ML intro– AdaBoost (bonus)

Link to the recording: https://youtu.be/uOPrF_w4u7U

AdaBoost (1)

- The main idea is to weight the predictions of the classifiers with their error
- Later classifiers focus on examples that were misclassified by earlier ones
- In a binary classification problem (± 1), it will produce a discriminant function:

$$g(x) = \sum_{t=1}^{T} \alpha_t f_t(x)\,, \alpha_t \geq 0$$

Where T is the number of classifiers we have and $f_t(x)$ are our “weak” classifiers

AdaBoost (2)

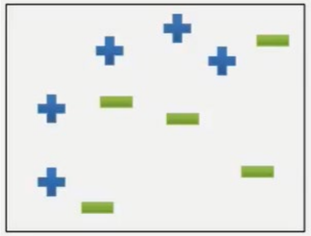
- Iterative algorithm
- Initially, all of the weights (one for each sample) are uniform
- Use some weak learner to classify the data with weights.
- Increase the weight of misclassified samples (and decrease the correct)
  - Therefore, forcing the weak learner to focus on the hard examples
- Relatively simple to implement, as long as we have an implementation of a weak learner with weights.
- Will work as long as the “basic” classifiers are at least better than random guess.
  - Can be applied to boost any ”stronger” classifier

AdaBoost Algorithm (1)

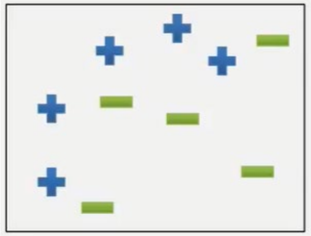
- נאתחל על התפלגות המשקלים על פני N הדגימות באימון, d(x), כך ש $\sum d(x_i) = 1$.
- נגדיר $d_0(x_i) = 1/N$, לכל דגימה מהאימון.
- בכל איטרציה t מתוך T מסווגים, בצע:
  - נמצא את המסווג החלש הכי טוב $f_i(x)$ בעזרת המשקלים $d_i(x)$ (למשל, העץ הכי טוב).
  - נחשב את השגיאה $\varepsilon_t = \sum_{i=1}^{N} d_t(x_i) \cdot \delta\big(y_t \neq f_t(x)\big)$ כאשר δ היא פונקציית 0-1.
  - נגדיר את α_t להיות $\alpha_t = \frac{1}{2}\log\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$ – פונקציה יורדת ממש. נרצה לתת משקל קטן ל ε_t גדול. בגלל שבחרנו מסווג טוב יותר מסתם ניחוש, $\varepsilon_t < 1/2 \Leftarrow \alpha_t > 0$.
  - לכל x_i, נעדכן את ההתפלגות: $d_{t+1}(x_i) = d_t(x_i) \cdot \exp(-\alpha_t y_i f_t(x_i))$ וננרמל את $d_{t+1}(x_i)$ כך שסכומם יהיה 1.
- $f_{FINALE} = sign(\sum_{t=1}^{T} \alpha_t \cdot f_t(x))$.

ln

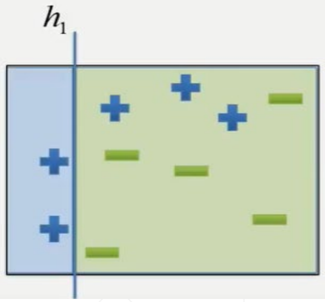
$y_i * f_t(x_i) \in \{-1,1\}$



All points start with equal weights.



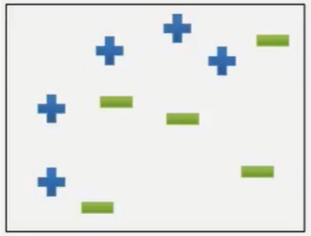
All points start with equal weights.



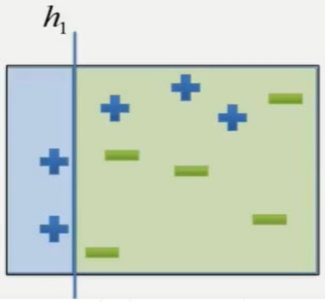
Run the weak learning algorithm to get a weak classifier.

Choose coefficient $\alpha_1 = .41$

Error is e=0.3 (3*1/10)



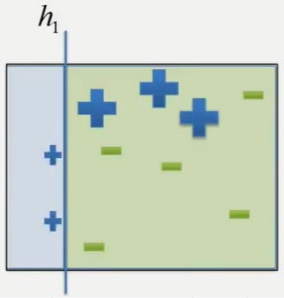
All points start with equal weights.



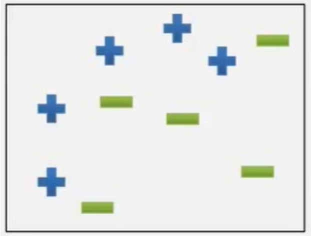
Run the weak learning algorithm to get a weak classifier.

