**Better learning techniques**

Configure capacity – no of hidden nodes and no of layers

Configure batch size- batch , stochastic, mini batch

Configure loss functions

Configure learning rate- whether modern adaptive learning rates would be appropriate

Data scaling – scaling of inputs at deeper layers, sensitivity of weights to the scale of inputs

Batch normalization-benefits of standardizing layer inputs to add consistency of input and stability to the learning process

Stochastic gradient descent is a general optimization algorithm that can be applied to a wide range of problems. Nevertheless, the optimization process (or learning process) can become unstable and specific interventions are required; for example:

Vanishing Gradients.- solution relu function

Exploding gradient -solution gradient clipping or gradient normalization -Large weight updates cause a numerical overflow or underflow making the network weights take on a NaN or Inf value

The lack of data on some predictive modeling problems can prevent effective learning. Specialized techniques can be used to jump-start the optimization process, providing a useful initial set of weights or even whole models that can be used for feature extraction

 Greedy Layer-Wise Pretraining

Transfer Learning

**Better regularization techniques**

Weight regularization - A change to the loss function that penalizes a model in proportion to the norm (magnitude) of the model weights, encouraging smaller weights and, in turn, a lower complexity model

Weight Constraint.- Update to the model to rescale the weights when the vector norm of the weights exceeds a threshold

 Dropout.

Early Stopping.

Input Noise. Addition of statistical variation or noise at the input layer or between hidden layers to reduce the model’s dependence on specific input values

Early Stopping. Monitor model performance on the hold out validation dataset during training and stop the training process when performance on the validation set starts to degrade

**Better prediction techniques**

Vary the training data used to fit each member.  Vary the members that contribute to the ensemble prediction.  Vary the way that the predictions from the ensemble members are combined.

Resampling Ensemble. Ensemble of models fit on different samples of the training dataset

Model Averaging Ensemble-Retrain models across multiple runs of the same learning algorithm on the same dataset

Weighted Average Ensemble (blending)- The contribution from each ensemble member to an ensemble prediction is weighted using learned coefficients that indicates the trust in each model

**Framework**

Step 1: Diagnose the Performance Problem

If the loss on the training dataset is poor, stuck, or fails to improve, perhaps you have a learning problem. 

If the loss or problem-specific metric on the training dataset continues to improve and gets worse on the validation dataset, perhaps you have a generalization problem. 

If the loss or problem-specific metric on the validation dataset shows a high variance towards the end of the run, perhaps you have a prediction problem.

Step 2: Select and Evaluate a Technique

Learning Problem: Tuning the hyperparameters of the learning algorithm; specifically, the learning rate offers the biggest leverage.

 Generalization Problem: Using weight regularization and early stopping works well on most models with most problems, or try dropout with early stopping.

 Prediction Problem: Average the prediction from models collected over multiple runs or multiple epochs on one run to add sufficient bias.

Step 3: Go To Step 1

Once you have identified an issue and addressed it with an intervention, repeat the process. Developing a better model is an iterative process that may require multiple interventions at multiple levels that complement each other. This is an empirical process.

**Learning curves**

A learning curve is a plot of model learning performance over experience or time

Learning curves are a widely used diagnostic tool in machine learning for algorithms that learn from a training dataset incrementally.

* Learning curves are plots that show changes in learning performance over time in terms of experience. 
* Learning curves of model performance on the train and validation datasets can be used to diagnose an underfit, overfit, or well-fit model. 
* Learning curves of model performance can be used to diagnose whether the train or validation datasets are not relatively representative of the problem domain.

Learning Curve: Line plot of learning (y-axis) over experience (x-axis)

