DS-GA 1003: Homework 5
Trees and Boosting

Due on Monday, April 4, 2016

Professor David Ronsenberg

See complete code at: *git@github.com:cryanzpj/1003.git*

Yuhao Zhao
Yz3085

2 Decision Trees

- **2.1 Trees on the Banana Dataset.**

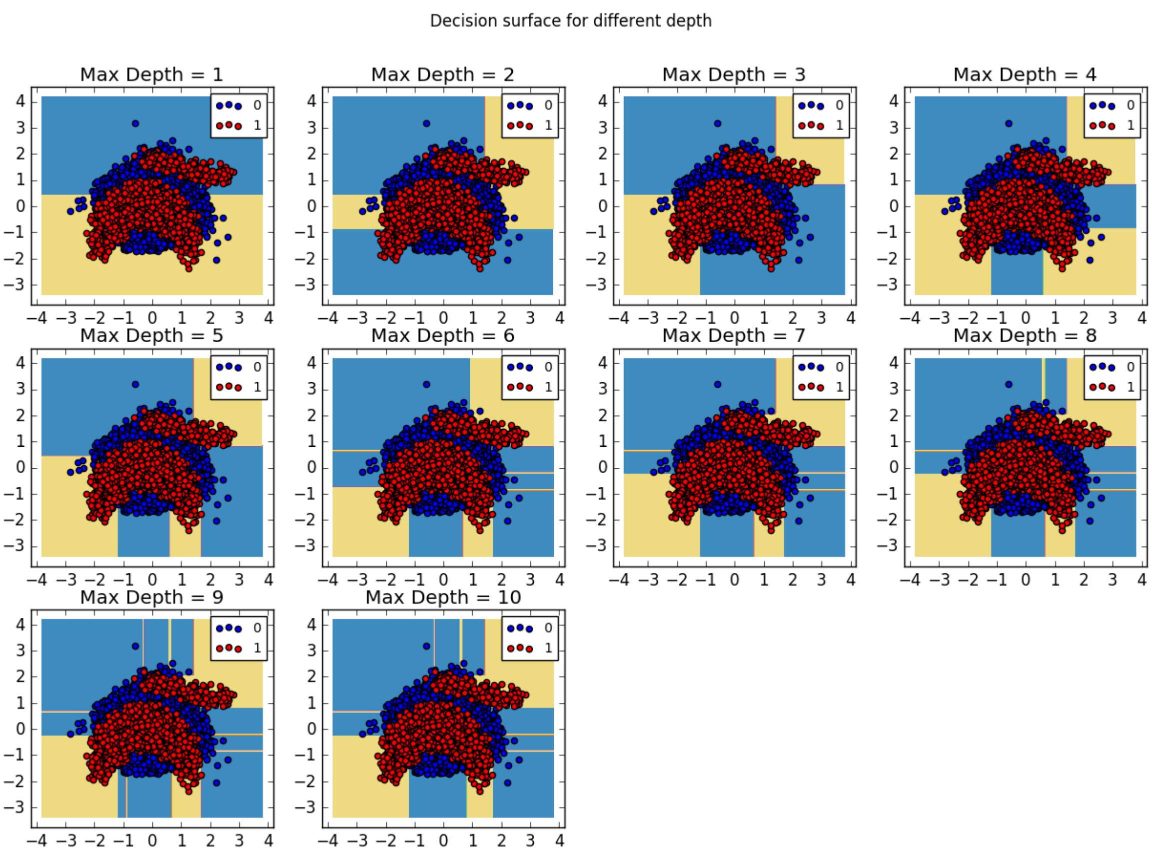
– **2.1.1.**

```
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier

file_train = open('data/banana_train.csv')
file_test = open('data/banana_test.csv')

train = np.array(map(lambda x: x[:2] + [x[-1].strip()],
                         [i.split(',') for i in file_train]), dtype='float')
test = np.array(map(lambda x: x[:2] + [x[-1].strip()],
                         [i.split(',') for i in file_test]), dtype='float')

y_train = np.array([0 if i == -1 else 1 for i in train[:, 0]])
y_test = np.array([0 if i == -1 else 1 for i in test[:, 0]])
X_train = train[:, 1:]
X_test = test[:, 1:]
```

