CSE/ECE-471: Statistical Methods in AI **Spring 2020**

Neural Network Checklist

Prepared by: 2019201942, 2019201034, 2019900085

1. **Input Type**

   Input can be vector, matrix, tensor, sequence

2. **Network Type**

   There are three types of networks they are, MLP, CNN, RNN

   Multilayer Perceptrons: Use MLP for the following

   Tabular datasets

   Classification prediction problems

   Regression prediction problems

   Convolutional Neural Networks: Use CNN for the following

   Image data

   Classification prediction problems

   Regression prediction problems

   Recurrent Neural Networks: Use RNN for the following

   Text data

   Speech data

   Classification prediction problems

   Regression prediction problems

   Generative models

   Dont Use RNN for the following

   Tabular data

   Image data

3. **Output Type**

   Output can be categorical(single class, multi class), real number, scalar, vector, matrix, tensor

4. **Task Type**

   Classification

   Regression

5. **Architecture**

   Artificial neural networks have two main hyper parameters that control the architecture or topology of the network: the number of layers and the number of nodes in each hidden layer.

   The most reliable way to configure these hyper parameters for your specific predictive modeling problem is via systematic experimentation with a robust test harness.

   Why Have Multiple Layers?

   Before we look at how many layers to specify, it is important to think about why we would want to have multiple layers. A single-layer neural network can only be used to represent linearly separable functions. This means very simple problems where, say, the two classes in a classification problem can be neatly separated by a line. If your problem is relatively simple, perhaps a single

layer network would be sufficient.
Most problems that we are interested in solving are not linearly separable.
A Multilayer Perceptron can be used to represent convex regions. This means that in effect, they can learn to draw shapes around examples in some high-dimensional space that can separate and classify them, overcoming the limitation of linear separability.

Type of layers

Convolution Layer

Pooling Layer

Fully connected

Regularizing

batchnorm, dropout

Type of Activation function

6. Optimization

Loss function helps in optimizing the parameters of the neural networks. Our objective is to minimize the loss for a neural network by optimizing its parameters(weights). The loss is calculated using loss function by matching the target(actual) value and predicted value by a neural network. Then we use the gradient descent method to update the weights of the neural network such that the loss is minimized. This is how we train a neural network.

Cross-Entropy Loss: Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label. So predicting a probability of .012 when the actual observation label is 1 would be bad and result in a high loss value. A perfect model would have a log loss of 0.

The graph above shows the range of possible loss values given a true observation (isDog = 1).

As the predicted probability approaches 1, log loss slowly decreases. As the predicted probability decreases, however, the log loss increases rapidly. Log loss penalizes both types of errors, but especially those predictions that are confident and wrong!

Cross-entropy and log loss are slightly different depending on context, but in machine learning when calculating error rates between 0 and 1 they resolve to the same thing.

In binary classification, where the number of classes M equals 2, cross-entropy can be calculated as:

$$-(y\log(p) + (1-y)\log(1-p))$$

If M greater than 2 (i.e. multiclass classification), we calculate a separate loss for each class label per observation and sum the result.

$$-\sum_{c=1}^{M} yo, c\log(po, c)$$

MSE Loss: Mean Squared Error loss, or MSE for short, is calculated as the average of the squared differences between the predicted and actual values. The result is always positive regardless of the sign of the predicted and actual values and a perfect value is 0.0. The loss value is minimized, although it can be used in a maximization optimization process by making the score negative.

Optimizer(Adam, Adagrad, RMSProp): Optimizers are algorithms or methods used to change the attributes of your neural network such as weights and learning rate in order to reduce the losses.

Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data. The name Adam is derived from adaptive moment estimation.

Adaptive Gradient Algorithm (AdaGrad) that maintains a per-parameter learning rate that improves performance on problems with sparse gradients (e.g. natural language and computer vision problems).

Root Mean Square Propagation (RMSProp) that also maintains per-parameter learning rates that are adapted based on the average of recent magnitudes of the gradients for the weight (e.g. how quickly it is changing). This means the algorithm does well on online and non-stationary problems (e.g. noisy).
Comparison b/w different optimizers

Hyper-parameters: In machine learning, a hyperparameter is a parameter whose value is set before the learning process begins. By contrast, the values of other parameters are derived via training.