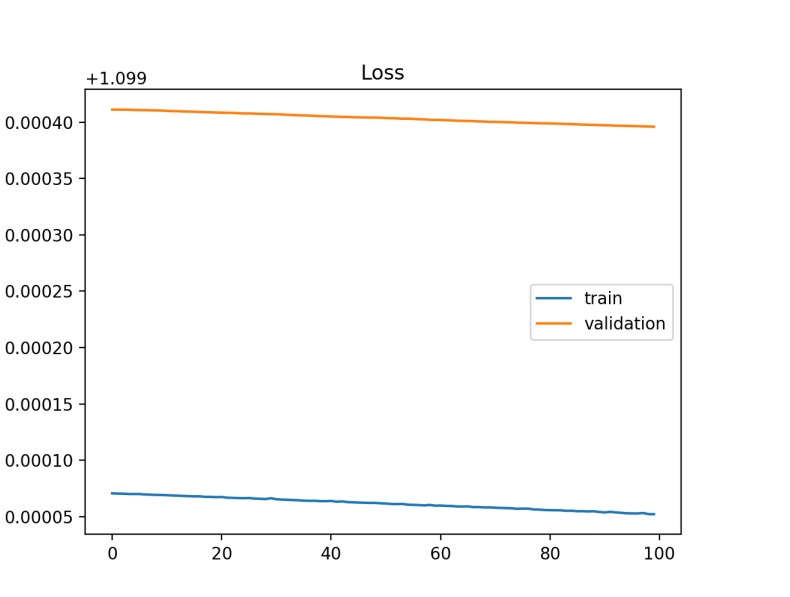
* Train Learning Curve: Learning curve calculated from the training dataset that gives an idea of how well the model is learning.
* Validation Learning Curve: Learning curve calculated from a hold-out validation dataset that gives an idea of how well the model is generalizing.
* Optimization Learning Curves: Learning curves calculated on the metric by which the parameters of the model are being optimized, e.g. loss. 
* Performance Learning Curves: Learning curves calculated on the metric by which the model will be evaluated and selected, e.g. accuracy.

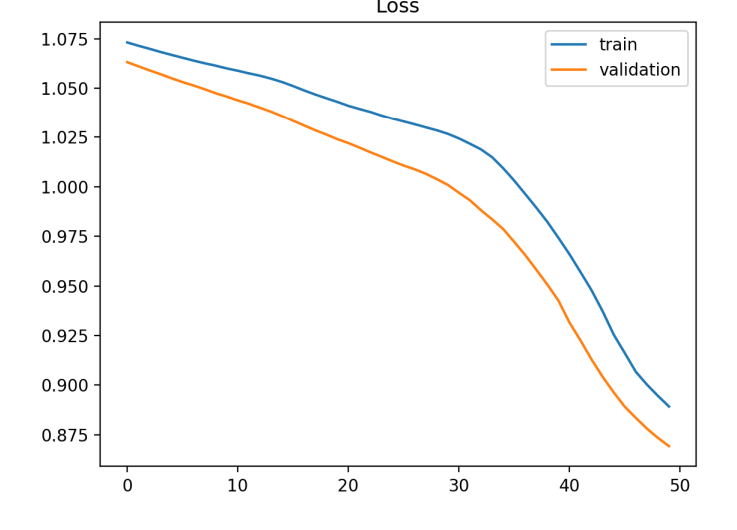
**Diagnosing Model Behavior**

There are three common dynamics that you are likely to observe in learning curves; they are:  Underfit.  Overfit.  Good Fit+

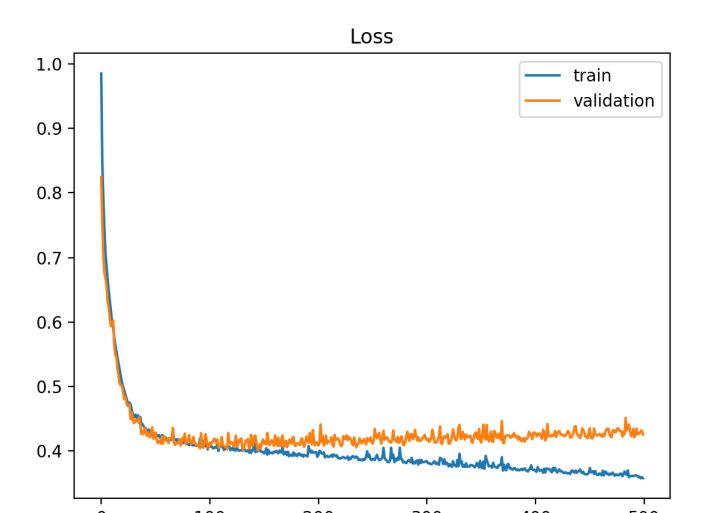
Underfit Learning Curves Underfitting refers to a model that cannot learn the training dataset

An underfit model can be identified from the learning curve of the training loss only. It may show a flat line or noisy values of relatively high loss, indicating that the model was unable to learn the training dataset at all.