– **2.1.2.**

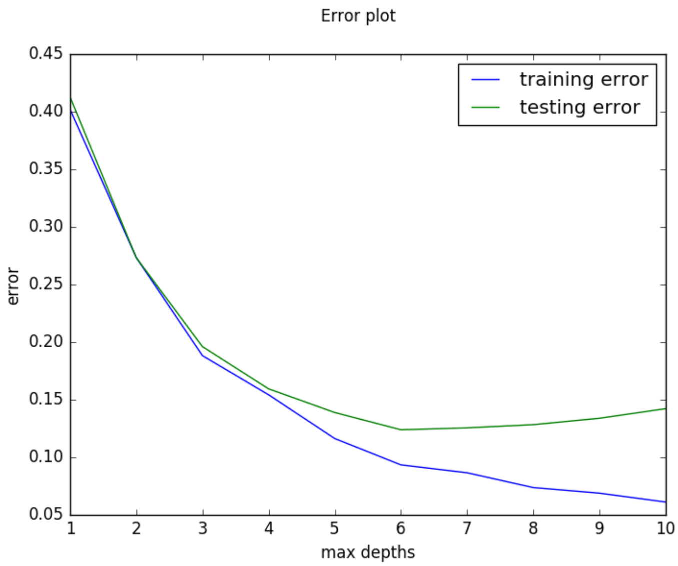
```
n_classes = 2
plot_colors = "bry"
plot_step = 0.02

error = np.zeros((2, 10))

for i in xrange(1, 11):
        idx = np.arange(X_train.shape[0])
        np.random.seed(1)
        np.random.shuffle(idx)
        X = X_train[idx]
        y = y_train[idx]

        mean = X.mean(axis=0)
        std = X.std(axis=0)
        X = (X - mean) / std

        clf = DecisionTreeClassifier(max_depth=i).fit(X, y)
        plt.subplot(2, 5, i)
        x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
        y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
        xx, yy = np.meshgrid(np.arange(x_min, x_max, plot_step),
        np.arange(y_min, y_max, plot_step))

        Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
```

```
        Z = Z.reshape(xx.shape)

        training_error = np.sum(np.equal(clf.predict(X_train),
                                         1 - y_train)) / float(y_train.shape[0])
        testing_error = np.sum(np.equal(clf.predict(X_test),
                                         1 - y_test)) / float(y_test.shape[0])
        error[:, i - 1] = np.array([training_error, testing_error])

        cs = plt.contourf(xx, yy, Z, cmap=plt.cm.Paired)

        plt.axis("tight")

        for i, color in zip(range(n_classes), plot_colors):
                idx = np.where(y == i)
                plt.scatter(X[idx, 0], X[idx, 1], c=color, label=str(i),
                                cmap=plt.cm.Paired)

        plt.axis("tight")
        plt.legend(fontsize=10)
plt.suptitle('Decision surface for different depth')
plt.show()
```



From the plot we can see that, the decision surface has very little change after max depth = 6. Most of the misclassification occurs at the center.

– **2.1.3.**



From the plot we see that the model is clearly over-fitting for large max depths. For max depth greater than 6 the testing error is increasing.

– **2.1.4.**

```
min_error = 1
ite = 0
for a in xrange(1, 21):
    for b in xrange(1, 21):
        for c in xrange(1, 21):
            idx = np.arange(X_train.shape[0])
            np.random.seed(1)
            np.random.shuffle(idx)
            X = X_train[idx]
            y = y_train[idx]

            mean = X.mean(axis=0)
            std = X.std(axis=0)
            X = (X - mean) / std
            #normalize testing data
            X_test_temp = (X_test -mean)/std

            clf = DecisionTreeClassifier(max_depth=a, min_samples_leaf=b,
                                 min_samples_split=c).fit(X, y)

            testing_error = np.sum(np.equal(clf.predict(X_test_temp), 1 - y_test)) /
                                 float(y_test.shape[0])
            if testing_error < min_error:
                min_error = testing_error
```

```
                par = [a, b, c]
            ite += 1
>>> min_error
>>> 0.1111111111111
>>> par
>>> [10, 11, 1]
```

I searched max depth, min samples leaf, min samples split from 1 to 20, the best testing error is 0.1111111111111, and the corresponding parameters are 10,11 and 1

