HW13 CS 189

1.
Who else did you work with on this homework? In case of course events, just describe the group. How did you work on this homework? Any comments about the homework?

I worked on this homework with Ehimare Okoyomon, Prashanth Ganeth, and Daniel Mockaitis. We worked by getting together throughout the week and communicating on facebook.

I certify that all solutions are entirely in my words and that I have not looked at another student's solutions. I have credited all external sources in this write up}

Nicholas Lorio, 26089160

Question 5 Own Question.

What are disadvantages of using Gradient Boosting?

GB is an ensemble learning method and models by combining outputs from individual trees.

GBs build trees one at a time and each new tree works to fix the errors from the previously trained tree. As each tree gets added, we get an increasingly expressive model. GBs utilize three parameters – the number of trees, the depth of trees, and the learning rate.

The disadvantage of GBs is that training takes a longer period of time because it must follow this sequential pattern of the current tree learning from the previous tree. When not taking into account the speed, GBs perform better than random forests. The tradeoff is speed.

2.a. $\mathcal{L}: V \to \mathbb{R}$, $V = \mathbb{R}^n$ to real values $\mathbb{R}$

$$\mathcal{L}(v) = \frac{1}{2}\|y - v\|_2^2$$

$v^{t+1} = v^t - \alpha_t \nabla_v \mathcal{L}(v)\big|_{v=v^t}$ $\quad \alpha_t$, scalar > 0

- $\nabla_v \mathcal{L}(v) = \nabla_v\left(\frac{1}{2}\|y-v\|_2^2\right) = \nabla_v\left(\frac{1}{2}(y-v)^T(y-v)\right)$

$$\nabla_v\left(\frac{1}{2}(y^Ty + v^Tv - 2v^Ty)\right) = \frac{1}{2}(0 + 2v - 2y) = v - y$$

- $v^{t+1} = v^t - \alpha_t \cdot (v^t - y)$

$$v^{t+1} = v^t(1-\alpha_t) + \alpha_t y$$

- $\nabla_v \mathcal{L}(v)\big|_{v=Xw} = Xw - y$ $\quad$ • let $v^t = Xw^t$

$$= Xw^t - \alpha_t \cdot (Xw^t - y)$$

$$v^{t+1} = Xw^t(1-\alpha_t) + \alpha_t y$$

- expression for Xw^{t+1} using ① $w^{t+1} = w^t - \alpha_t X^T(Xw^t - y)$

$$Xw^{t+1} = X(w^t - \alpha_t X^T(Xw^t - y)) = Xw^t - X\alpha_t X^T(Xw^t - y)$$

$$- \alpha_t XX^T(Xw^t - y)$$

$$Xw^t - \alpha_t XX^T(Xw^t - y)$$

- Compare/interpret differences

$$v^{t+1} = v^t - \alpha_t(v^t - y)$$

$$Xw^{t+1} = Xw^t - \alpha_t XX^T(Xw^t - y)$$

interpret differences (?)

- $w \in \mathbb{R}^d \Rightarrow Xw \in \mathbb{R}^n$
- $v \in \mathbb{R}^n$

$\Rightarrow$ they are not equivalent

2. a. Cont

* interpret differences between updates

$w \in \mathbb{R}^d$ -> after we find the gradient descent
Next iteration we weight the feature
$v \in \mathbb{R}^k$ Matrix X by the greedy optimal
direction(w) in the $\mathbb{R}^k$ subspace of $\mathbb{R}^d$

$$Xw^{t+1} \in \mathbb{R}^k$$

-> the GD using these different loss function
mappings will only give the same result
if the direction taken is the one that
is closest to the gradient in some restricted space

-> the updates for v^{t+1} are computed without
having the feature matrix, Xw is plugged in
after the fact to determine an optimal direction,
& we MM loss without having Xw, & map directly
from $\mathbb{R}^k$ to the scalar opt direction as opposed
to our trained Sword loss error w/ Feature matrix appear

o if X was orthogonal then the updates
Xw^{t+1} & v^{t+1} would be equal.
-> XX^T is a projection onto subspace X.
-> we project the greedy update vector onto the
subspace spanned by the data.

