Tensorflow+ H2o +Spark New super tool for DS

Jiqiong QIU

About me

- Master of engineer degree in Bioinformatic and modelization at INSA de Lyon
- PhD in Color Formulation by Statistical Learning at UTC and BASF Coatings
  (EP 2887275A1: Method and system for determining a color formula)
- Data scientist at Sfeir

TensorFlow + H2o + Spark

1. What is H2o?
2. What is Spark?
3. What is TensorFlow?
4. TensorFlow + H2o + Spark

What is H2O

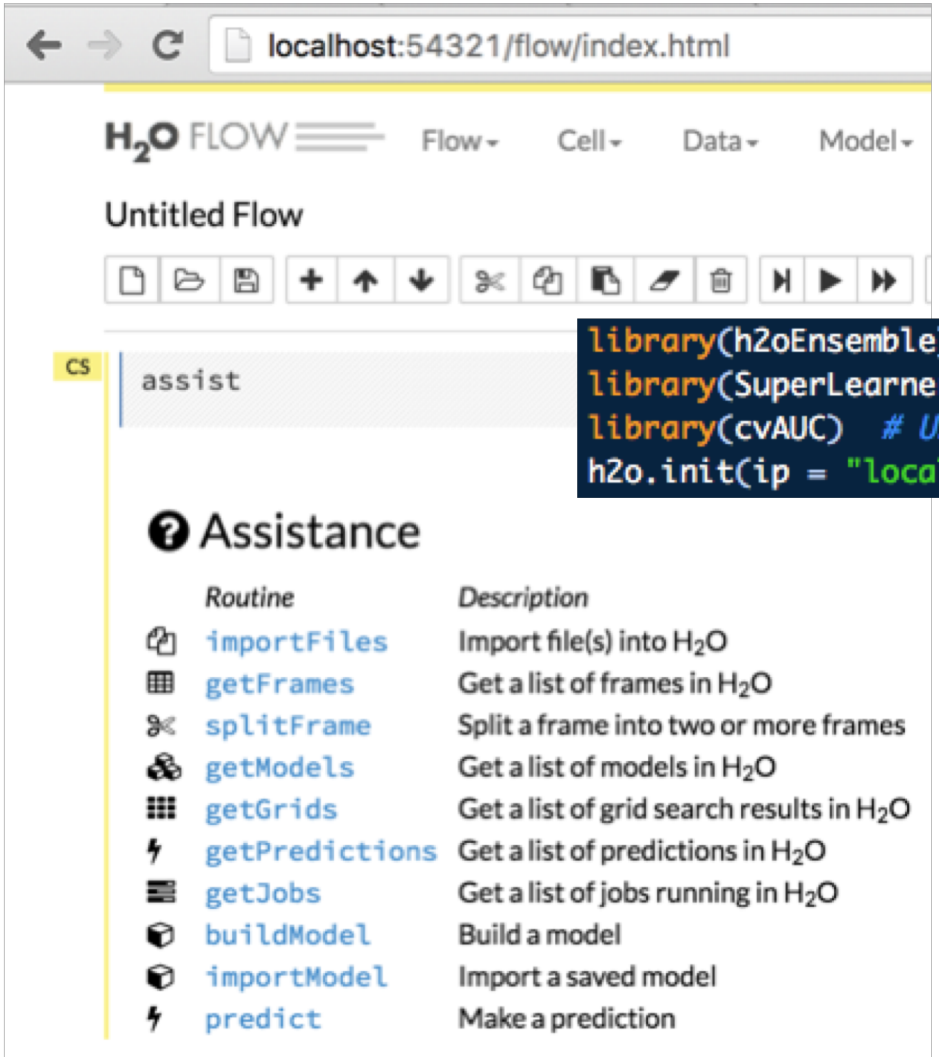
http://www.h2o.ai/

Open-source software for big-data Analysis

What is H2O

1. Open Source
2. Easy-to-use WebUI and Familiar Interfaces:
    1. Web-based user interface
    2. R, Java, Scala, Python, JSON, APIs by H2o
3. Data Agnostic Support for all Common Database and File Types: HDFS, S3, SQL and NoSQL data sources
4. Massively Scalable Big Data Analysis
5. Real-time Data Scoring