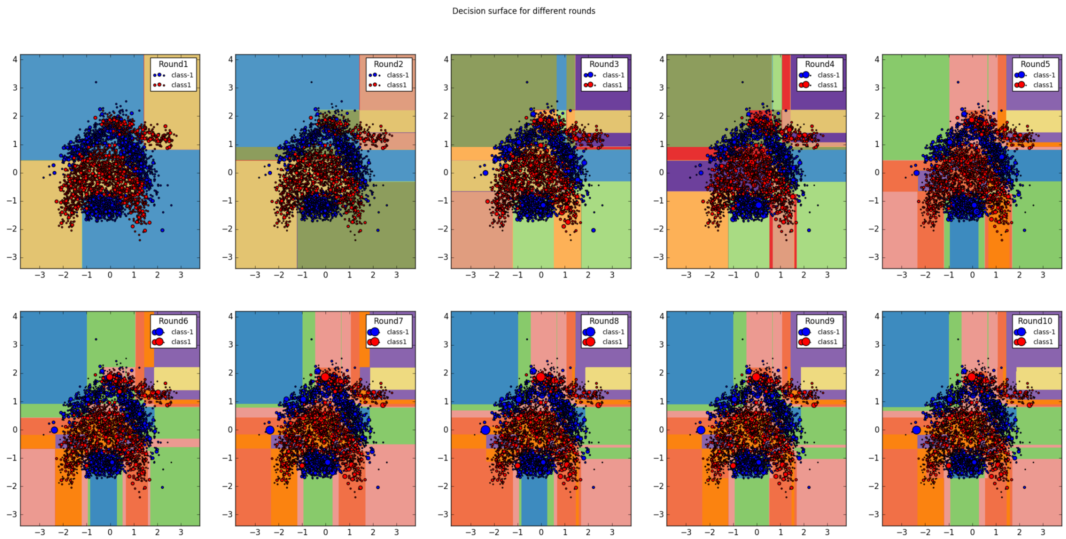
3 Ada Boost

- **3.1 Implementation.**

  – **3.1.1.**

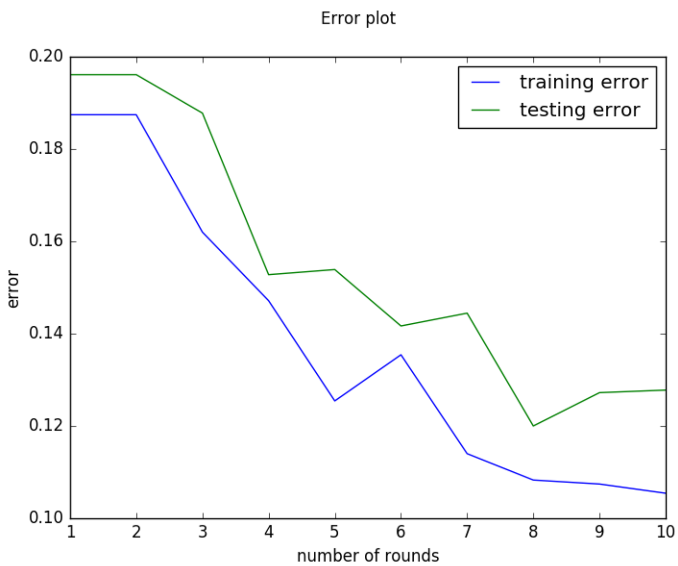
```
def AdaBoost(X, y,n_round=5,test_x = None,test_y= None,visual = False):
    '''
    :param nd-array X: Training data
    :param 1d-array y: Training labels
    :param int n_round: Number of rounds default = 5
    :param nd-array test_x: Testing data default None
    :param 1d-array test_y: Testing labels default None
    :param Boolean visual: True for visualize default None
    :return: trainining error/ testing error of the Final model
    '''

    n_instance = X.shape[0]
    w = np.ones(n_instance) / n_instance
    models = np.zeros(n_round,dtype = object)
    error = np.zeros(n_round)
    alphas = np.zeros(n_round)

    mean = X.mean(axis=0)
    std = X.std(axis=0)
    X = (X - mean) / std


    for n in xrange(1, n_round + 1):
        W = np.sum(w)
        clf = DecisionTreeClassifier(max_depth=3).fit(X,y,sample_weight=w)
        models[n-1] = clf
        error = np.sum((1-np.equal(clf.predict(X), y)) * w)/W
        alphas[n-1] = np.log((1-error)/error)
        w = np.exp((1- np.equal(clf.predict(X),y))*alphas[n-1])*w

        #visualizer
        if visual:
            plot_colors = 'br'
            plt.subplot(2, 5, n)
```

```
            x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
            y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
            xx, yy = np.meshgrid(np.arange(x_min, x_max, plot_step),
                                 np.arange(y_min, y_max, plot_step))

            Z = np.sum(map(lambda i:alphas[i] * models[i].predict(np.c_[xx.ravel(),
                                     yy.ravel()]),list(xrange(n))),0)
            Z = Z.reshape(xx.shape)
            cs = plt.contourf(xx, yy, Z, cmap=plt.cm.Paired)
            plt.axis("tight")

            for i, color in zip([-1,1], plot_colors):
                idx = np.where(y == i)
                plt.scatter(X[idx, 0], X[idx, 1],s=10000*(w/W) , c=color,
                                        label="class" + str(i), cmap=plt.cm.Paired)
                plt.legend(fontsize=10,title = 'Round' + str(n))
    plt.suptitle('Decision surface for different rounds')
    plt.show()

    # record rounds errors
    G_n = np.array(map(lambda i: alphas[i] * models[i].predict(X), list(xrange(n))))
    train_error = 1 - np.sum(np.equal(np.sign(np.sum(G_n,0)),y))/float(n_instance)

    if  (test_x != None) and  (test_y != None) :
        G_n_test = np.array(map(lambda i: alphas[i] * models[i].predict(test_x),
                            list(xrange(n))))
        test_error = 1 - np.sum(np.equal(np.sign(np.sum(G_n_test,0)),test_y))/
                           float(test_y.shape[0])
        return [train_error,test_error]

    else:
        return train_error
>>> AdaBoost(X_train,y_train,10,visual = True)
```