2.b. $\tilde{V} = \{\tilde{v} = Xw : \|w\|_2 \leq 1, w \in \mathbb{R}^d\} \subset V$

$v^t = Xw^t$ (By Hint)

i)

$\tilde{v}^t = \arg\max \langle -\nabla R(v^t), Xw \rangle = X \cdot \arg\max \langle -\nabla R(Xw^t), Xw \rangle$

$\tilde{v} \in \tilde{V}, \tilde{v} = Xw, \|w\| \leq 1$ $\quad$ $w, \|w\|_2 \leq 1$

$\tilde{V}$ on this domain $= Xw$, using 2.a.

$= X \cdot \arg\max \langle -\nabla R(Xw^t), Xw \rangle = X \cdot \arg\max_w -(Xw^t - y)^T Xw$

$-\nabla_w (w^T X^T Xw - y^T Xw) = -X^T Xw^t + X^T y = -X^T (Xw^t - y)$ *

$X \cdot (-X^T (Xw^t - y)) = -XX^T (Xw^t - y)$ (is our value of $\tilde{v}^t$)

* (Set equal to zero "$= 0$")

• thus $\tilde{v}^t$ aligns w/ $-XX^T(Xw^t - y)$ as required.

• Maximized when the two vectors are colinear

ii) $v^{t+1} = v^t + \alpha_t \tilde{v}^t = Xw^t - \alpha_t \cdot XX^T (Xw^t - y)$

2.c. $X \in \mathbb{R}^{n \times d}$ strong feature matrix, $r^t = y - X w^t$

$X \begin{cases} \text{col} & \phi_i(x) \quad i=1,\dots,d \\ \text{rows} & x_1, \dots x_n \end{cases} \quad | \quad x = (x_1, \dots, x_n)^T = (\phi_1(x), \dots, \phi_d(x))$

$t = \arg\max_{i=1,\dots,d} |\langle r^t, \phi_i(x) \rangle|$ (5)

$$w^{t+1} = w^t + \frac{\langle r^t, \phi_{i_t}(x) \rangle}{\|\phi_{i_t}(x)\|_2^2} \cdot \phi_{i_t}(x) \quad (6)$$

$v^t = w^t$

Find $\tilde{v}$ s.t. (5) $\iff$ (3) w/ $\tilde{v}^t = \text{sign}(\langle r^t, \phi_{i_t}(x) \rangle \phi_{i_t}(x))$

(5) case 1 : $\langle r^t, \phi_{i_t}(x) \rangle > 0$

$$\arg\max_{i=1,\dots,d} \langle r^t, \phi_{i_t}(x) \rangle = \max \langle y - X w^t, \phi_{i_t}(x) \rangle$$

$$\max_{i=1,\dots,d} (y - X w^t)^T \phi_{i_t}(x) = \max \{ (y - X w^t)^T \phi_{1_t}(x), (y - X w^t)^T \phi_{2_t}(x), \dots \}$$

$$= \min \{ (X w^t - y)^T \phi_{1_t}(x), \dots, (X w^t - y)^T \phi_{d_t}(x) \}$$

Let b be the minimum

4.a. ◦ compute distance between every 8 points
◦ find the shortest

$$O(n+n) \cong O(n)$$

4.b.

1-Dimensional: 3 cells (that cell & it's adjacent cells)

2-D: 9 cells 

d-dimensional: 3^d

the runtime is $O(3^d)$ if each cell has 1 point.

4.c. $\langle a,b \rangle = \|a\|\|b\| \cos\theta$, thus $\mathbb{P}(|\langle v^k, u \rangle| \leq |\langle v^s, u \rangle|)$

$= \mathbb{P}(|\langle v^k, u'' \rangle| \leq |\langle v^s, u'' \rangle|)$

$= \mathbb{P}(\|v^k\|\|u''\||\cos\theta| \leq \|v^s\|\|u''\||\cos\theta_s|)$ $\theta_s :=$ angle between u'' & v^s

$= \mathbb{P}(\|v^k\||\cos\theta| \leq \|v^s\||\cos\theta_s|)$

$= \mathbb{P}(|\cos\theta| \leq \frac{\|v^s\||\cos\theta_s|}{\|v^k\|})$ & $|\cos\theta_s| \leq 1$

$\leq \mathbb{P}(|\cos\theta| \leq \frac{\|v^s\|}{\|v^k\|})$ as required

4.D.