Hyperparameters can be classified as model hyperparameters, that cannot be inferred while fitting the machine to the training set because they refer to the model selection task, or algorithm hyperparameters, that in principle have no influence on the performance of the model but affect the speed and quality of the learning process. An example of the first type is the topology and size of a neural network. An example of the second type is learning rate or mini-batch size.

learning-rate: Specifically, the 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. The learning rate controls how quickly the model is adapted to the problem.

momentum-weight: Neural network momentum is a simple technique that often improves both training speed and accuracy. Training a neural network is the process of finding values for the weights and biases so that for a given set of input values, the computed output values closely match the known, correct, target values.

7. Training

In this era of deep learning, where machines have already surpassed human intelligence it's fascinating to see how these machines are learning just by looking at examples. When we say that we are training the model, it's gradient descent behind the scenes who trains it.

Gradient descent is a first-order iterative optimization algorithm for finding the minimum of a function.

Types of gradient descent are :

- Batch Gradient Descent
- Stochastic Gradient Descent
- Mini Batch Gradient Descent

Batch Gradient Descent

In Batch Gradient Descent, all the training data is taken into consideration to take a single step. We take the average of the gradients of all the training examples and then use that mean gradient to update our parameters. So that's just one step of gradient descent in one epoch.

Mini Batch Gradient Descent
Neither we use all the dataset all at once nor we use the single example at a time. We use a batch of a fixed number of training examples which is less than the actual dataset and call it a mini-batch. Doing this helps us achieve the advantages of both the former variants we saw. So, after creating the mini-batches of fixed size, we do the following steps in one epoch:

1 . Pick a mini-batch
2 . Feed it to Neural Network
3 . Calculate the mean gradient of the mini-batch
4 . Use the mean gradient we calculated in step 3 to update the weights
5 . Repeat steps 1–4 for the mini-batches we created

Stochastic Gradient Descent
In Stochastic Gradient Descent (SGD), we consider just one example at a time to take a single step. We do the following steps in one epoch for SGD:

1 . Take an example
2 . Feed it to Neural Network
3 . Calculate it's gradient
4 . Use the gradient we calculated in step 3 to update the weights
5 . Repeat steps 1–4 for all the examples in training dataset

Number of Epochs
The number of epochs is the number of complete passes through the training dataset. The size of a batch must be more than or equal to one and less than or equal to the number of samples in the training dataset. The number of epochs can be set to an integer value between one and infinity.

8. Data Split(Train, Validation, Split)

While training a model is a key step, how the model generalizes on unseen data is an equally important aspect that should be considered in every machine learning pipeline. We need to know whether it actually works and, consequently, if we can trust its predictions.
The above issues can be handled by evaluating the performance of a machine learning model. Model evaluation aims to estimate the generalization accuracy of a model on future (unseen/out-of-sample) data.

Training set is a subset of the dataset used to build predictive models.

Validation set is a subset of the dataset used to assess the performance of the model built in the training phase. It provides a test platform for fine-tuning a model's parameters and selecting the best performing model. Not all modeling algorithms need a validation set.

Test set, or unseen data, is a subset of the dataset used to assess the likely future performance of a model. If a model fits to the training set much better than it fits the test set, over-fitting is probably the cause.

9. Evaluation Measures

Model evaluation metrics are required to quantify model performance. The choice of evaluation metrics depends on a given machine learning task (such as classification, regression, ranking, clustering, topic modeling, among others).

Accuracy: Model accuracy in terms of classification models can be defined as the ratio of correctly classified samples to the total number of samples.

Precision: In a classification task, the precision for a class is the number of true positives divided by the total number of elements labeled as belonging to the positive class.
High precision means that an algorithm returned substantially more relevant results than irrelevant ones.

Recall: is defined as the number of true positives divided by the total number of elements that actually belong to the positive class.
High recall means that an algorithm returned most of the relevant results.

F-1 score: is the harmonic mean of the precision and recall, where an F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0.

MSE: is the average of the squared error that is used as the loss function. It is the sum, over all the data points, of the square of the difference between the predicted and actual target variables, divided by the number of data points. RMSE is the square root of MSE.

MAE: measures the average magnitude of the errors in a set of predictions, without considering their direction. It's the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

References

[1] https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/

[2] https://machinelearningmastery.com/how-to-configure-the-number-of-layers-and-nodes-in-a-neural-network/

[3] https://towardsdatascience.com/precision-vs-recall-386cf9f89488

[4] https://heartbeat.fritz.ai/evaluation-metrics-for-machine-learning-models-d42138496366

[5] https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f

[6] https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/

[7] http://neuralnetworksanddeeplearning.com/chap6.html

[8] https://www.cs.umd.edu/ tomg/projects/landscapes/

[9] https://ruder.io/optimizing-gradient-descent/