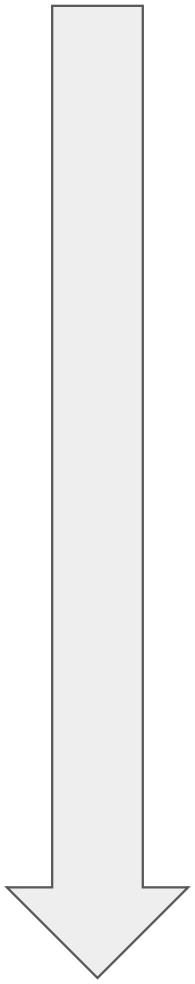
What is H2o




```
library(h2oEnsemble)
library(SuperLearner)  # For metalearner such as "SL.glm"
library(cvAUC)  # Used to calculate test set AUC (requires version >=1.0.1 of cvAUC)
h2o.init(ip = "localhost", port = 54321, startH2O = TRUE, nthreads = -1)
```

Web Interface available at:
Localhost:54321/

Easy Ensemble learning with H2o



Import Data to H2o

```
test.hex <- as.h2o(valideData)
train.hex <- as.h2o(trainData)
y <- names(train)[ncol(train)]
x <- names(train)[-ncol(train)]
family <- "binomial"
```

Create ML model

```
h2o.randomForest.1 <- function(..., ntrees = 1000, nbins = 100, seed =
h2o.deeplearning.1 <- function(..., hidden = c(400,200), hidden_dropou
h2o.gbm.1 <- function(...,n.trees = 350,n.minobsinnode = 1,shrinkage =
learner <- c("h2o.randomForest.1", "h2o.deeplearning.1", "h2o.gbm.1")
metalearner <- c("SL.glm")
```

Create final model

```
# Train the ensemble using 4-fold CV to generate level-one data
# More CV folds will take longer to train, but should increase performance
fit <- h2o.ensemble(x = x, y = y, training_frame = train.hex, family = family,
                    learner = learner, metalearner = metalearner,
                    cvControl = list(V=4))
# Generate predictions on the test set
pred <- predict(fit, train.hex)
```