$$0 \le |\cos\theta| \le \frac{\|v^s\|}{\|v^l\|} \Rightarrow \frac{-\|v^s\|}{\|v^l\|} \le \cos\theta \le 0$$

$$\Rightarrow \cos^{-1}\left(-\frac{\|v^s\|}{\|v^l\|}\right) \ge \theta \ge \cos^{-1}(0)$$

$$= \pi - \cos^{-1}\left(\frac{\|v^s\|}{\|v^l\|}\right) \ge \theta \ge \pi/2$$

$$= -\cos^{-1}\left(\frac{\|v^s\|}{\|v^l\|}\right) \ge \theta - \pi \ge \pi/2 - \pi = -\pi/2$$

$$\frac{2}{\pi}\cos^{-1}\left(\frac{\|v^s\|}{\|v^l\|}\right) \ge \frac{2\theta}{\pi} - 2 \ge -1 \quad \cdot 2/\pi$$

$$1 - \frac{2}{\pi}\cos^{-1}\left(\frac{\|v^s\|}{\|v^l\|}\right) \ge \frac{2\theta}{\pi} - 1 \ge 0$$

- So for $|\cos\theta| \le \frac{\|v^s\|}{\|v^l\|}$, $0 \le \frac{2\theta}{\pi} - 1 \le 1 - \frac{2}{\pi}\cos^{-1}\left(\frac{\|v^s\|}{\|v^l\|}\right)$
- $P\left(|\cos\theta| \le \frac{\|v^s\|}{\|v^l\|}\right) = P\left(\left|\frac{2\theta}{\pi} - 1\right| \le 1 - \frac{2}{\pi}\cos^{-1}\left(\frac{\|v^s\|}{\|v^l\|}\right)\right)$

→ By Hint θ is uniform Distributed

$$= P\left(|\theta| \le 1 - \frac{2}{\pi}\cos^{-1}\left(\frac{\|v^s\|}{\|v^l\|}\right)\right) = 1 - \frac{2}{\pi}\cos^{-1}\left(\frac{\|v^s\|}{\|v^l\|}\right)$$

i.e. $\mathbb{P}(\text{algorithm Fails}) = \mathbb{P}(\text{@ least } \tilde{k} \text{ points are closer to } z \text{ than } x^{(1)} \text{ after projection } U)$

$$\leq \frac{1}{\tilde{k}} \sum_{i=1}^{\hat{k}} \mathbb{P}(x^{(i)} \text{ is closer to } z \text{ than } x^{(1)})$$

$$= \frac{1}{\tilde{k}} \sum_{i=1}^{\hat{k}} \mathbb{P}(|\langle v^{(i)}, U \rangle| \leq |\langle v^{(1)}, U \rangle|)$$

$$\leq \frac{1}{\tilde{k}} \sum_{i=1}^{\hat{k}} \mathbb{P}\left(|\cos\theta| \leq \frac{\|v^{(i)}\|}{\|v^{(1)}\|}\right)$$

$$= \frac{1}{\tilde{k}} \sum_{i=1}^{\hat{k}} \underbrace{1 - \frac{2}{\pi}\cos^{-1}\left(\frac{\|v^{(i)}\|}{\|v^{(1)}\|}\right)}_{\leq \frac{\|v^{(i)}\|}{\|v^{(1)}\|}}$$

$$\leq \frac{1}{\tilde{k}} \sum_{i=1}^{\hat{k}} \frac{\|v^{(i)}\|}{\|v^{(1)}\|}$$

4.F Show $\sum_{i=2}^{\infty} \frac{\|x^{(i)} - z\|_2}{\|x^{(i)} - z\|_2} = \sum_{i=2}^{\infty} \left(\frac{1}{i}\right)^{1/d}$

