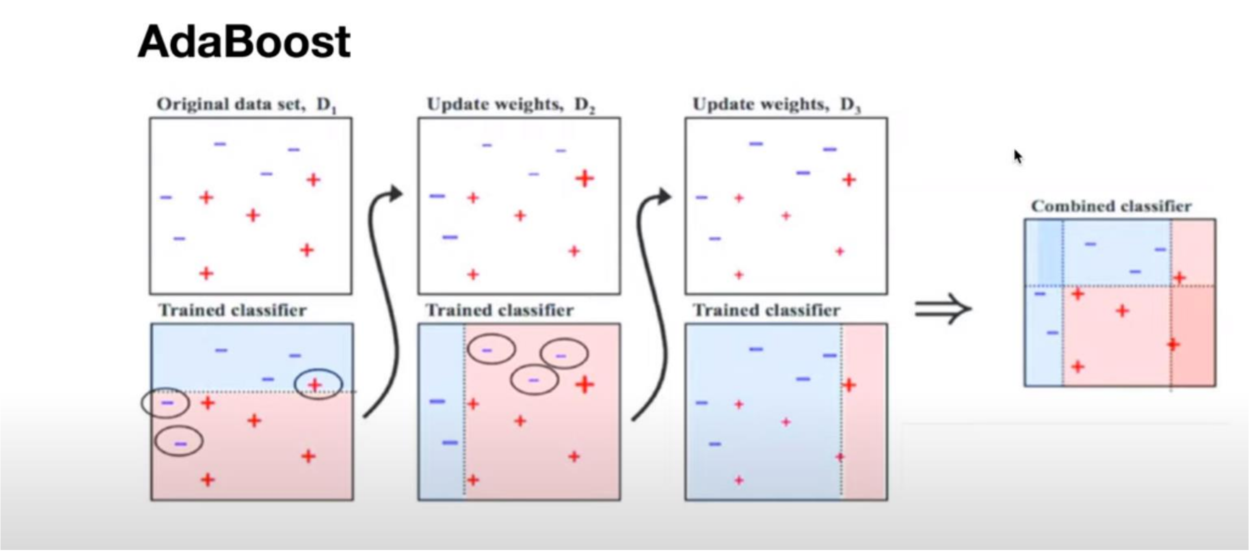
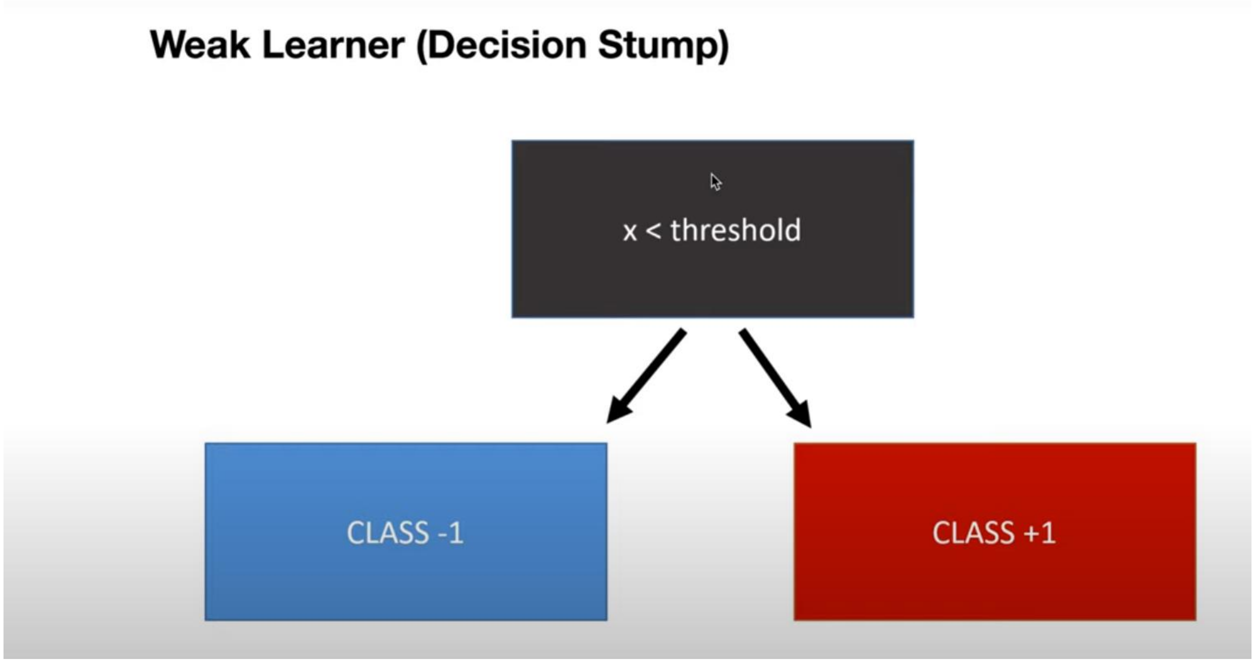