From the plot we can see that some points near the decision surface are getting larger weights with further boosting.

– **3.1.3.**



From the plot we see that the both of the training and testing error are decreasing when adding rounds. The over-fitting is not clear yet, so we may try to add more rounds to see how the AdaBoost works on this data set.

4 Gradient Boosting Machines

- **4.1.**

For square loss: $l(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$

From the Gradient Boosting Machines,

$$(g_m)_i = \frac{\partial}{\partial f(x_i)} \sum_{i=1}^{n} \frac{1}{2}(y_i - f(x_i))^2|_{f(x_i)=f_{m-1}(x_i)} \tag{1}$$
$$= \frac{\partial}{\partial f(x_i)} \frac{1}{2}(y_i - f(x_i))^2|_{f(x_i)=f_{m-1}(x_i)} \tag{2}$$
$$= -(y_i - f(x_i))|_{f(x_i)=f_{m-1}(x_i)} \tag{3}$$
$$= -(y_i - f_{m-1}(x_i)) \tag{4}$$

Then

$$h_m = \underset{h\in\mathcal{F}}{\operatorname{argmin}} \sum_{i=1}^{n}((-g_m)_i - h(x_i))^2 \tag{5}$$
$$= \underset{h\in\mathcal{F}}{\operatorname{argmin}} \sum_{i=1}^{n}((y_i - f_{m-1}(x_i)) - h(x_i))^2 \tag{6}$$

- **4.2.**

For logistic loss: $l(m) = ln(1 + e^{-m})$
From the Gradient Boosting Machines,

$$(g_m)_i = \frac{\partial}{\partial f(x_i)} \sum_{i=1}^{n} ln(1 + e^{-y_i f(x_i)})|_{f(x_i)=f_{m-1}(x_i)} \tag{7}$$
$$= \frac{\partial}{\partial f(x_i)} ln(1 + e^{-y_i f(x_i)})|_{f(x_i)=f_{m-1}(x_i)} \tag{8}$$
$$= \frac{-y_i e^{-y_i f(x_i)}}{1 + e^{-y_i f(x_i)}}|_{f(x_i)=f_{m-1}(x_i)} \tag{9}$$
$$= \frac{-y_i e^{-y_i f_{m-1}(x_i)}}{1 + e^{-y_i f_{m-1}(x_i)}} \tag{10}$$