- $\|x^{(c \cdot r^d)} - z\| = r$

So $\|x^{(i)} - z\| = \left(\frac{c}{i}\right)^d = \frac{c^d}{i^d}$

$\Rightarrow \|x^{(i)} - z\| = c^d$

$\therefore$

$$= \sum_{i=2}^{\infty} \frac{c^d}{\frac{c^d}{i^d}} = \sum_{i=2}^{\infty} i^d$$

$$= \sum_{i=2}^{\infty} \left(\frac{1}{i}\right)^{1/d}$$

as required

$i = c \cdot r^d$

$\frac{i}{c} = r^d$

$\log_r\left(\frac{i}{c}\right) = d$

$\log_r(i/c) = d$

$\frac{\log_e(i/c)}{\log_e(r)} = d$

$\log(r) = \frac{\log(i/c)}{d} = \log\left(\frac{i}{c}\right)^{1/d}$

$\Rightarrow r = \left(\frac{i}{c}\right)^{1/d}$

$r = \left(\frac{c}{i}\right)^d$

4.i

exhaustive: $O(n)$

DCI: $O(dn^{(1-1/d)})$

Space partitioning: $O(3^d)$

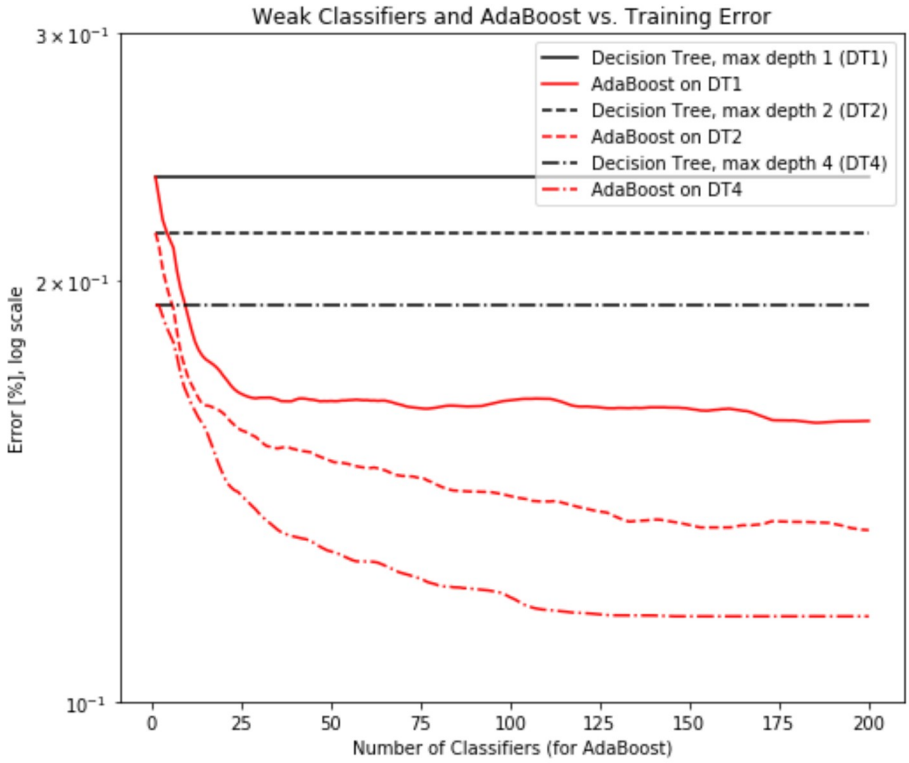
green: space partitioning

