Strela Net Documentation

Timothy Liu

October 2018

Contents

1 Introduction

Strela Net is an implementation of a neural network that emulates some of the basic capabilities of a standard neural net library such as Pytorch. Strela currently supports binary classification, multi-class classification, L2 regularization, and several different error functions.

Strela was written over the course of three months in 2018. It is currently implemented in python and uses numpy; no other packages are used. The package was written to better understand how neural networks work and are implemented.

1.1 Repo contents

The github repository for strela can be found here:

github.com/timsliu/strela/tree/master/python

- /archive - archived past versions of strela.py. These versions are outdated and have subsets of the capabilities of strela.py at the top level.
- Figure_1.png - illustration of binary classification
- Figure_2.png - illustration of multiclass classification
- Implementing a Neural...pdf - online article about implementing a neural net referenced during development
- slides10.pdf - lecture 10 slide from Caltech course CS156a; referenced during development
- strela_notes.docx - development log
- strela.py - strela_net class and help function
- strela_helpers.py - helper functions for strela_net
- strela_test.py - test functions for strela

2 Design notes

Strela is written as a single class (strela_net) that can be initialized with multiple required and optional arguments. For a description of each method in the strela_net class, see the documentation in strela.py. This section briefly describes the high level design choices in strela.

The indexing and notation of strela is largely based on Lecture 10 of the Caltech CS156a course. The major difference is that only the input layer of strela has a bias term while the neural net described in the slides has a bias term at every layer.

2.1 Training and predicting

Strela trains using stochastic gradient descent and back propagation. The weight vectors and outputs of each layer are stored as 2D numpy arrays. Weights are initialized to random floats from -0.25 to 0.25. For consistency, data fed into strela is converted to a 2D numpy array, allowing it to take in both lists and numpy arrays.

Strela was designed to be expanded with new activation functions, error functions, and output encodings. There are separate functions for calculating $\frac{de}{dx}$ and $\frac{dx}{dx}$ where additional error and activation functions can be added. Output encodings convert the raw values generated into binary or multiclass classifications.

2.2 Error checking

Strela includes an error checking method that is called during initialization. The method checks that the arguments are of the correct data type and that the passed activation function, error function, and output encodings are supported. If an error is found, then a warning is printed and initialization is aborted.

3 Usage

3.1 Arguments

New instances of the strela_net class have four required arguments and eight additional optional arguments. Below is the class and initialization method declaration.

```
class strela_net():
    '''this class is a neural net that can be used for classifcation and
      prediction. The class does not inherit from a superclass'''
    def __init__(self, inputs, outputs, h_layers, h_layers_d, \
        lr = 1e-3, reg = 0, epochs = 25, \
        loss = "squared", act = "tanh", out_encode = "none", softmax = False,
        quiet = False):
```

The required arguments are:

1. inputs (int) - dimension of input space
2. outputs (int) - dimension of output space
3. h_layers (int) - number of hidden layers
4. h_layers_d (int) - number of nodes per hidden layer

The optional arguments are:

1. lr (flt) - learning rate
2. reg (flt) - regularization lambda
3. epochs (int) - number of passes through training data. Increased passes will slow down training time.
4. loss (str) - loss function
5. act (str) - activation function
6. out_encode (str) - output encoding. Output encodings change the raw output of the neural net into binary or one-hot encoded arrays that can be interpreted as labels.
7. softmax (bool) - turn on softmax. Softmax maps an output array so that all values are from 0 to 1 and sum to 1. When softmax is implemented, the outputs can be interpreted as the probability that the data belongs to a label.
8. quiet (bool) - suppress print statements. By default, strela prints out information when it is initialized and prints out progress statements during training.

3.2 Training

Strela accepts a range of list and list likes inputs for training. The class will convert the input type into a 2D numpy array regardless of the input. For binary classifications, the y values should be either +1 or -1 and the output encoding should be set to "binary". For multiclass training, each y value should be a one hot encoded array and the output encoding should be set to "one_hot".

Below is an example of creating a new instance of strela_net() for multiclass classification. Additional examples of usage can be found in the file strela_test.py

```
def test_multiclass(h_layers = 1, h_layers_d = 40, lr = 1e-3):
    n_input = 2         # dimensionality of the space
    n_output = 3        # number of classes

    # generate random x values
    x_train = 10 * (np.random.rand(train_size, n_input) - 0.5)
    x_test = 10 * (np.random.rand(test_size, n_input) - 0.5)

    # use helper function to create y_values
    y_train = tag_multiclass(x_train, n_output)
    y_test = tag_multiclass(x_test, n_output)

    # create instance of strela; use softmax and categorical cross entropy
    my_strela = strela_net(n_input, n_output, h_layers, h_layers_d, lr,\
     epochs = 25, softmax = True, loss = "squared", out_encode = "one_hot")

    my_strela.train(x_train, y_train)
    # predict the test set
    print("Generating predictions on test set...")
    y_pre = my_strela.predict(x_test)
```

3.3 Help

The file strela.py comes with a help function that describes the arguments for strela_net. Import strela (the file, not just the class) and call the help function for information:

```
>>> import strela

# I don't know what to do!
>>> strela.strela_help()

=== Strela net required and optional arguments: ===
Arg 1: inputs    (int) - dimension of input space
Arg 2: outputs   (int) - dimension of output space
Arg 3: h_layers  (int) - number of hidden layers
Arg 4: h_layers_d (int) - number of nodes per hidden layer
-----
Opt.:  lr        (flt) - learning rate              default: 0.001
Opt.:  reg       (flt) - regularization lambda      default: 0
Opt.:  epochs    (int) - passes through training data default: 25
-----
Opt.:  loss      (str) - loss function              default: squared
Opt.:  act       (str) - activation function        default: tanh
Opt.:  out_encode (str) - output encoding           default: none
Opt.:  softmax   (bool)- turn on softmax            default: False
Opt.:  quiet     (bool)- suppress print statements  default: True
-----
Supported loss functions:     ['squared', 'cce']
Supported activation functions: ['tanh']
Supported output encodings:   ['none', 'one_hot', 'binary']
-----
>>>
```

4 About the author

Timothy Liu graduated from Caltech with a B.S. in electrical engineering. He is currently an associate at Boston Consulting Group. For questions please contact timliu92@gmail.com.