****

****

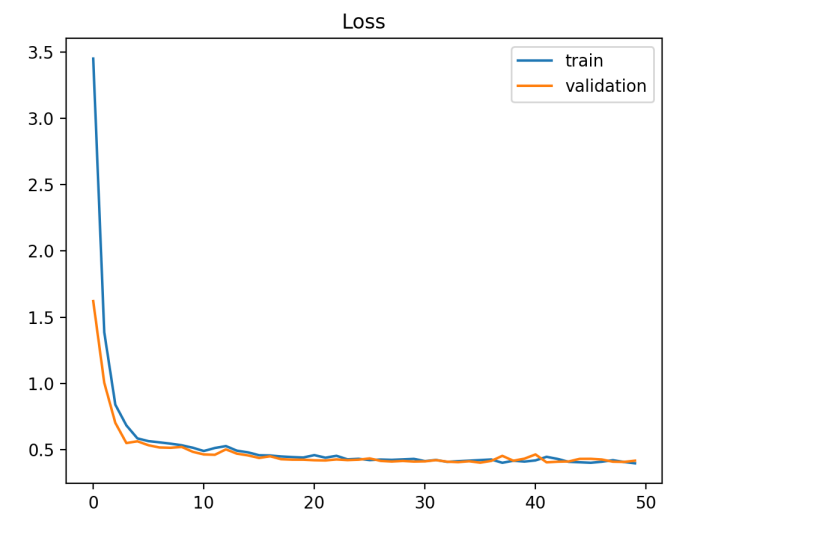
Overfit Learning Curves Overfitting refers to a model that has learned the training dataset too well, including the statistical noise or random fluctuations in the training dataset

****

Good Fit Learning Curves

The plot of training loss decreases to a point of stability. 

The plot of validation loss decreases to a point of stability and has a small gap with the training loss.

****

Training a deep learning neural network model using stochastic gradient descent with backpropagation involves choosing a number of components and hyperparameters, they are:  Network Topology. 

Loss Function. 

Weight Initialization. 

Batch Size. 

Learning Rate. 

Epochs. 

Data Preparation

The capacity of a neural network can be controlled by two aspects of the model:

 Number of Nodes.

 Number of Layers.

A model with more nodes or more layers has a greater capacity and, in turn, is potentially capable of navigating a larger set of mapping functions. The number of nodes in a layer is referred to as the width. in practice, we don’t know how many nodes are sufficient or how to train such a model. The number of layers in a model is referred to as its depth. Increasing the depth increases the capacity of the model. Training deep models, e.g. those with many hidden layers, can be computationally more efficient than training a single layer network with a vast number of nodes.

**Nodes and Layers Keras API**

.. layer = Dense(32)

Example of specifying the number of nodes for a Dense layer.

layer = LSTM(32)

Example of specifying the number of nodes for an LSTM layer.

layer = Conv2D(32, (3,3))

Example of specifying the number of filter maps for a CNN layer.

Configuring Model Layers

model = Sequential()

model.add(Dense(32))

model.add(Dense(64))

model = Sequential()

model.add(LSTM(32, return\_sequences=True))

model.add(LSTM(32))

Example

The capacity of a neural network can be controlled by two aspects of the model:

 Number of Nodes.

 Number of Layers.

**Configure** gradient precision with batch size

Neural networks are trained using gradient descent where the estimate of the error used to update the weights is calculated based on a subset of the training dataset. The number of examples from the training dataset used in the estimate of the error gradient is called the batch size and is an important hyperparameter that influences the dynamics of the learning algorithm

Batch size controls the accuracy of the estimate of the error gradient when training neural networks.  Batch, Stochastic, and Minibatch gradient descent are the three main flavors of the learning algorithm.

Batch Gradient Descent. Batch size is set to the total number of examples in the training dataset.

 Stochastic Gradient Descent or online gradient descent. Batch size is set to one. 

Minibatch Gradient Descent. Batch size is set to more than one and less than the total number of examples in the training dataset

For shorthand, the algorithm is often referred to as stochastic gradient descent regardless of the batch size . Given that very large datasets are often used to train deep learning neural networks, the batch size is rarely set to the size of the training dataset

Smaller batch sizes are noisy, offering a regularizing effect and lower generalization error.

 Smaller batch sizes make it easier to fit one batch worth of training data in memory (i.e. when using a GPU that has access to less local memory than system RAM).

Small batch sizes such as 32 do work well generally.

model.fit(trainX, trainy, batch\_size=1)

Example of stochastic gradient descent in Keras.

model.fit(trainX, trainy, batch\_size=len(trainX))

model.fit(trainX, trainy, batch\_size=64)

Example of minibatch gradient descent in Keras.