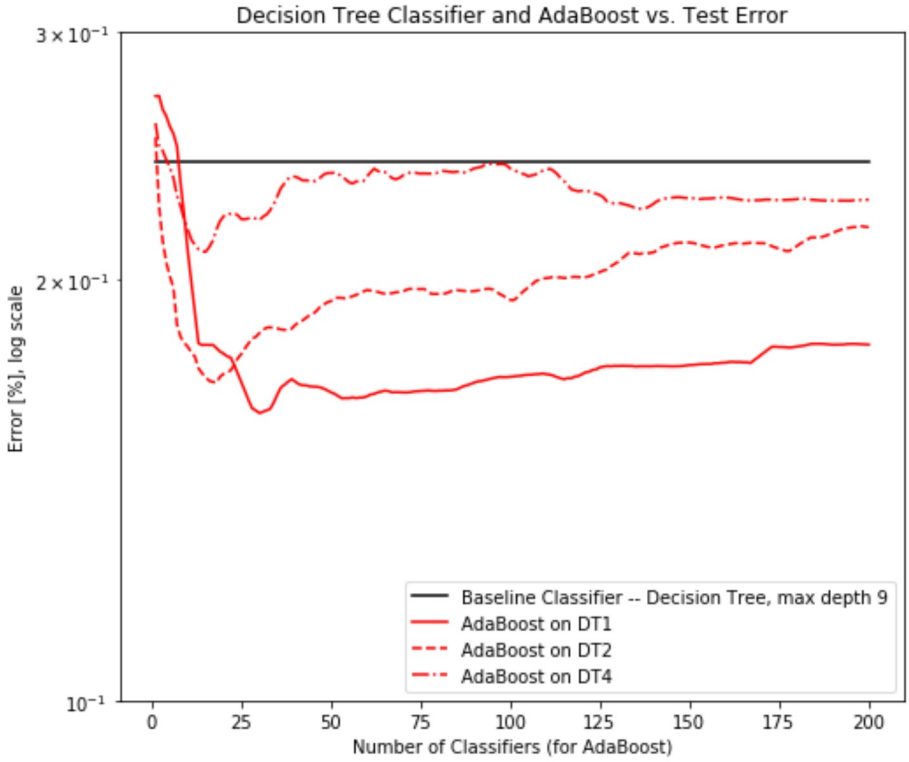
blue: exhaustive

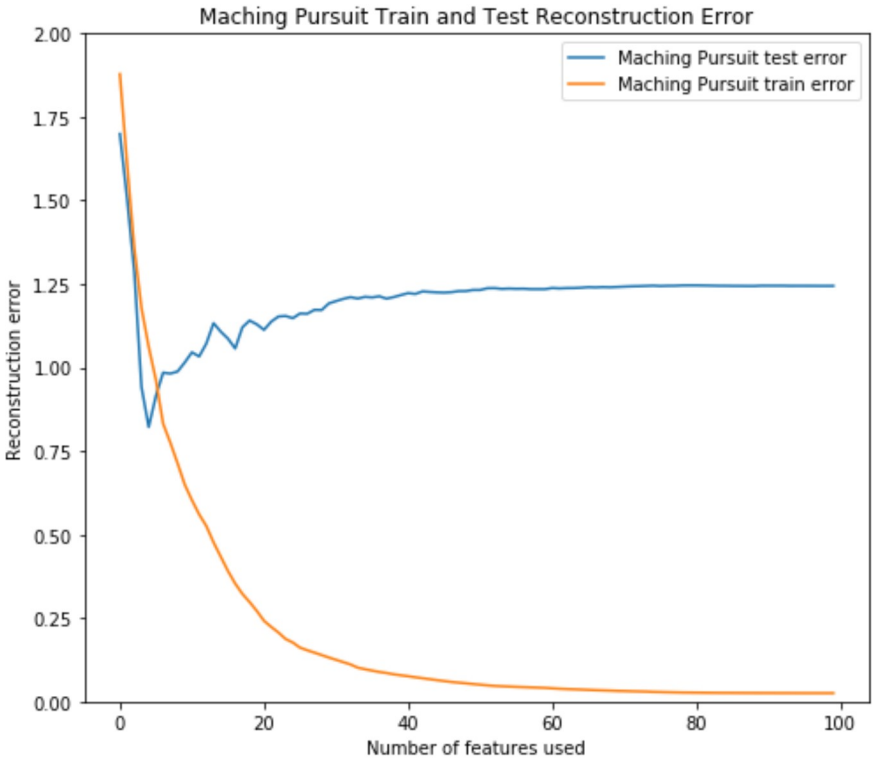
red: DCI

Question 2: Gradient Boosting & Early Stopping

In [5]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import loadmat

from sklearn.datasets import make_sparse_coded_signal
from sklearn.ensemble import AdaBoostClassifier, RandomForestClassifier
from sklearn.metrics import zero_one_loss
from sklearn.tree import DecisionTreeClassifier

# globals
n_estimators = 200
DT1 = DecisionTreeClassifier(max_depth=1, min_samples_leaf=15)
DT2 = DecisionTreeClassifier(max_depth=2, min_samples_leaf=15)
DT4 = DecisionTreeClassifier(max_depth=4, min_samples_leaf=15)
DT9 = DecisionTreeClassifier(max_depth=9, min_samples_leaf=1)

