Adaboost, MLP and CNN applied to W/Z classification

朱永峰

2017 年 12 月 18 日

the result of Bingyang(using Xgboost)

```
for file_type in ["PN_MN_FN","PN_MN_FY","PN_MY_FN","PN_MY_FY"]:
    data_train,data_test=get_data(file_type)
    x_train=data_train[:,1:]
    y_train=data_train[:,0]
    x_test =data_test[:,1:]
    y_test =data_test[:,0]
    print len(y_train)
    xgb = XGBClassifier(
            learning_rate=0.02,
            n_estimators=1000,
            max_depth=9,
            min_child_weight=1,
            gamma=0.05,
            subsample=0.9,
            scale_pos_weight=1.0
            )
    xgb.fit(x_train,y_train)
    model_path='./model/'+file_type+'.pickle.dat'
    pickle.dump(xgb,open(model_path,'wb'))
    xgb=pickle.load(open(model_path,'rb'))
    label_path='./label/'+file_type+'.npy'
    predict=xgb.predict(x_test)
    np.save(label_path,predict)
    print("Type:"+file_type)
    print("Accuracy:"+str(accuracy_score(y_test,predict)))
    E_ww,E_zz=accuracy(y_test,predict)
    print("E_ww:"+str(E_ww))
    print("E_zz:"+str(E_zz))
```

```
76958
Type:PN_MN_FN
Accuracy:0.653863146132
E_ww:0.624380421314
E_zz:0.683742347339
76958
Type:PN_MN_FY
Accuracy:0.707346864524
E_ww:0.736194740363
E_zz:0.677918941621
76958
Type:PN_MY_FN
Accuracy:0.849606278749
E_ww:0.815485142353
E_zz:0.883882058762
76958
Type:PN_MY_FY
Accuracy:0.865277164167
E_ww:0.839453811938
E_zz:0.891355445286
```

my result:using Adaptive boost, 10 weak learners, the accuracy is:

```
0.576314353284
W accuracy is:0.537318071166
Z accuracy is:0.615585228458
0.624730372411
W accuracy is:0.63998973306
Z accuracy is:0.609084688668
0.580472465501
W accuracy is:0.584266749637
Z accuracy is:0.576660588695
0.627173263338
W accuracy is:0.652896412486
Z accuracy is:0.601242041541
```

for PN-MN-FN, using 100 weak learners ,the accuracy is:

```
(76958, 17)
accuracy:0.59349255438
W accuracy is:0.599718074554
Z accuracy is:0.587322121604
```

for PN-MN-FY, using 100 weak learner, the accuracy is:

```
(76958, 617)
accuracy:0.646534473349
W accuracy is:0.656991132192
Z accuracy is:0.63590630404
```

We can see the accuracy is improving with the increasing of the number of weak learners. I want to use 1000 weak learners, but the computer.....

Adaptive boost

1. In daily life, if you were gong to make an important decision, you'd probably get the advice of multiple experts instead of trusting one person.

Adaptive boost

1. In daily life, if you were gong to make an important decision, you'd probably get the advice of multiple experts instead of trusting one person.
2. For machine learning, one idea that natually arises is combining multiple classifiers. Methods that do this are known as ensemble methods, such as bagging and boosting. Today, we focus on boosting.

Adaptive boost

1. In daily life, if you were gong to make an important decision, you'd probably get the advice of multiple experts instead of trusting one person.
2. For machine learning, one idea that natually arises is combining multiple classifiers. Methods that do this are known as ensemble methods, such as bagging and boosting. Today, we focus on boosting.
3. The boosting using the same type of classifier.The different classifiers are trained sequentially.Each new classifier is trained based on the performance of those already trained.Boosting makes new classifiers focus on data that was previously misclassified by previous classifiers.The output is calculated from a weighted sum of all classifiers.The weights are based on how successful the classifier was in the previous iteration.The Adaboost is one of the boosting.