Batch Gradient Descent: Use a relatively larger learning rate and more training epochs.

 Stochastic Gradient Descent: Use a relatively smaller learning rate and fewer training epochs.

**Configure What to Optimize with Loss Functions**

There are many loss functions to choose from and it can be challenging to know what to choose, or even what a loss function is and the role it plays when training a neural network

Typically, with neural networks, we seek to minimize the error. As such, the objective function is often referred to as a cost function or a loss function and the value calculated by the loss function is referred to as simply

Loss Functions

Maximum Likelihood- Maximum likelihood seeks to find the optimum values for the parameters by maximizing a likelihood function derived from the training data.

Under the maximum likelihood framework, the error between two probability distributions is measured using cross-entropy. When modeling a classification problem where we are interested in mapping input variables to a class label, we can model the problem as predicting the probability of an example belonging to each class

A few basic functions are very commonly used. The mean squared error is popular for function approximation (regression) problems [...] The cross-entropy error function is often used for classification problems when outputs are interpreted as probabilities of membership in an indicated class.

Regression Problem

A problem where you predict a real-value quantity.

 Output Layer Configuration: One node with a linear activation unit.

 Loss Function: Mean Squared Error (MSE).

Binary Classification Problem

Output Layer Configuration:

One node with a sigmoid activation unit.

 Loss Function: Cross-Entropy, also referred to as Logarithmic loss.

Multiclass Classification Problem

Output Layer Configuration: One node for each class using the softmax activation function.  Loss Function: Cross-Entropy, also referred to as Logarithmic loss

Loss functions

Mean squared error loss

Mean squared log error loss- It has the effect of relaxing the punishing effect of large differences in large predicted values

Mean Absolute Error Loss- On some regression problems, the distribution of the target variable may be mostly Gaussian, but may have outliers, e.g. large or small values far from the mean value. The Mean Absolute Error, or MAE, loss is an appropriate loss function in this case as it is more robust to outliers

Classification problem

Binary Cross-Entropy Loss- Cross-entropy is the default loss function to use for binary classification problems. It is intended for use with binary classification where the target values are in the set {0, 1}. Mathematically, it is the preferred loss function under the inference framework of maximum likelihood

Cross-entropy will calculate a score that summarizes the average difference between the actual and predicted probability distributions for predicting class 1. The score is minimized and a perfect crossentropy value is 0

Hinge Loss

An alternative to cross-entropy for binary classification problems is the hinge loss function, primarily developed for use with Support Vector Machine (SVM) models. It is intended for use with binary classification where the target values are in the set {-1, 1}

The hinge loss function has many extensions, often the subject of investigation with SVM models. A popular extension is called the squared hinge loss

Configure Speed of Learning with Learning Rate

The amount of change to the model during each step of this search process, or the step size, is called the learning rate and provides perhaps the most important hyperparameter to tune for your neural network in order to achieve good performance on your problem

learning rate is a configurable hyperparameter used in the training of neural networks that has a small positive value, often in the range between 0.0 and 1.0

When the learning rate is too large, gradient descent can inadvertently increase rather than decrease the training error. [...] When the learning rate is too small, training is not only slower, but may become permanently stuck with a high training error. —

The method of momentum is designed to accelerate learning, especially in the face of high curvature, small but consistent gradients, or noisy gradients.

The amount of inertia of past updates is controlled via the addition of a new hyperparameter, often referred to as the momentum or velocity and uses the notation of the Greek lowercase letter alpha (α)

Common values of [momentum] used in practice include .5, .9, and .99

**Use a Learning Rate Schedule**

An alternative to using a fixed learning rate is to instead vary the learning rate over the training process. The way in which the learning rate changes over time (training epochs) is referred to as the learning rate schedule or learning rate decay

A reasonable choice of optimization algorithm is SGD with momentum with a decaying learning rate (popular decay schemes that perform better or worse on different problems include decaying linearly until reaching a fixed minimum learning rate, decaying exponentially, or decreasing the learning rate by a factor of 2-10 each time validation error plateaus).

The difficulty of choosing a good learning rate a priori is one of the reasons adaptive learning rate methods are so useful and popular. A good adaptive algorithm will usually converge much faster than simple back-propagation with a poorly chosen fixed learning rate.

there are three adaptive learning rate methods that have proven to be robust over many types of neural network architectures and problem types. They are AdaGrad, RMSProp, and Adam, and all maintain and adapt learning rates for each of the weights in the model.

**https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6**