Then

$$h_m = \underset{h\in\mathcal{F}}{\operatorname{argmin}} \sum_{i=1}^{n}((-g_m)_i - h(x_i))^2 \tag{11}$$
$$= \underset{h\in\mathcal{F}}{\operatorname{argmin}} \sum_{i=1}^{n}(\frac{y_i e^{-y_i f_{m-1}(x_i)}}{1 + e^{-y_i f_{m-1}(x_i)}} - h(x_i))^2 \tag{12}$$

5 From Margins to Conditional Probabilities

- **5.1.**

Since $y \in \{1, -1\}$

$$E_y[l(yf(x))|x] = P(y = -1|x)l(-f(x)) + P(y = 1|x)l(f(x)) \tag{13}$$
$$= (1 - P(y = 1|x))l(-f(x)) + P(y = 1|x)l(f(x)) \tag{14}$$
$$= (1 - \pi(x))l(-f(x)) + \pi(x)l(f(x)) \tag{15}$$

- **5.2.**

For exponential loss $l(y, f(x)) = e^{-yf(x)}$:

$$f^* = \underset{f}{\operatorname{argmin}}\ E_y[l(yf(x))|x] \tag{16}$$
$$= \underset{f}{\operatorname{argmin}}\ (1-\pi(x))e^{f(x)} + \pi(x)e^{-f(x)} \tag{17}$$

If we take partial derivative of the target function w.r.t f:

$$\frac{\partial}{\partial f}(1-\pi(x))e^{f(x)} + \pi(x)e^{-f(x)} = (1-\pi(x))e^{f(x)} - \pi(x)e^{-f(x)} = 0 \tag{18}$$

We have:

$$e^{2f(x)} = \frac{\pi(x)}{1-\pi(x)} \tag{19}$$
$$f^*(x) = f(x) = \frac{1}{2}ln(\frac{\pi(x)}{1-\pi(x)}) \tag{20}$$

On the contrary, if we are given f^*, we can solve $\pi(x)$ from eqn(20):

$$\frac{1}{2}ln(\frac{\pi(x)}{1-\pi(x)}) = \frac{1}{2}ln(\frac{1}{1-\pi(x)} - 1) = f^*(x) \tag{21}$$
$$\frac{1}{1-\pi(x)} = e^{2f^*(x)} + 1 \tag{22}$$
$$\pi(x) = \frac{e^{2f^*(x)}}{e^{2f^*(x)}+1} = \frac{1}{1+e^{-2f^*(x)}} \tag{23}$$
$$\tag{24}$$

- **5.3.**

For the logistic loss function $l(y, f(x)) = ln(1+e^{-yf(x)})$:

$$f^* = \underset{f}{\operatorname{argmin}}\ E_y[l(yf(x))|x] \tag{25}$$
$$= \underset{f}{\operatorname{argmin}}\ (1-\pi(x))ln(1+e^{f(x)}) + \pi(x)ln(1+e^{-f(x)}) \tag{26}$$

If we take partial derivative of the target function w.r.t f:

$$(1-\pi(x))\frac{e^{f(x)}}{1+e^{f(x)}} + \pi(x)\frac{-e^{-f(x)}}{1+e^{-f(x)}}) = 0 \tag{27}$$

We have:

$$\frac{(1-\pi(x))e^{f(x)}}{1+e^{f(x)}} - \frac{\pi(x)}{e^{f(x)}+1} = 0 \tag{28}$$

Since $e^{f(x)} + 1 > 0$:

$$\pi(x) = (1-\pi(x))e^{f(x)} \tag{29}$$
$$f^*(x) = f(x) = ln(\frac{\pi(x)}{1-\pi(x)}) \tag{30}$$

If we are given f^*, we can solve $\pi(x)$ from eqn(30):

$$f^*(x) = ln(\frac{1}{1-\pi(x)} - 1) \tag{31}$$

$$\pi(x) = 1 - \frac{1}{e^{f^*(x)}+1} \tag{32}$$

$$= \frac{e^{f^*(x)}}{e^{f^*(x)}+1} \tag{33}$$

$$= \frac{1}{1+e^{-f^*(x)}} \tag{34}$$

- **5.4.**

For the hinge loss $l(y, f(x)) = max(0, 1 - yf(x))$:

$$f^* = \underset{f}{\text{argmin}}\ E_y[l(yf(x))|x] \tag{35}$$

$$= \underset{f}{\text{argmin}}\ (1-\pi(x))max(0, 1+f(x)) + \pi(x)max(0, 1-f(x)) \tag{36}$$