Adaptive boost

1. In daily life, if you were gong to make an important decision, you'd probably get the advice of multiple experts instead of trusting one person.
2. For machine learning, one idea that natually arises is combining multiple classifiers. Methods that do this are known as ensemble methods, such as bagging and boosting. Today, we focus on boosting.
3. The boosting using the same type of classifier.The different classifiers are trained sequentially.Each new classifier is trained based on the performance of those already trained.Boosting makes new classifiers focus on data that was previously misclassified by previous classifiers.The output is calculated from a weighted sum of all classifiers.The weights are based on how successful the classifier was in the previous iteration.The Adaboost is one of the boosting.
4. Adaboost works this way: A weight is applied to every example in the training data.Initially, these weights are all equal. A weak classifier(or weak learner, for Adaboost is decision stump:A decision stump is a simple decision tree which makes a decision on one feature only.) is first trained on the training data.The errors from the weak classifier are calculated, and the weak classifier is trained a second time with the same dataset.This second time the weak classifier is trained, the weights of the training set are adjusted, the examples properly classified in the first time are weighted less and incorrectly classified in the first iteration are weighted more.

MLP

MLP, four hidden layers, 12 neurons each layer, 10000 training samples, 500 testing samples, the droupout is 0.9, the final result is as follows:

```
    hidden1 = tf.nn.relu(tf.matmul(x, W1) + b1)
    hidden1_drop = tf.nn.dropout(hidden1, keep_prob)
    hidden2 = tf.nn.relu(tf.matmul(hidden1_drop, W2) + b2)
    hidden2_drop = tf.nn.dropout(hidden2, keep_prob)
    hidden3 = tf.nn.relu(tf.matmul(hidden2_drop, W3) + b3)
    hidden3_drop = tf.nn.dropout(hidden3, keep_prob)
    hidden4 = tf.nn.relu(tf.matmul(hidden3_drop, W4) + b4)
    hidden4_drop = tf.nn.dropout(hidden4, keep_prob)
    y = tf.nn.softmax(tf.matmul(hidden4_drop, W5) + b5)

# Define loss and optimizer
    y_ = tf.placeholder(tf.float32, [None, 2])
    cross_entropy = tf.reduce_mean(-tf.reduce_sum(y_ * tf.log(y), reduction_indices=[1]))
    train_step = tf.train.GradientDescentOptimizer(0.0001).minimize(cross_entropy)
```

```
step 0,test_accuracy 0.476
step 100,test_accuracy 0.524
step 200,test_accuracy 0.524
step 300,test_accuracy 0.524
step 400,test_accuracy 0.524
step 500,test_accuracy 0.524
step 600,test_accuracy 0.524
step 700,test_accuracy 0.524
step 800,test_accuracy 0.524
step 900,test_accuracy 0.524
step 1000,test_accuracy 0.524
step 1100,test_accuracy 0.524
step 1200,test_accuracy 0.524
step 1300,test_accuracy 0.524
```

Bingyang told me that my model is so simple, next I will change the model of MLP.

- A multilayer perceptron consists of at least three layers of nodes.
- Except for the input nodes, each node is a neuron that uses a nonlinear activation function.
- MLP uses a supervised learning technique called backpropogation for training.
- MLP can distinguish data that is not linearly separable(Hidden layers will use a space convertion to make data linearly distinguished.)

CNN

```
W_conv1 = weight_variable([1, 3, 1, 32])
b_conv1 = bias_variable([32])
h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)
#h_pool1 = max_pool_2x2(h_conv1)

W_conv2 = weight_variable([1, 3, 32, 64])
b_conv2 = bias_variable([64])
h_conv2 = tf.nn.relu(conv2d(h_conv1, W_conv2) + b_conv2)
#h_pool2 = max_pool_2x2(h_conv2)

W_fc1 = weight_variable([17 * 64, 300])
b_fc1 = bias_variable([300])
h_conv2_flat = tf.reshape(h_conv2, [-1, 17*64])
h_fc1 = tf.nn.relu(tf.matmul(h_conv2_flat, W_fc1) + b_fc1)
```

```
step 0, training accuracy 0.48
step 100, training accuracy 0.36
step 200, training accuracy 0.46
step 300, training accuracy 0.54
step 400, training accuracy 0.4
step 500, training accuracy 0.4
step 600, training accuracy 0.56
step 700, training accuracy 0.44
step 800, training accuracy 0.42
step 900, training accuracy 0.44
step 1000, training accuracy 0.44
step 1100, training accuracy 0.58
step 1200, training accuracy 0.46
step 1300, training accuracy 0.4
step 1400, training accuracy 0.56
step 1500, training accuracy 0.48
```