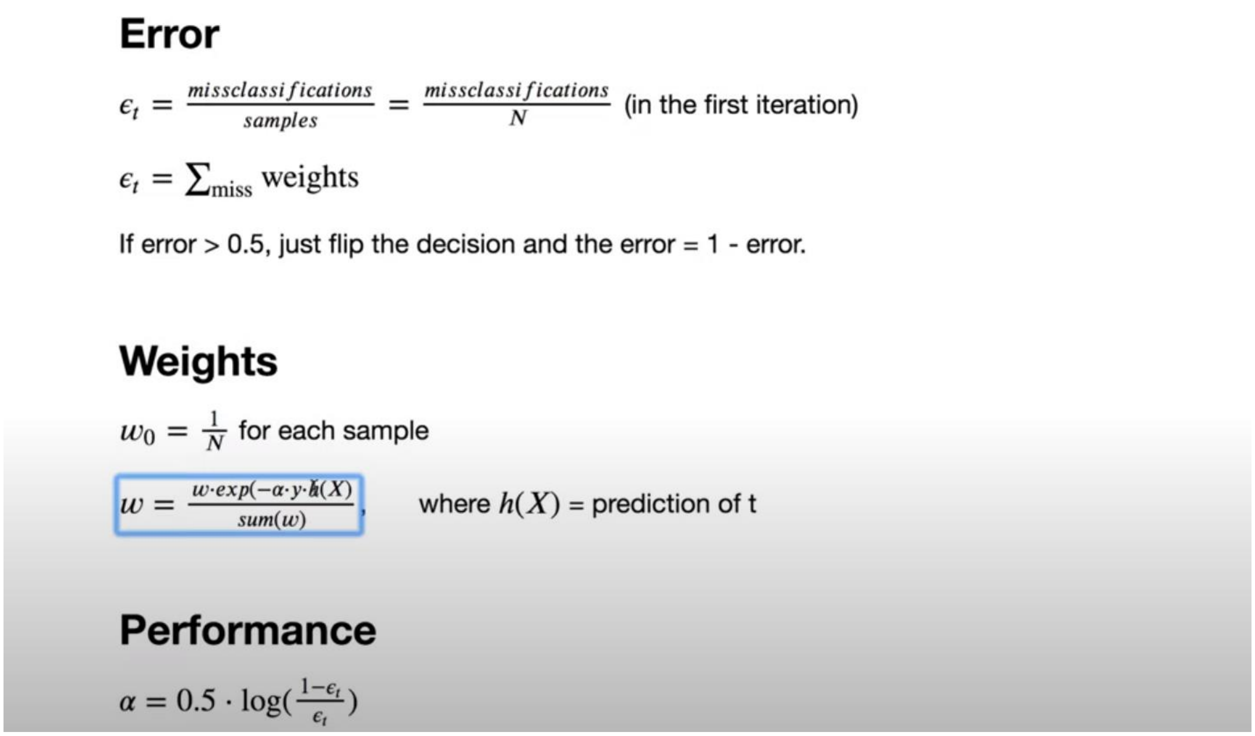