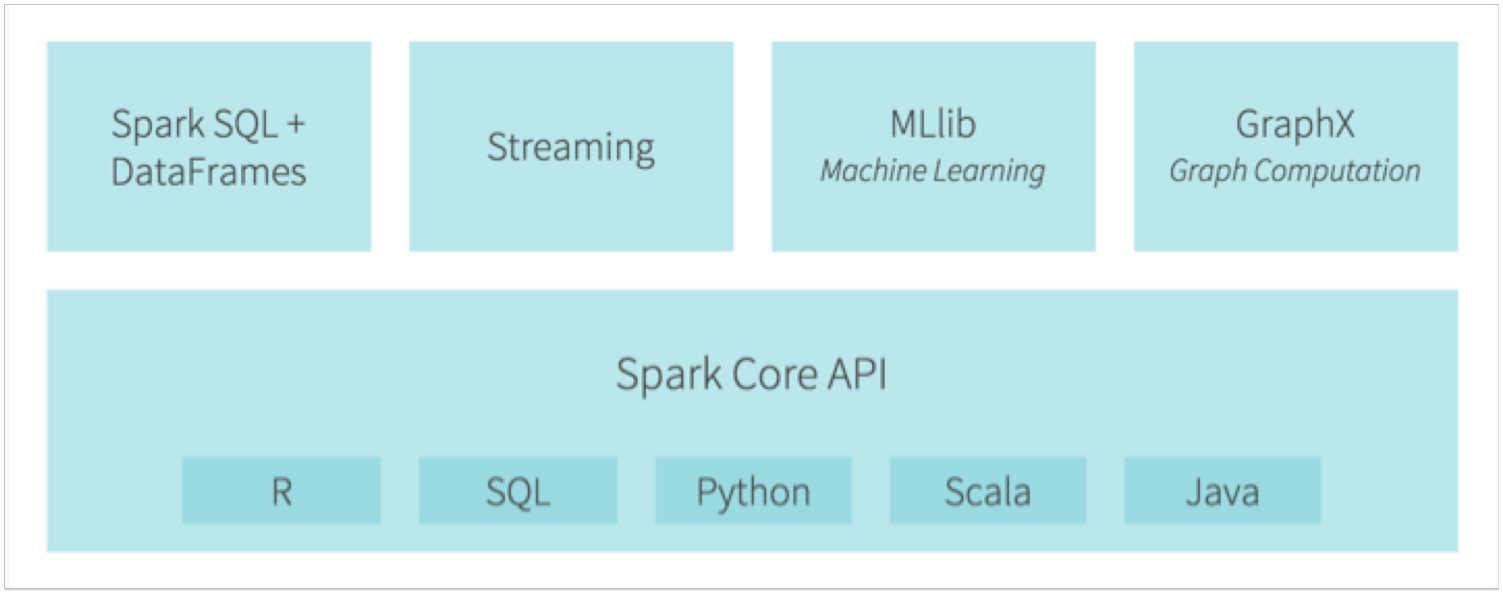
What is Spark

Speed

Ease of use

Generality

Runs Everywhere



What is Spark

Machine Learning in Spark

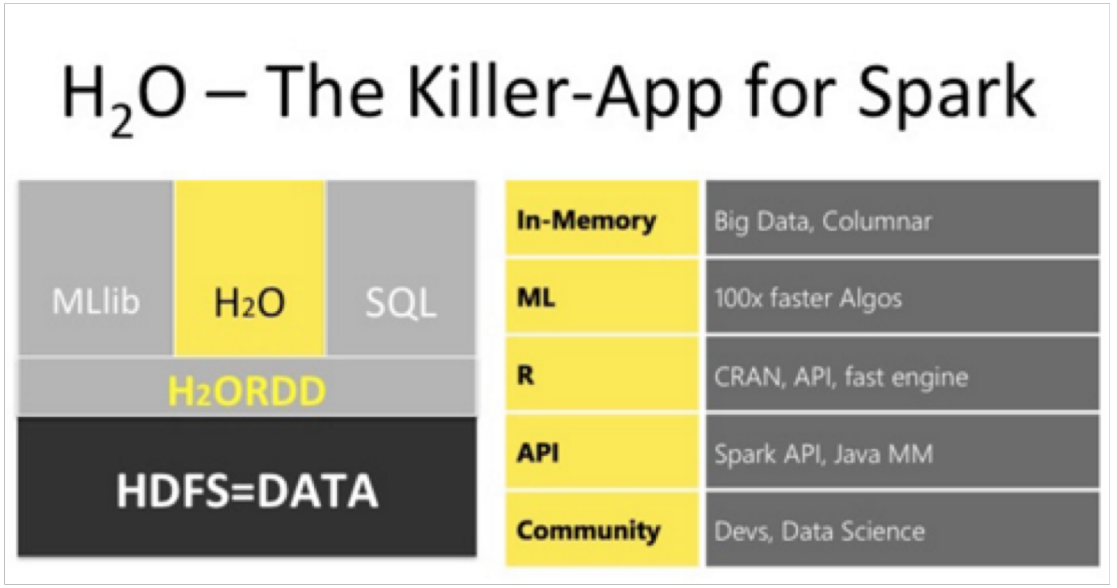
```
from pyspark.mllib.classification import LogisticRegressionWithLBFGS, LogisticRegressionModel
from pyspark.mllib.regression import LabeledPoint

# Load and parse the data
def parsePoint(line):
    values = [float(x) for x in line.split(' ')]
    return LabeledPoint(values[0], values[1:])

data = sc.textFile("data/mllib/sample_svm_data.txt")
parsedData = data.map(parsePoint)

# Build the model
model = LogisticRegressionWithLBFGS.train(parsedData)
```