$$= \underset{f}{\text{argmin}}\ F \tag{37}$$

i) if $f \leq -1$:

$$F = \pi(x)(1-f(x)) \tag{38}$$

$$\frac{\partial F}{\partial f} = -\pi(x) < 0 \tag{39}$$

The function F is decreasing, therefore to minimize F we choose $f^*(x)$ = -1
ii) if $f \geq 1$:

$$F = (1-\pi(x))(1+f(x)) \tag{40}$$

$$\frac{\partial F}{\partial f} = 1 - \pi(x) > 0 \tag{41}$$

The function F is increasing, therefore to minimize F we choose $f^*(x)s$ 1
iii) if $-1 \leq f \leq 1$:

$$F = (1-\pi(x))(1+f(x)) + \pi(x)(1-f(x)) \tag{42}$$

$$\frac{\partial F}{\partial f} = 1 - 2\pi(x) \tag{43}$$

For $\pi(x) \leq \frac{1}{2}$, $\frac{\partial F}{\partial f} \geq 0$, the function is increasing, then we choose $f^*(x) = -1$
For $\pi(x) \geq \frac{1}{2}$, $\frac{\partial F}{\partial f} \leq 0$, the function is decreasing, then we choose $f^*(x) = 1$
From above it's equivalent to $f^*(x) = sign(\pi(x) - \frac{1}{2})$

6 AdaBoost Actually Works

- **Exponential Bound on the training loss.**

  – **6.1.**

for any function g into {-1, +1}
If $g(x) \neq y, yg(x) = -1, exp(-yg(x)) = e^{-1}, I(g(x) \neq y) = 1 < exp(1)$
If $g(x) = y, yg(x) = 1, exp(-yg(x)) = e^{1}, I(g(x) \neq y) = 0 < exp(-1)$
Therefore, $I(g(x) \neq y) < exp(-yg(x))$

– 6.2.

$L(G,D) = \frac{1}{n}\sum I(G(x_i) \neq y_i)$
From 6.1, $I(G(x_i) \neq y_i) < exp(-y_i G(x_i)) = exp(-y_i f_t(x_i))$
Therefore, $L(G,D) = \frac{1}{n}\sum I(G(x_i) \neq y_i) < \frac{1}{n}\sum exp(-y_i f_t(x_i)) = Z_T$

– 6.3.

$$\begin{aligned} w_i^{t+1} &= w_i^t exp(-\alpha_t y_i G_t(x_i)) && (44)\\ &= exp(y_i \sum_{s=1}^{t} -\alpha_s G_s(x_i)) && (45)\\ &= exp(-y_i f_t(x_i)) && (46) \end{aligned}$$

– 6.4.

$$\begin{aligned} \frac{Z_{t+1}}{Z_t} &= \frac{\frac{1}{n}\sum_{i=1}^{n} exp(-y_i f_{t+1}(x_i))}{\frac{1}{n}\sum_{i=1}^{n} exp(-y_i f_t(x_i))} && (47)\\ &= \frac{\sum w_i^{t+2}}{\sum w_i^{t+1}} && (48)\\ &= \frac{\sum w_i^{t+1} exp(-\alpha_{t+1} y_i G_{t+1}(x_i))}{\sum w_i^{t+1}} && (49)\\ &= \frac{\sum_{y_i = G_{t+1}(x_i)} w_i^{t+1} exp(-\alpha_{t+1}) + \sum_{y_i \neq G_{t+1}(x_i)} w_i^{t+1} exp(\alpha_{t+1})}{\sum w_i^{t+1}} && (50)\\ &= (1 - err_{t+1})e^{-\alpha_{t+1}} + err_{t+1}e^{\alpha_{t+1}} && (51)\\ &= (1 - err_{t+1})e^{-\frac{1}{2}log(\frac{1-err_{t+1}}{err_{t+1}})} + err_{t+1}e^{-\frac{1}{2}log(\frac{1-err_{t+1}}{err_{t+1}})} && (52)\\ &= 2\sqrt{err_{t+1}(1 - err_{t+1})} && (53) \end{aligned}$$

– 6.5.