AdaBoost



Weak Learner (Decision Stump)



Error

$\epsilon_t = \frac{missclassifications}{samples} = \frac{missclassifications}{N}$ (in the first iteration)

$\epsilon_t = \sum_{\text{miss}} \text{weights}$

If error > 0.5, just flip the decision and the error = 1 - error.

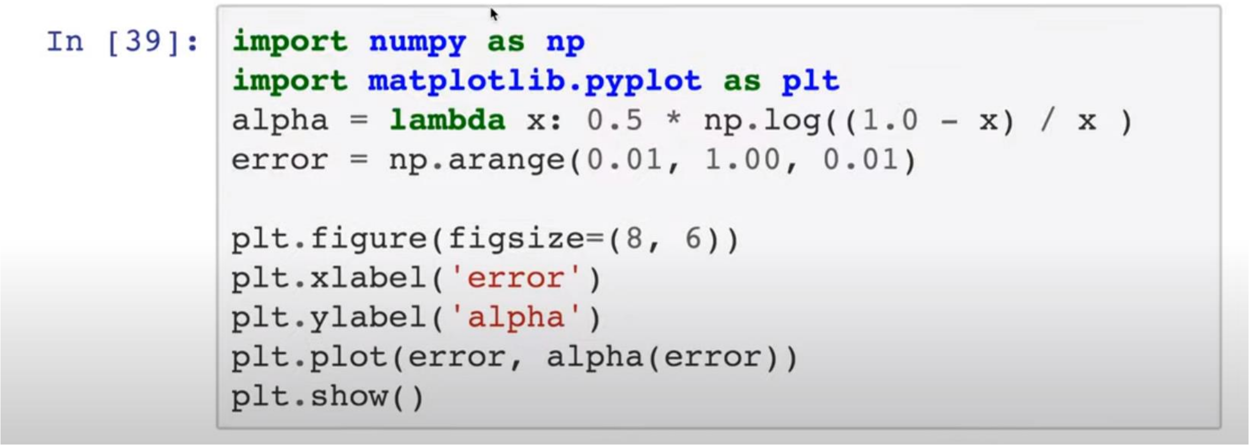
Weights

$w_0 = \frac{1}{N}$ for each sample

$w = \frac{w \cdot exp(-\alpha \cdot y \cdot h(X)}{sum(w)}$, where $h(X)$ = prediction of t