"""Loads the training data from the SPAM dataset used in HW12."""
def load_data():
    # load data
    data = loadmat("boosting/datasets/spam_data/spam_data.mat")
    # training data
    data_, labels_ = data["training_data"], np.squeeze(data["training_labels"])
    X_train, y_train = data_, labels_
    # test data
    y_test=[]
    with open("boosting/datasets/spam_data/spam_test_labels.txt","r") as f:
        for l in f.readlines():
            y_test.append(int(l.split(",")[1]))
    y_test = np.array(y_test)
    X_test = data['test_data']

    return X_train, y_train, X_test, y_test

"""Runs the maching pursuit algorithm."""
def mp(y, X, w_true, y_test, X_test):
    train_err = []
    test_err = []
    X_ = X
    y = np.copy(y); X = np.copy(X)
    curr = np.copy(y)
    w_est = np.zeros(len(X[0]))
    for j in range(len(X[0])):
        i = np.argmax(np.abs(np.dot(X.T, curr)))
        col = np.copy(X[:,i])
        # use each column only once
        X[:,i] = 0
        w_est[i] = np.dot(col, curr)
        curr = curr - col*w_est[i]
        # error defined here as ||y - D x_hat||_2
        train_err.append(np.linalg.norm(X_.dot(w_est) - y))
        test_err.append(np.linalg.norm(X_test.dot(w_est)-y_test))

    return w_est, train_err, test_err

if __name__ == "__main__":
    ###### CHANGE THESE VARIABLES TO RUN PROBLEM PARTS
    PART_H = True
    PART_I = True
    PART_K = True
    ######
    X_train, y_train, X_test, y_test = load_data()

    ### PART H
    if PART_H:
```


Weak Classifiers and AdaBoost vs. Training Error
Decision Tree, max depth 1 (DT1)
AdaBoost on DT1
Decision Tree, max depth 2 (DT2)
AdaBoost on DT2
Decision Tree, max depth 4 (DT4)
AdaBoost on DT4
Error [%], log scale
Number of Classifiers (for AdaBoost)


Decision Tree Classifier and AdaBoost vs. Test Error
Baseline Classifier -- Decision Tree, max depth 9
AdaBoost on DT1
AdaBoost on DT2
AdaBoost on DT4
Error [%], log scale
Number of Classifiers (for AdaBoost)



Part H Observations

- All adaboost function errors decrease as a function of increasing classifiers/number of trees we are fitting. Adaboost results in lower error than standard decision trees.
- Deeper trees result in lower training error.
- **Why?:** Deeper trees mean that the data points can be classified to a greater extent. Deeper trees = more classes, greater fit and less potential for erroneously mislabeling a data point at each iteration of adaboost algorithm. If less points are mislabeled at each iteration than those data points that are mislabeled get higher probabilities. This means they will be more likeley to be sampled from the feature matrix X to be classified in the tree of the next iteration of the adaboost algorithm. The deeper tree means that the data points that we're missclassified (and that given their higher probability of being sampled we're sampled) previously are less likely to be misclassified again in the next tree.
- Our loss function is exponential.

Part I Observations

- Yes, there is a difference in the behavior of the training and test error. The decision tree of depth 1 works best for AdaBoost algorithm on the test data. This is potentially due to overfitting which arise when the depth of the trees is too high.
- Decision tree depth 4 has lowest error for training data. Greater depth leads to lower bias and thus lower error. However, the plot of the testing error performance in part I shows that this overfits and does not generalize well to the testing data.
- Test error decreases as a function of boosting iterations in the beginning but eventually it starts to increase when the number of decision trees in Adaboostis pretty large.

Part J

- **Do you think limiting the number of base classifiers used for AdaBoost would help?** Yes. From the plots we see that the error initially decreases up to a certain point (to a minimum error) as a function of the number of base classifiers used for adaboost. Intuitively it makes sense that we would want to limit/use up to X number of classifiers before the error begins to increase again as it is shown to do for the test data in the plot of part I.

- **Which base classifier can we run more boosting iterations on before the test error starts increasing?**
- *Types of base classifiers(?): Tree based classifiers, Random Forests, Bagging, Bayes classifier*

Tree Based Decision Tree Classifiers are the most common.

"Each round of boosting chooses a new classifier from a set of potential classifiers constructed from training data weighted, or resampled, according to the mis-classifications of the previous round. The new classifier is selected so as to minimise the total ensemble error.

Each new base classifier is specifically required to focus on the weak points of the existing ensemble, with the result that boosting aggressively drives down training error."

In early stage of ensemble, boosting has few weak classifiers, each focused on different areas of training. Primarily reduces bias, as ensemble size grows, scope for bias reduction decreases and the error from variance is improved.

Instability in base classifiers for boosting is good because as the ensemble grows, the number of remaining mis-classified examples decreases. Thus a higher degree of difference is needed to generated a classifier with a different view of the remaning samples than its predecessors (something that we do not want).

We can run more boosting iterations on instable base classifiers before the test error starts increasisng.

Part K Observations

- **Explain the shape of the training error plot. Does the plot for test error look similar to the one from part (i)** Yes, the plot for test error looks very similar to the test error plot for adaboost as a function of number of classifiers utilized. Similarily the training data error plot looks very similar to the plot in part H, the curves follow an exponential decay as well, the loss function is exponential with respect to the number of features used.

- For MP, the behavior is the same as that of adaboost, however, its behavior is the same as a function of number of features used rather than of number of classifiers used for adaboost.

- **The shape of the training error plot:** As t increases matching pursuit builds an estimate using a greater number of features. It fits the data closer. In the case of the training data, it reduces the bias and hence the error. As discussed earlier for the adaboost algorithm this overfits as can be seen from the behavior of the test data. MP works on sparcity of the data set, if we used too many features than the estimator has been overfitted and performs worse (at a certain point) as can be seen in the plot.

Question 3: CNN On Fruits

3.a

In [6]:
```
import numpy as np
import tensorflow as tf
#import yolo.config_card as cfg