H2o + Spark

Sparkling Water

Spark + H_2O

SPARKLING WATER



TensorFlow + H2o + Spark

Now it is possible!!!

https://github.com/h2oai/sparkling-water/blob/master/py/examples/notebooks/TensorFlowDeepLearning.ipynb

H2O TensorFlow Deep Learning Demo

Prerequisites

1. Download Sparkling Water from http://www.h2o.ai/download/sparkling-water/spark16
2. Follow instructions to setup PySparkling (especially steps 1 and 2)
3. Launch a Jupyter Notebook that connects to PySparkling:

```
cd ~/Downloads
unzip sparkling-water-1.6.3.zip
cd sparkling-water-1.6.3
~/sparkling-water-1.6.3$ IPYTHON_OPTS="notebook" bin/pysparkling
```

4. Point the Notebook to this file (e.g., download it first, then upload into the Notebook)

Introduction

In this tutorial, we'll build a simple 2-layer deep artificial neural network to classify handwritten digits MNIST. If you are not familiar with these terms, please check out our Deep Learning Booklet.

TensorFlow + H2o + Spark

1. Installation of **Sparkling Water:** http://www.h2o.ai/download/sparkling-water/spark16
2. Not enough! Build Error
    1. In fact: for python you need install: h2o, grip, tabulate, wheel, scikit-learn
    2. Please tests if you have: scikit-learn, numpy, scipy, pandas, statsmodels, patsy,future

```
## Read MNIST data into H2O
import h2o
from pysparkling import H2OContext
print h2o.__version__
hc = H2OContext(sc).start()
print(hc)
DATASET_DIR="http://s3.amazonaws.com/h2o-public-test-data/bigdata/laptop/mnist"
train_frame = h2o.import_file("{}/{}".format(DATASET_DIR, "train.csv.gz"))
test_frame = h2o.import_file("{}/{}".format(DATASET_DIR, "test.csv.gz"))
```

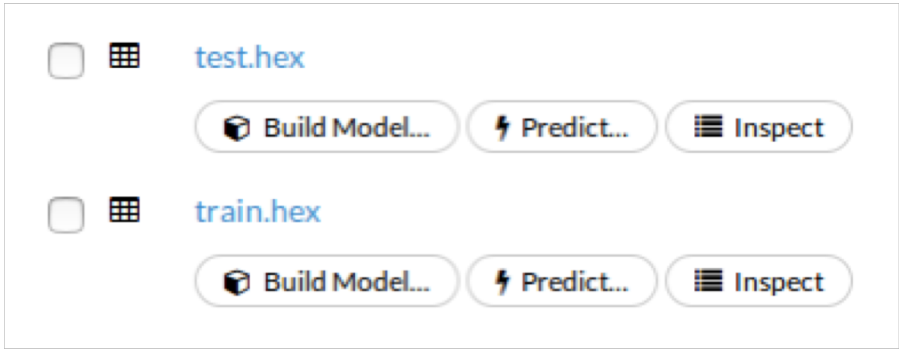
https://groups.google.com/forum/#!topic/h2ostream/QU-jaqH4vF4

TensorFlow + H2o + Spark

H2O cluster uptime:	4 seconds 929 milliseconds
H2O cluster version:	3.8.2.3
H2O cluster name:	sparkling-water-sanofi_1687948464
H2O cluster total nodes:	3
H2O cluster total free memory:	2.88 GB
H2O cluster total cores:	24
H2O cluster allowed cores:	24
H2O cluster healthy:	True
H2O Connection ip:	192.168.1.28
H2O Connection port:	54327
H2O Connection proxy:	None
Python Version:	2.7.11+

