that large weights in the network can signify an unstable and overfit model, large output values in the learned features can signify the same problems.

It is desirable to have small values in the learned features, e.g. small outputs or activations from the encoder network.

**Encourage Small Activations**

The loss function of the network can be updated to penalize models in proportion to the magnitude of their activation.

This is similar to “*weight regularization*” where the loss function is updated to penalize the model in proportion to the magnitude of the weights. The output of a layer is referred to as its ‘*activation*,’ as such, this form of penalty or regularization is referred to as ‘*activation regularization*‘ or ‘*activity regularization*‘

*… place a penalty on the activations of the units in a neural network, encouraging their activations to be sparse.*

— Page 254, [Deep Learning](https://amzn.to/2NJW3gE), 2016.

The output of an encoder or, generally, the output of a hidden layer in a neural network may be considered the representation of the problem at that point in the model. As such, this type of penalty may also be referred to as ‘*representation regularization*.’

The desire to have small activations or even very few activations with mostly zero values is also called a desire for [sparsity](https://machinelearningmastery.com/sparse-matrices-for-machine-learning/). As such, this type of penalty is also referred to as ‘*sparse feature learning*.’

*One way to limit the information content of an overcomplete code is to make it sparse.*

— [Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition](https://ieeexplore.ieee.org/document/4270182/), 2007.

The encouragement of sparse learned features in autoencoder models is referred to as ‘*sparse autoencoders*.’

*A sparse autoencoder is simply an autoencoder whose training criterion involves a sparsity penalty on the code layer, in addition to the reconstruction error*

— Page 505, [Deep Learning](https://amzn.to/2NJW3gE), 2016.

Sparsity is most commonly sought when a larger-than-required hidden layer (e.g. over-complete) is used to learn features that may encourage over-fitting. The introduction of a sparsity penalty counters this problem and encourages better generalization.

A sparse overcomplete learned feature has been shown to be more effective than other types of learned features offering better robustness to noise and even transforms in the input, e.g. learned features of images may have improved invariance to the position of objects in the image.

*Sparse-overcomplete representations have a number of theoretical and practical advantages, as demonstrated in a number of recent studies. In particular, they have good robustness to noise, and provide a good tiling of the joint space of location and frequency. In addition, they are advantageous for classifiers because classification is more likely to be easier in higher dimensional spaces.*

— [Sparse Feature Learning for Deep Belief Networks](http://papers.nips.cc/paper/3363-sparse-feature-learning-for-deep-belief-networks), 2007.

There is a general focus on sparsity of the representations rather than small vector magnitudes. A study of these representations that is more general than the use of neural networks is known as ‘*sparse coding*.’

*Sparse coding provides a class of algorithms for finding succinct representations of stimuli; given only unlabeled input data, it learns basis functions that capture higher-level features in the data.*

— [Efficient sparse coding algorithms](https://dl.acm.org/citation.cfm?id=2976557), 2007.

**How to Encourage Small Activations**

An activation penalty can be applied per-layer, perhaps only at one layer that is the focus of the learned representation, such as the output of the encoder model or the middle (bottleneck) of an autoencoder model.

A constraint can be applied that adds a penalty proportional to the magnitude of the vector output of the layer.

The activation values may be positive or negative, so we cannot simply sum the values.

Two common methods for calculating the magnitude of the activation are:

* Sum of the absolute activation values, called [l1 vector norm](https://machinelearningmastery.com/vector-norms-machine-learning/).
* Sum of the squared activation values, called the [l2 vector norm](https://machinelearningmastery.com/vector-norms-machine-learning/).

The L1 norm encourages sparsity, e.g. allows some activations to become zero, whereas the l2 norm encourages small activations values in general. Use of the L1 norm may be a more commonly used penalty for activation regularization.

A hyperparameter must be specified that indicates the amount or degree that the loss function will weight or pay attention to the penalty. Common values are on a logarithmic scale between 0 and 0.1, such as 0.1, 0.001, 0.0001, etc.