import IPython

slim = tf.contrib.slim


class CNN(object):

    def __init__(self,classes,image_size):
        '''
        Initializes the size of the network
        '''

        self.classes = classes
        self.num_class = len(self.classes)
        self.image_size = image_size

        self.output_size = self.num_class
        self.batch_size = 40

        self.images = tf.placeholder(tf.float32, [None, self.image_size,self.image_
size,3], name='images')


        self.logits = self.build_network(self.images, num_outputs=self.output_size)

        self.labels = tf.placeholder(tf.float32, [None, self.num_class])

        self.loss_layer(self.logits, self.labels)
        self.total_loss = tf.losses.get_total_loss()
        tf.summary.scalar('total_loss', self.total_loss)

    def build_network(self,
                      images,
                      num_outputs,
                      scope='yolo'):

        with tf.variable_scope(scope, reuse=tf.AUTO_REUSE):
            with slim.arg_scope([slim.conv2d, slim.fully_connected],
                                weights_initializer=tf.truncated_normal_initializer
(0.0, 0.01),
                                weights_regularizer=slim.l2_regularizer(0.0005)):

                '''
                Fill in network architecutre here
                Network should start out with the images function
                Then it should return net
                '''

                ###SLIM BY DEFAULT ADDS A RELU AT THE END OF conv2d and fully_conne
cted
                ###SLIM SPECIFYING A CONV LAYER WITH 5 FILters as SIZE 15 by 15
                net = slim.conv2d(images, 5, [15,15], scope='conv_0')
                ### SLIM USING MAX POOLING ON THE NETWORK. THE POOLING REGION CONSI
DERED IS 3 by 3
                net = slim.max_pool2d(net, [3, 3], scope='pool')
                ## Need to flatten because convolution happens in 2D arrays
                ## and fully connected layers are usually done in 1D arrays.
                net = slim.flatten(net)
                ###SLIM SPECIFYING A FULLY CONNECTED LAYER WHOSE OUT IS 512
                net = slim.fully_connected(net, 512, scope='fc_2')
```

In [7]:
```
from release_code_cnn_tensorflow.data_manager import data_manager
#from release_code_cnn_tensorflow.cnn import CNN
from release_code_cnn_tensorflow.trainer import Solver
import tensorflow as tf
import random

from release_code_cnn_tensorflow.confusion_mat import Confusion_Matrix

random.seed(0)

CLASS_LABELS = ['apple','banana','nectarine','plum','peach','watermelon','pear','mango','grape','orange','strawberry','pineapple',
    'radish','carrot','potato','tomato','bellpepper','broccoli','cabbage','cauliflower','celery','eggplant','garlic','spinach','ginger']

image_size = 90
classes = CLASS_LABELS
dm = data_manager(classes, image_size)

cnn = CNN(classes,image_size)

sess = tf.Session()
sess.run(tf.global_variables_initializer())

val_data = dm.val_data
train_data = dm.train_data



cm = Confusion_Matrix(val_data,train_data,CLASS_LABELS,sess)

cm.test_net(cnn)
```