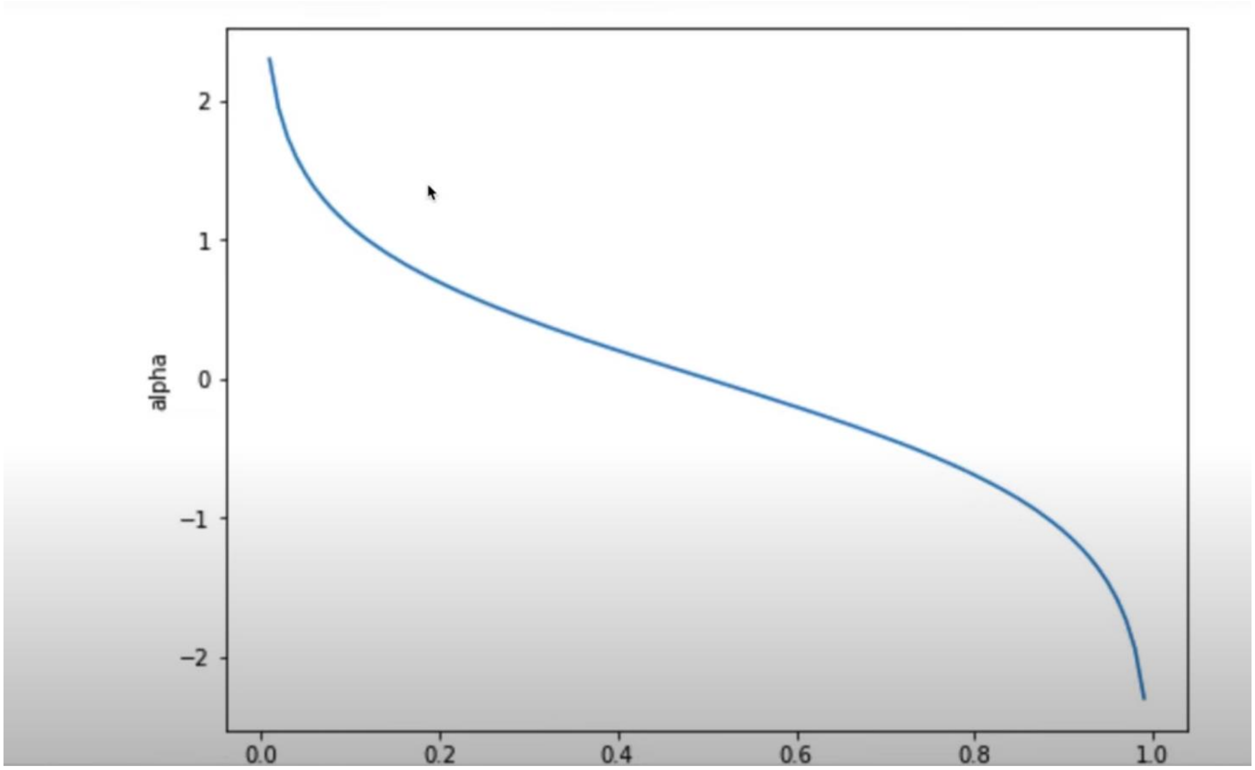
Performance

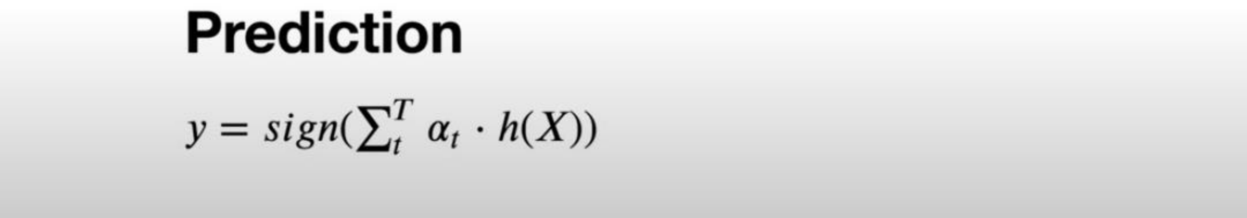
$\alpha = 0.5 \cdot \log(\frac{1-\epsilon_t}{\epsilon_t})$



In [39]:

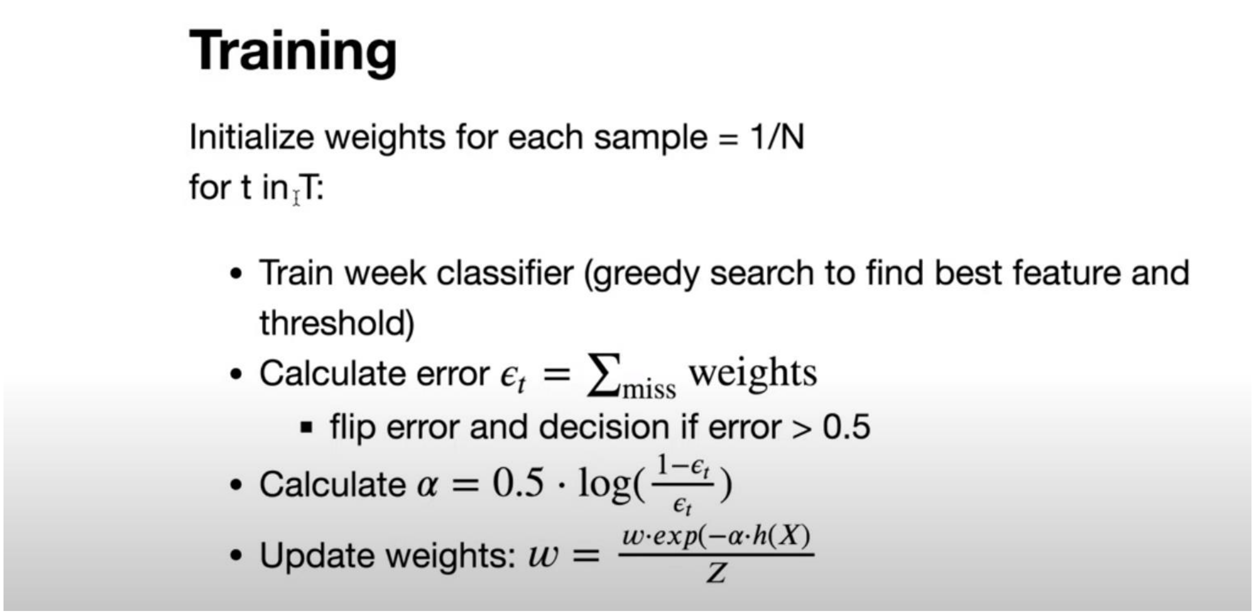
```
import numpy as np
import matplotlib.pyplot as plt
alpha = lambda x: 0.5 * np.log((1.0 - x) / x )
error = np.arange(0.01, 1.00, 0.01)

plt.figure(figsize=(8, 6))
plt.xlabel('error')
plt.ylabel('alpha')
plt.plot(error, alpha(error))
plt.show()
```





Prediction

$y = sign(\sum_t^T \alpha_t \cdot h(X))$



Training

Initialize weights for each sample = 1/N

for t in T:

- Train week classifier (greedy search to find best feature and threshold)
- Calculate error $\epsilon_t = \sum_{\text{miss}} \text{weights}$
  - flip error and decision if error > 0.5
- Calculate $\alpha = 0.5 \cdot \log(\frac{1-\epsilon_t}{\epsilon_t})$
- Update weights: $w = \frac{w \cdot exp(-\alpha \cdot h(X)}{Z}$