H2o Web available at:
http://192.168.1.28:54327/flow/index.html

getFrames



TensorFlow + H2o + Spark

```
import tensorflow
## Initialize TensorFlow session and test it
def map_fun(i):
  import tensorflow as tf
  with tf.Graph().as_default() as g:
    hello = tf.constant('Sparkling, TensorFlow!', name="hello_constant")
    with tf.Session() as sess:
      return sess.run(hello)
sc.parallelize(range(NODES), NODES).map(map_fun).collect()
```

```
['Sparkling, TensorFlow!', 'Sparkling, TensorFlow!']
```

TensorFlow + H2o + Spark

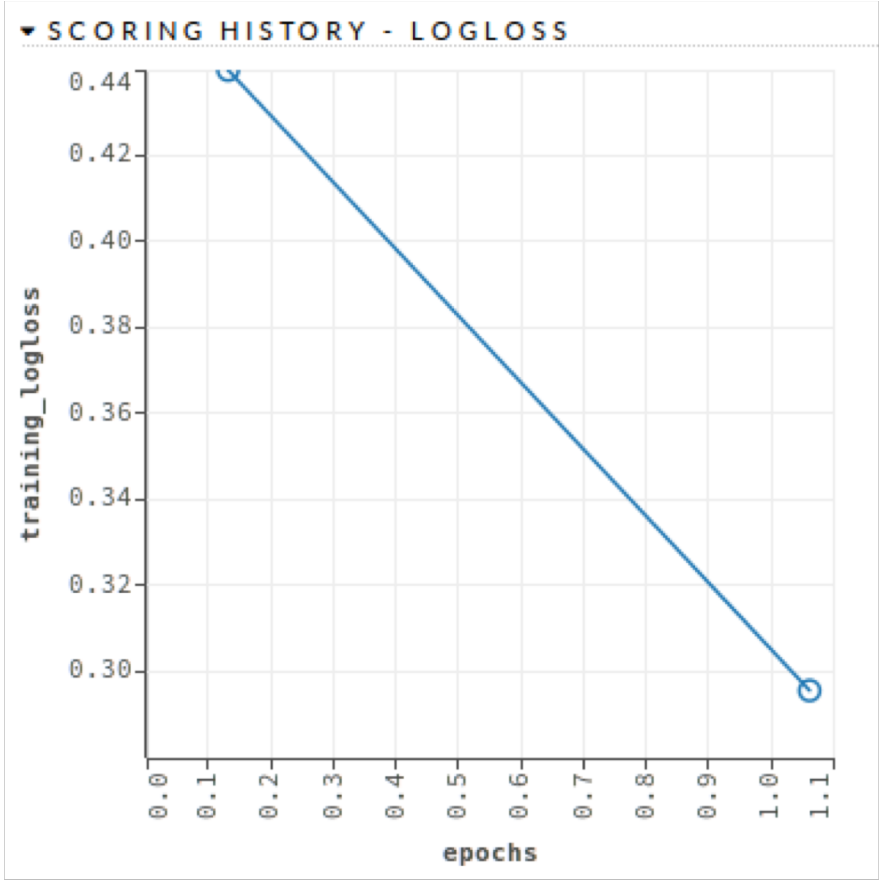
```
train_df = hc.as_spark_frame(train_frame).repartition(NODES)
test_df = hc.as_spark_frame(test_frame).repartition(NODES)
#train_df.printSchema()
```

