as07

June 10, 2018

1 Team Members

1.1 RaviKiran Bhat

1.2 Rubanraj Ravichandran

1.3 Mohammad Wasil

1.4 Ramesh Kumar

```
In [1]: import tensorflow as tf
        from sklearn.linear_model import LinearRegression, LogisticRegression
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.metrics import log_loss
        import numpy as np
        import pickle
        from sklearn.externals import joblib
        import matplotlib.pyplot as plt
        %matplotlib inline
```

```
/home/ramesh/anaconda2/lib/python2.7/site-packages/h5py/__init__.py:34: FutureWarning: Conversio
  from ._conv import register_converters as _register_converters
```

```
In [2]: #Load MNIST data from tf
        from tensorflow.examples.tutorials.mnist import input_data
        mnist = input_data.read_data_sets("MNIST_data/", one_hot=False)
        mnist_one_hot = input_data.read_data_sets("MNIST_data/", one_hot=True)
```

```
WARNING:tensorflow:From <ipython-input-2-64bc5dd34d72>:3: read_data_sets (from tensorflow.contri
Instructions for updating:
Please use alternatives such as official/mnist/dataset.py from tensorflow/models.
WARNING:tensorflow:From /home/ramesh/anaconda2/lib/python2.7/site-packages/tensorflow/contrib/le
Instructions for updating:
Please write your own downloading logic.
WARNING:tensorflow:From /home/ramesh/anaconda2/lib/python2.7/site-packages/tensorflow/contrib/le
Instructions for updating:
Please use tf.data to implement this functionality.
Extracting MNIST_data/train-images-idx3-ubyte.gz
```

```
WARNING:tensorflow:From /home/ramesh/anaconda2/lib/python2.7/site-packages/tensorflow/contrib/le
Instructions for updating:
Please use tf.data to implement this functionality.
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
WARNING:tensorflow:From /home/ramesh/anaconda2/lib/python2.7/site-packages/tensorflow/contrib/le
Instructions for updating:
Please use alternatives such as official/mnist/dataset.py from tensorflow/models.
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
WARNING:tensorflow:From /home/ramesh/anaconda2/lib/python2.7/site-packages/tensorflow/contrib/le
Instructions for updating:
Please use tf.one_hot on tensors.
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
```

```
In [3]: def plot_img(image, index):
            f, ax = plt.subplots(1, len(index))
            for i in range(len(index)):
                ax[i].imshow(np.reshape(image[index[i]], (28,28)), cmap='Greys')
                ax[i].set_yticklabels([])
                ax[i].set_xticklabels([])
            plt.show()
```

```
In [24]: class mnistTwoClassifiers(object):
             def __init__(self):
                 self.step = 100

             def update_datasets(self, train_images, train_labels, test_images, test_labels):
                 self.train_images = train_images
                 self.train_labels = train_labels
                 self.test_images = test_images
                 self.test_labels = test_labels

             def logistic_regression(self, train_size):
                 self.logistic_model = LogisticRegression()
                 print self.train_images[:train_size].shape
                 print self.train_labels[:train_size].shape
                 self.logistic_model.fit(self.train_images[:train_size], self.train_labels[:trai
                 return self.logistic_model

             def random_forest(self, train_size):
                 self.random_forest_model = RandomForestClassifier()
                 %time self.random_forest_model.fit(self.train_images[:train_size], self.train_l
                 return self.random_forest_model
```

```
    def generate_adversarial_example(self, num_of_samples, clf, epsilon, test_images, t
        adv_samples = np.zeros((num_of_samples, 784))
        adv_labels = np.zeros((num_of_samples, 10))
        adv_labels_true = np.zeros((num_of_samples, 1))
        for i in range(num_of_samples):
            y_predict = None
            if clf == "logistic_regression":
                y_predict = self.logistic_model.predict_proba(test_images[i:i+1])
            elif clf == "random_forest":
                y_predict = self.random_forest_model.predict_proba(test_images[i:i+1])

            y_true = test_labels[i:i+1]
            y_true_index = np.where(y_true == 1)[1][0]
            predictions = y_predict
            error = (predictions - y_true)**(2)
            error = error[0][y_true_index]
            gradient = error * train_images[i:i+1]
            gradient /= len(train_images[i:i+1])
            signs = np.sign(gradient)
            img_adversarial = epsilon * signs * test_images[i:i+1]
            adv_samples[i] = img_adversarial
            adv_labels[i] = y_true
            adv_labels_true[i] = y_true_index
        adv_samples = np.asarray(adv_samples)
        return adv_samples, adv_labels,adv_labels_true

    def test(self, test_size, classifier):
        if classifier == "logistic_regression":
            print(self.logistic_model.score(self.test_images[:test_size], self.test_lab
        elif classifier == "random_forest":
            print(self.random_forest_model.score(self.test_images[:test_size], self.tes

    def predict(self, model, test_images, test_size):
        return model.predict(test_images[:test_size])

    def test_with_adversarial(self, model, adv_example):
        return model.predict(adv_example)
```

```
In [25]: train_images = mnist.train.images
         train_labels = mnist.train.labels
         test_images = mnist.test.images
         test_labels = mnist.test.labels
         test_labels_one_hot = mnist_one_hot.test.labels
         test_images_one_hot = mnist_one_hot.test.images
         train_labels_one_hot = mnist_one_hot.train.labels
         train_images_one_hot = mnist_one_hot.train.images
         two_clfs = mnistTwoClassifiers()
```

```
        two_clfs.update_datasets(train_images, train_labels, test_images, test_labels)
```