Activity regularization can be used in conjunction with other regularization techniques, such as weight regularization.

**Examples of Activation Regularization**

This section provides some examples of activation regularization in order to provide some context for how the technique may be used in practice.

Regularized or sparse activations were originally sought as an approach to support the development of much deeper neural networks, early in the history of deep learning. As such, many examples may make use of architectures like restricted Boltzmann machines (RBMs) that have been replaced by more modern methods. Another big application of weight regularization is in autoencoders with semi-labeled or unlabeled data, so-called sparse autoencoders.

Xavier Glorot, et al. at the University of Montreal introduced the use of the rectified linear activation function to encourage sparsity of representation. They used an L1 penalty and evaluate deep supervised MLPs on a range of classical computer vision classification tasks such as MNIST and CIFAR10.

*Additionally, an L1 penalty on the activations with a coefficient of 0.001 was added to the cost function during pre-training and fine-tuning in order to increase the amount of sparsity in the learned representations*

— [Deep Sparse Rectifier Neural Networks](http://proceedings.mlr.press/v15/glorot11a.html), 2011.

Stephen Merity, et al. from Salesforce Research used L2 activation regularization with LSTMs on outputs and recurrent outputs for natural language process in conjunction with dropout regularization. They tested a suite of different activation regularization coefficient values on a range of language modeling problems.

*While simple to implement, activity regularization and temporal activity regularization are competitive with other far more complex regularization techniques and offer equivalent or better results.*

— [Revisiting Activation Regularization for Language RNNs](https://arxiv.org/abs/1708.01009), 2017.

**Tips for Using Activation Regularization**

This section provides some tips for using activation regularization with your neural network.

**Use With All Network Types**

Activation regularization is a generic approach.

It can be used with most, perhaps all, types of neural network models, not least the most common network types of Multilayer Perceptrons, Convolutional Neural Networks, and Long Short-Term Memory Recurrent Neural Networks.

**Use With Autoencoders and Encoder-Decoders**

Activity regularization may be best suited to those model types that explicitly seek an efficient learned representation.

These include models such as autoencoders (i.e. sparse autoencoders) and encoder-decoder models, such as encoder-decoder LSTMs used for sequence-to-sequence prediction problems.

**Experiment With Different Norms**

The most common activation regularization is the L1 norm as it encourages sparsity.

Experiment with other types of regularization such as the L2 norm or using both the L1 and L2 norms at the same time, e.g. like the Elastic Net linear regression algorithm.

**Use Rectified Linear**

The [rectified linear activation function, also called relu](https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/), is an activation function that is now widely used in the hidden layer of deep neural networks.

Unlike classical activation functions such as tanh (hyperbolic tangent function) and sigmoid (logistic function), the relu function allows exact zero values easily. This makes it a good candidate when learning sparse representations, such as with the l1 vector norm activation regularization.

**Grid Search Parameters**

It is common to use small values for the regularization hyperparameter that controls the contribution of each activation to the penalty.

Perhaps start by testing values on a log scale, such as 0.1, 0.001, and 0.0001. Then use a grid search at the order of magnitude that shows the most promise.

**Standardize Input Data**

It is a generally good practice to rescale input variables to have the same scale.

When input variables have different scales, the scale of the weights of the network will, in turn, vary accordingly. Large weights can saturate the nonlinear transfer function and reduce the variance in the output from the layer. This may introduce a problem when using activation regularization.

This problem can be addressed by either normalizing or standardizing input variables.

**Use an Overcomplete Representation**

Configure the layer chosen to be the learned features, e.g. the output of the encoder or the bottleneck in the autoencoder, to have more nodes that may be required.

This is called an overcomplete representation that will encourage the network to overfit the training examples. This can be countered with a strong activation regularization in order to encourage a rich learned representation that is also sparse.