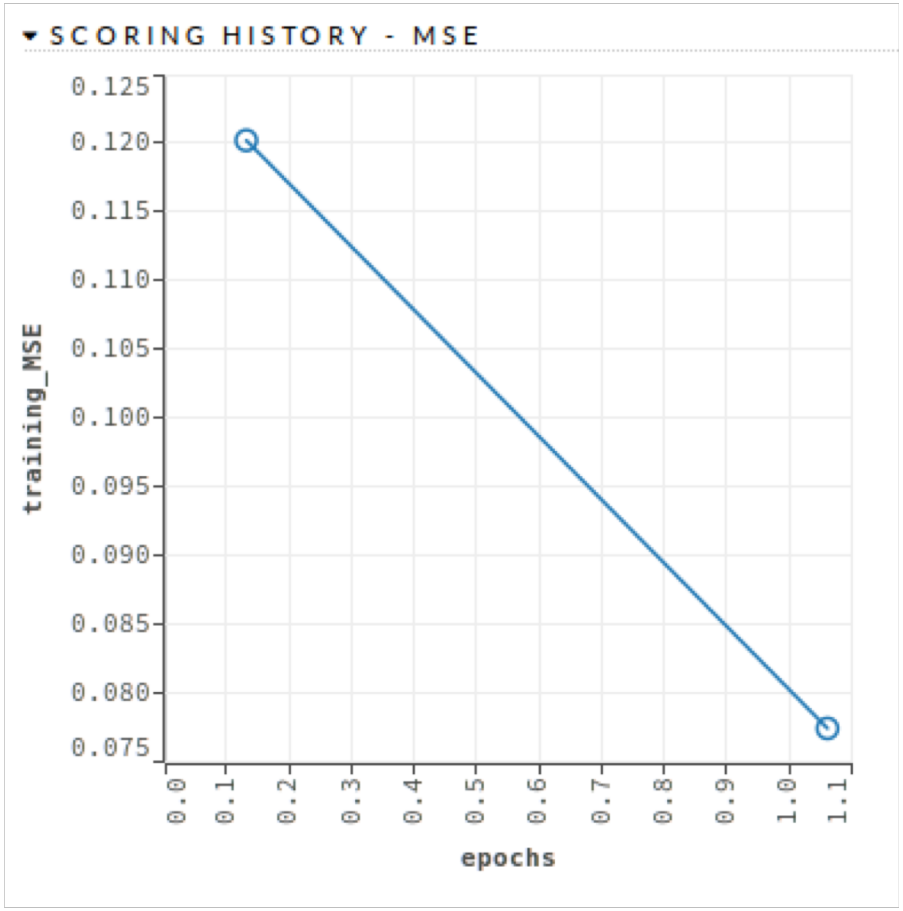
Expose the data in H2O through the Spark DataFrame API