2 Task 1 -5

```
In [26]: #train the models
         logistic_model = two_clfs.logistic_regression(55000)
         random_forest_model = two_clfs.random_forest(55000)
         clf_logistic = two_clfs.logistic_model
         clf_random_forest = two_clfs.random_forest_model
         #Save classifier to pickle file
         joblib.dump(clf_logistic, 'logistic_regression.pkl')
         joblib.dump(clf_random_forest, 'random_forest_model.pkl')
```

```
(55000, 784)
(55000,)
CPU times: user 7.95 s, sys: 4.04 ms, total: 7.95 s
Wall time: 8.14 s
```

```
Out[26]: ['random_forest_model.pkl']
```

```
In [7]: #Load classifier
        # logistic_clf = joblib.load('/tmp/logistic_regression.pkl')
        # random_forest_clf = joblib.load('/tmp/random_forest_model.pkl')
```

3 Test trained Model

```
In [27]: #test the model

         two_clfs.test(test_images.shape[0], "logistic_regression")
         two_clfs.test(test_images.shape[0], "random_forest")
```

```
0.9198
0.9476
```

4 Use Test dataset to generate Adversarial Examples

```
In [28]: #Use test dataset
         num_adv_example = 100
         epsilon = 0.007
         adv_imgs_logistic, labels_logistic, adv_labels_true_logistic = two_clfs.generate_advers
                                                           epsilon, test_images_one_hot, test_lab
         adv_imgs_rnd_forest, labels_rnd_forest, adv_labels_true_rnd_forest = two_clfs.generate_
                                                           epsilon, test_images_one_hot, test_lab
```

```
In [29]: #evaluate trained model using random forest on  adversarial examples
         clf_random_forest.score(adv_imgs_logistic, test_labels[:num_adv_example])
```

```
Out[29]: 0.27
```

```
In [30]: #evaluate trained model using logistics on  adversarial examples
         clf_logistic.score(adv_imgs_rnd_forest, train_labels[:num_adv_example])
```

```
Out[30]: 0.08
```

5 Task 6

6 Generate 55000 adversarial examples

```
In [42]: #Use original training dataset and generate adversarial examples
         num_adv_example_task_six = 55000
         epsilon_task_six = 0.007
         adv_imgs_logistic_task_six, logic_labels_task_six, logic_labels_true_task_six = two_clf
                                                          epsilon, train_images_one_hot, train_l
         adv_imgs_rnd_forest_task_six, forest_labels_task_six, forest_labels_true_task_six = two
                                                          epsilon, train_images_one_hot, train_l
```

```
In [43]: new_training_set = np.concatenate((train_images,adv_imgs_logistic_task_six))
         new_training_set_label = np.concatenate((train_labels,logic_labels_true_task_six.T[0]))
         new_training_set_label_hot = np.concatenate((train_labels_one_hot,logic_labels_task_six
```

```
In [44]: adv_classifiers = mnistTwoClassifiers()
         adv_classifiers.update_datasets(new_training_set, new_training_set_label, test_images,
```

```
In [45]: #train the models
         logistic_model = adv_classifiers.logistic_regression(new_training_set.shape[0])
         random_forest_model = adv_classifiers.random_forest(new_training_set.shape[0])
         clf_logistic = adv_classifiers.logistic_model
         clf_random_forest = adv_classifiers.random_forest_model
```

```
(65000, 784)
(65000,)
CPU times: user 10 s, sys: 108 ms, total: 10.1 s
Wall time: 10.5 s
```

7 Evaluate model trained on Adversarial examples and training dataset

```
In [49]: print "Classification accuracy on adversarial dataset using logistic regression"
         adv_classifiers.test(test_images.shape[0], "logistic_regression")
```

```
Classification accuracy on adversarial dataset using logistic regression
0.9199
```

```
In [50]: print "Classification accuracy on adversarial dataset using random forest"
         adv_classifiers.test(test_images.shape[0], "random_forest")
```

```
Classification accuracy on adversarial dataset using random forest
0.9465
```

8 Does classification performance improve?

Yes, classification performance of models trained on combination of adversarial examples and original dataset improves significantly

9 Is the new model less or more susceptible to adversarial examples?

New model is more robust with adversarial examples because it is trained on adversarial examples also.

10 Do you think you can use a regularization method in order to make the model less susceptible to adversarial examples?

No, state of the art shows that Generic regularization strategies such as dropout, pretraining, and model averaging do not confer a significant reduction in a model's vulnerability to adversarial examples, but changing to nonlinear model families such as RBF networks can do so.

```
In [51]: # np.save("adv_imgs_logistic", adv_imgs_logistic_task_six)
         # np.save("logic_labels", logic_labels_task_six)
         # np.save("adv_imgs_rnd_forest", adv_imgs_rnd_forest_task_six)
         # np.save("forest_labels", forest_labels_task_six)
```