Choose coefficient $\alpha_1 = .41$

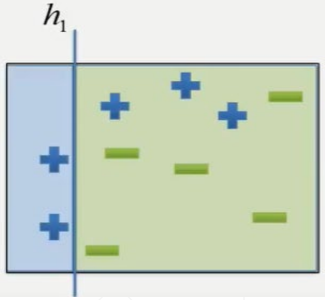
Error is e=0.3 (3*1/10)



Increase the weights on the misclassified points, decrease the weights on the correctly classified points.



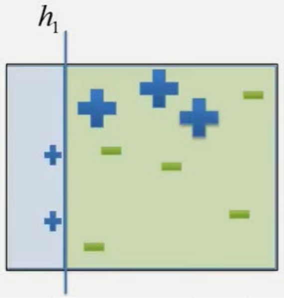
All points start with equal weights.



Run the weak learning algorithm to get a weak classifier.

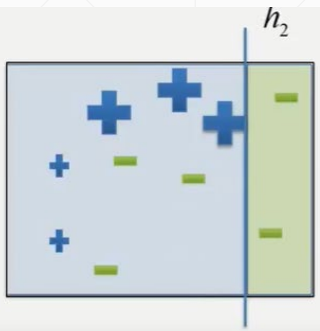
Choose coefficient $\alpha_1 = .41$

Error is e=0.3 (3*1/10)



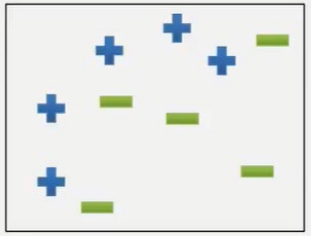
Increase the weights on the misclassified points, decrease the weights on the correctly classified points.

Higher α, because the 3 highly weighted points are classified correctly

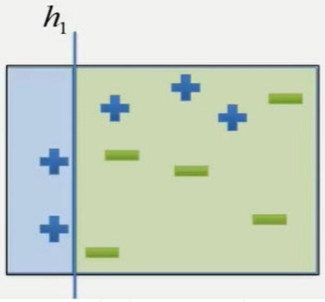


Run the weak learning algorithm to get a weak classifier for the weighted data.

Choose coefficient $\alpha_2 = .66$



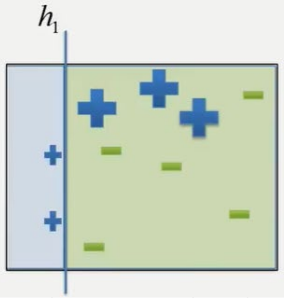
All points start with equal weights.



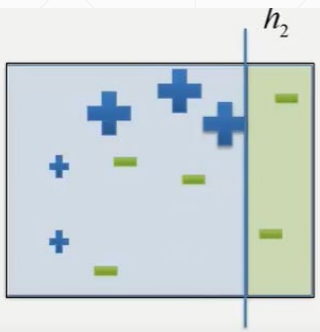
Run the weak learning algorithm to get a weak classifier.

Choose coefficient $\alpha_1 = .41$

Error is e=0.3 (3*1/10)



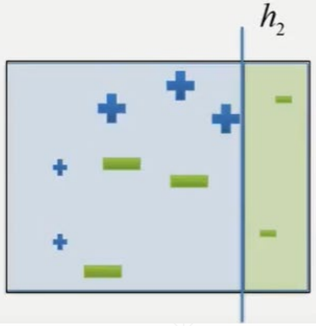
Increase the weights on the misclassified points, decrease the weights on the correctly classified points.



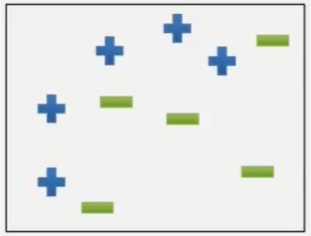
Run the weak learning algorithm to get a weak classifier for the weighted data.

Choose coefficient $\alpha_2 = .66$

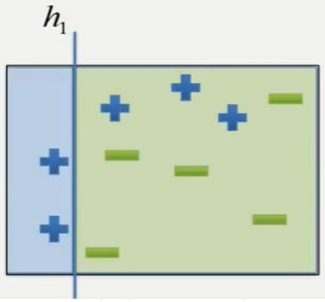
Higher α, because the 3 highly weighted points are classified correctly



Increase the weights on the misclassified points, decrease the weights on the correctly classified points.



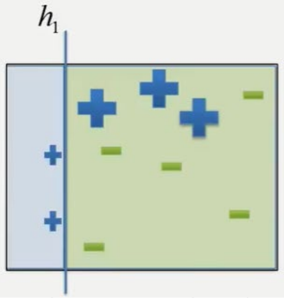
All points start with equal weights.



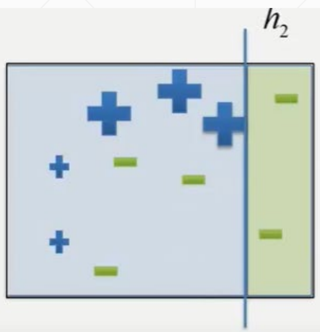
Run the weak learning algorithm to get a weak classifier.