$g(a) = a(1-a)$, $g'(a) = 1 - 2a > 0$ for $a \in [0, \frac{1}{2}]$, so g is monotonically increasing on $[0, \frac{1}{2}]$
$h(a) = exp(-a) + a - 1, h'(a) = -ae^{-a} + 1 \geq 0$ for $a \in [0, \frac{1}{2}]$, $h(a) \geq h(0) = 0$, so $1 - a \leq exp(-a)$
From 6.4 $\frac{Z_{t+1}}{Z_t} = 2\sqrt{err_{t+1}(1 - err_{t+1})}$, we know $err_{t+1} \leq \frac{1}{2} - \gamma$:

$$\begin{aligned} \frac{Z_{t+1}}{Z_t} \leq 2\sqrt{(\frac{1}{2} - \gamma)(\frac{1}{2} + \gamma)} &= \sqrt{1 - 4\gamma^2} && (54)\\ 1 - 4\gamma^2 \leq e^{-4\gamma^2}, \sqrt{1 - 4\gamma^2} &\leq e^{-2\gamma^2} && (55)\\ & && (56) \end{aligned}$$

Therefore:

$$\frac{Z_{t+1}}{Z_t} \leq e^{-2\gamma^2}$$

– 6.6.

$$Z_{t+1} = \frac{Z_{t+1}}{Z_t}\frac{Z_t}{Z_{t-1}}... \leq e^{-2t\gamma^2} \qquad (57)$$

Therefore the error has exponential decay.

7 AdaBoost is FSAM With Exponential Loss

– **7.1.**

Since $L(y, f(x)) = exp(-yf(x))$

$$(\alpha_t, G_t) = \underset{\alpha,G}{\operatorname{argmin}} \sum_{i=1}^{n} e^{-y_i(f_{t-1}(x_i)+\alpha G(x_i))} \tag{58}$$

$$= \underset{\alpha,G}{\operatorname{argmin}} \sum_{i=1}^{n} e^{-y_i f_{t-1}(x_i)} e^{-y_i \alpha G(x_i)} \tag{59}$$

$$= \underset{\alpha,G}{\operatorname{argmin}} \sum_{i=1}^{n} w_i^t e^{-y_i \alpha G(x_i)} \tag{60}$$

– **7.2.**

For fixed positive alpha:

$$G_t = \underset{G}{\operatorname{argmin}} \sum_{y_i = G(x_i)} w_i^t e^{-\alpha} + \sum_{y_i \neq G(x_i)} w_i^t e^{\alpha} \tag{61}$$

$$= \underset{G}{\operatorname{argmin}} e^{-\alpha} \sum_{y_i = G(x_i)} w_i^t + e^{\alpha} \sum_{y_i \neq G(x_i)} w_i^t \tag{62}$$

$$= \underset{G}{\operatorname{argmin}} e^{-\alpha} \sum_{i=1}^{n} w_i^t + (e^{\alpha} - e^{-\alpha}) \sum_{y_i \neq G(x_i)} w_i^t \tag{63}$$

Since $e^{-\alpha}, e^{\alpha}$ are positive constant, $e^{\alpha} - e^{-\alpha} > 0$:

$$G_t = \underset{G}{\operatorname{argmin}} \sum_{y_i \neq G(x_i)} w_i^t = \underset{G}{\operatorname{argmin}} \sum_{i=1}^{n} w_i^t I(G(x_i) \neq y_i)$$

– **7.3.**

$$\alpha_t = \underset{\alpha}{\operatorname{argmin}} \sum_{i=1}^{n} w_i^t e^{-y_i \alpha G_t(x_i)} \tag{64}$$

$$= \underset{\alpha}{\operatorname{argmin}} \sum_{y_i = G_t(x_i)} w_i^t e^{-\alpha} + \sum_{y_i \neq G_t(x_i)} w_i^t e^{\alpha} \tag{65}$$

$$= \underset{\alpha}{\operatorname{argmin}}\ err_t\ e^{\alpha} + (1 - err_t) e^{-\alpha} \tag{66}$$

$$\frac{\partial}{\partial \alpha} err_t\ e^{\alpha} + (1 - err_t) e^{-\alpha} = err_t\ e^{\alpha} - (1 - err_t) e^{-\alpha} = 0 \tag{67}$$

$$e^{2\alpha} = \frac{1 - err_t}{err_t} \tag{68}$$

$$\alpha = \frac{1}{2} log(\frac{1 - err_t}{err_t}) \tag{69}$$