金融作业汇报稿

量化模型

1) 通过历史股票交易数据预估未来的股票交易数据，从而实现盈利

2) XGBOOST

https://xgboost.ai/

https://xgboost.readthedocs.io/en/latest/

3) LSTM

https://keras.io/layers/recurrent/#lstm

参考：

数据获取

从网易爬取每日上海证券交易所所有股票交易数据

（ 代码见spider部分 ）

模型训练

1) XGBOOST

训练代码

```
def train(self, X_train_scaled, y_train_scaled):
    '''
    Train model
    Inputs
        X_train_scaled     : features for training. Scaled to have mean 0 and variance 1
        y_train_scaled     : target for training. Scaled to have mean 0 and variance 1
    '''
    model = XGBRegressor(seed=self.model_seed,
                         n_estimators=self.n_estimators,
                         max_depth=self.max_depth,
                         learning_rate=self.learning_rate,
                         min_child_weight=self.min_child_weight,
                         subsample=self.subsample,
                         colsample_bytree=self.colsample_bytree,
                         colsample_bylevel=self.colsample_bylevel,
                         gamma=self.gamma)

    # Train the regressor
    model.fit(X_train_scaled, y_train_scaled)

    self._save_model(model)
```

2) LSTM

训练代码

```
def train(self, x_train_scaled, y_train_scaled):
    '''
    Train model
    Inputs
        x_train_scaled  : e.g. x_train_scaled.shape=(451, 9, 1). Here we are using the past 9 values to
predict the next value
        y_train_scaled  : e.g. y_train_scaled.shape=(451, 1)
    '''
    # Create the LSTM network
    model = Sequential()
    model.add(LSTM(units=self.lstm_units, return_sequences=True, input_shape=(x_train_scaled.shape[1], 1)))
    model.add(Dropout(self.dropout_prob))  # Add dropout with a probability of 0.5
    model.add(LSTM(units=self.lstm_units))
    model.add(Dropout(self.dropout_prob))  # Add dropout with a probability of 0.5
    model.add(Dense(1))

    # Compile and fit the LSTM network
    model.compile(loss='mean_squared_error', optimizer=self.optimizer)
    model.fit(x_train_scaled, y_train_scaled, epochs=self.epochs, batch_size=self.batch_size, verbose=0)

    # Print model summary
    model.summary()

    # Save model
    self._save_model(model)
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 3, 128)            66560
_________________________________________________________________
dropout_1 (Dropout)          (None, 3, 128)            0
_________________________________________________________________
lstm_2 (LSTM)                (None, 128)               131584
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 129
=================================================================
```

模型预测

1) XGBOOST

预测代码

```
def predict(self, X_test_scaled, y_test, col_mean, col_std):
    '''
    predict
    Inputs
        X_test_scaled      : features for test. Each sample is scaled to mean 0 and variance 1
        y_test             : target for test. Actual values, not scaled.
        col_mean           : means used to scale each sample of X_test_scaled. Same length as
X_test_scaled and y_test
        col_std            : standard deviations used to scale each sample of X_test_scaled. Same length
as X_test_scaled and y_test
    Outputs
        rmse               : root mean square error of y_test and est
        mape               : mean absolute percentage error of y_test and est
        est                : predicted values. Same length as y_test
    '''
    model = self._load_model()

    # Get predicted labels and scale back to original range
    est_scaled = model.predict(X_test_scaled)
    est = est_scaled * col_std + col_mean

    # Calculate RMSE
    rmse = utils.get_rmse(y_test, est)

    # Calculate MAPE
    mape = utils.get_mape(y_test, est)

    return rmse, mape, est
```

预测结果：

RMSE on test set = 1.170
MAPE on test set = 0.583%

2) LSTM

预测代码

```
def predict(self, \
          x_cv_scaled, \
          y_cv, \
          mu_cv_list, \
          std_cv_list):
    '''
    Train model, do prediction, scale back to original range and do evaluation
    Use LSTM here.
    Returns rmse, mape and predicted values
    Inputs
        x_cv_scaled     : use this to do predictions
        y_cv            : actual value of the predictions
        mu_cv_list      : list of the means. Same length as x_scaled and y
        std_cv_list     : list of the std devs. Same length as x_scaled and y
    Outputs
        rmse            : root mean square error
        mape            : mean absolute percentage error
        est             : predictions
    '''
    model = self._load_model()

    # Do prediction
    est_scaled = model.predict(x_cv_scaled)
    est = (est_scaled * np.array(std_cv_list).reshape(-1, 1)) + np.array(mu_cv_list).reshape(-1, 1)

    # Calculate RMSE and MAPE
    rmse = utils.get_rmse(y_cv, est)
    mape = utils.get_mape(y_cv, est)

    return rmse, mape, est
```

预测结果：

RMSE on test set = 1.169
MAPE on test set = 0.586%

盈利分析

待补充

附录

代码 >> （github地址）