Run TensorFlow on each H2O/PySparkling node

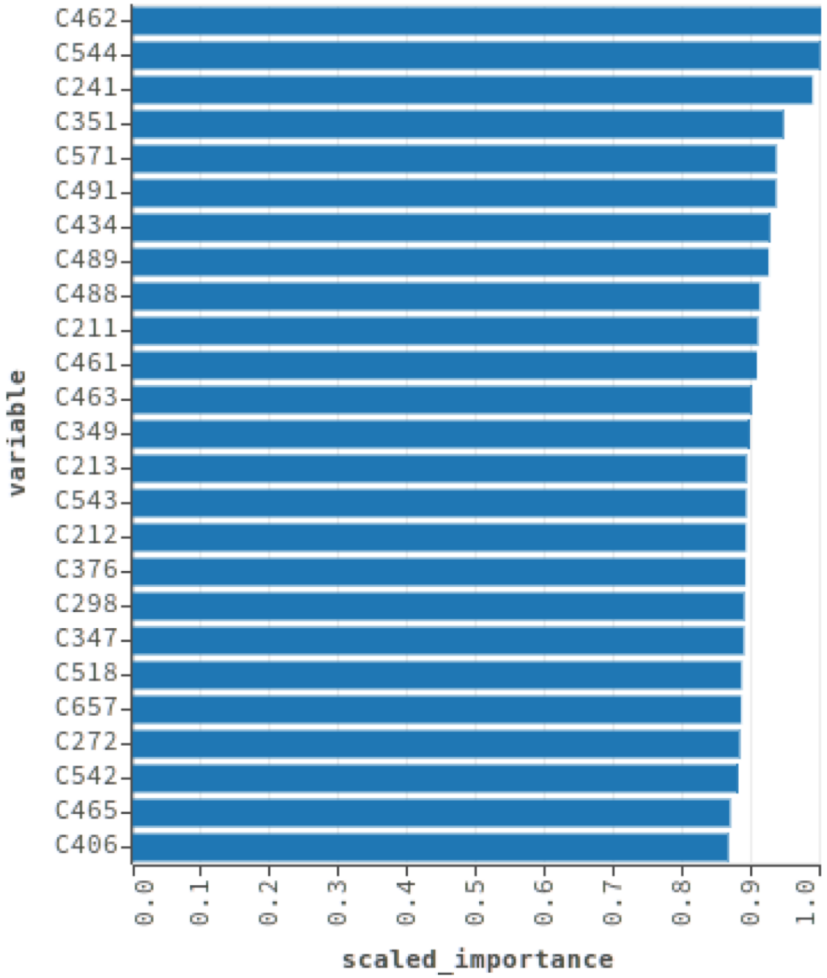
```
def create_nn(data_train, data_test, iterations, batch_size):
    ## input
    x = tf.placeholder(tf.float32, [None, 784])
    ## weights
    W = [tf.Variable(tf.random_normal([784,HN],stddev=0.1))
        ,tf.Variable(tf.random_normal([HN, HN],stddev=0.1))
        ,tf.Variable(tf.random_normal([HN, 10],stddev=0.1))]
    ## biases
    b = [tf.Variable(tf.random_normal([HN],    stddev=0.1))
        ,tf.Variable(tf.random_normal([HN],    stddev=0.1))
        ,tf.Variable(tf.random_normal([10],    stddev=0.1))]
    ## hidden layer activation
    h1 = tf.nn.relu(   tf.matmul(x,  W[0]) + b[0])
    h2 = tf.nn.relu(   tf.matmul(h1, W[1]) + b[1])
    ## output
    y = tf.nn.softmax( tf.matmul(h2, W[2]) + b[2])
    ## storage for actual labels
    y_ = tf.placeholder(tf.float32, [None, 10])
    ## cost function
    cross_entropy = -tf.reduce_sum(y_*tf.log(y))
    ## optimizer
    train_step = tf.train.GradientDescentOptimizer(0.01).minimize(cross_entropy)
```

getModel "model_from_TF" to display the training results





VARIABLE IMPORTANCES



TRAINING METRICS - CONFUSION MATRIX VERTICAL: ACTUAL; ACROSS: PREDICTED

	0	1	2	3	4	5	6	7	8	9	Error	Rate
0	902	0	13	1	0	3	8	2	17	3	**0.0495**	**47 / 949**
1	0	1110	12	5	1	3	1	1	17	3	**0.0373**	**43 / 1,153**
2	11	5	896	20	12	1	18	6	27	6	**0.1058**	**106 / 1,002**
3	0	4	25	921	2	23	5	13	25	8	**0.1023**	**105 / 1,026**
4	0	5	9	1	854	0	7	1	11	38	**0.0778**	**72 / 926**
5	7	3	9	33	3	772	11	3	35	6	**0.1247**	**110 / 882**
6	12	2	14	0	6	27	935	0	5	0	**0.0659**	**66 / 1,001**
7	3	6	18	9	10	0	0	960	10	38	**0.0892**	**94 / 1,054**
8	6	6	16	23	4	22	6	3	847	16	**0.1075**	**102 / 949**
9	7	1	2	18	32	4	0	30	15	890	**0.1091**	**109 / 999**
Total	**948**	**1142**	**1014**	**1031**	**924**	**855**	**991**	**1019**	**1009**	**1008**	**0.0859**	**854 / 9,941**