Choose coefficient $\alpha_1 = .41$

Error is e=0.3 (3*1/10)



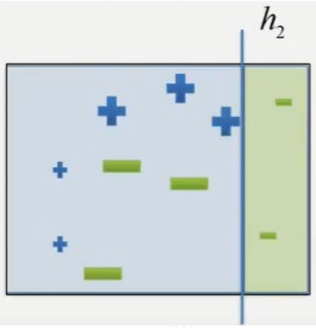
Increase the weights on the misclassified points, decrease the weights on the correctly classified points.



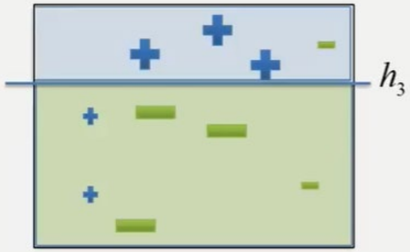
Run the weak learning algorithm to get a weak classifier for the weighted data.

Choose coefficient $\alpha_2 = .66$

Higher α, because the 3 highly weighted points are classified correctly

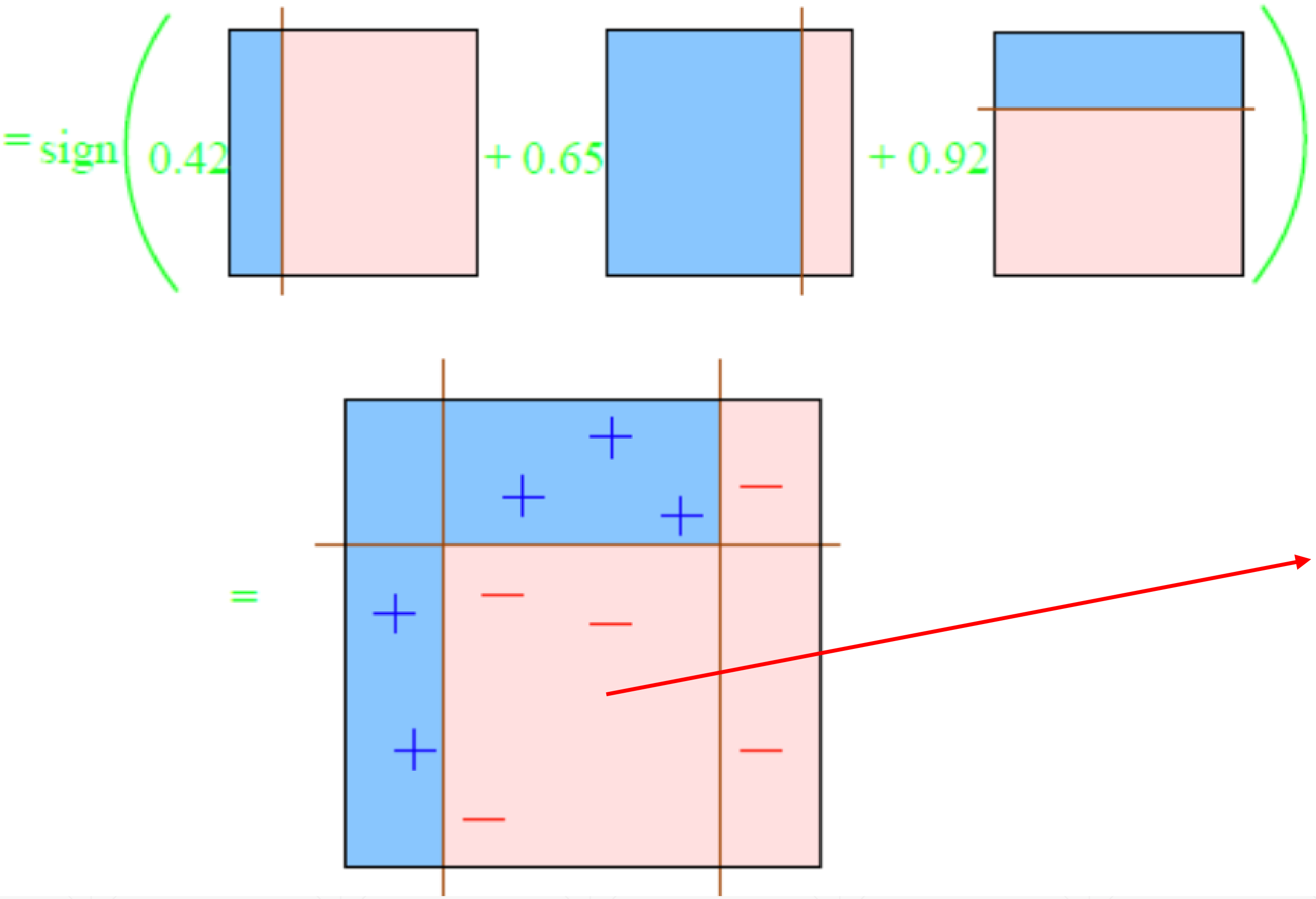


Increase the weights on the misclassified points, decrease the weights on the correctly classified points.



Choose coefficient $\alpha_3 = .93$

Example – end

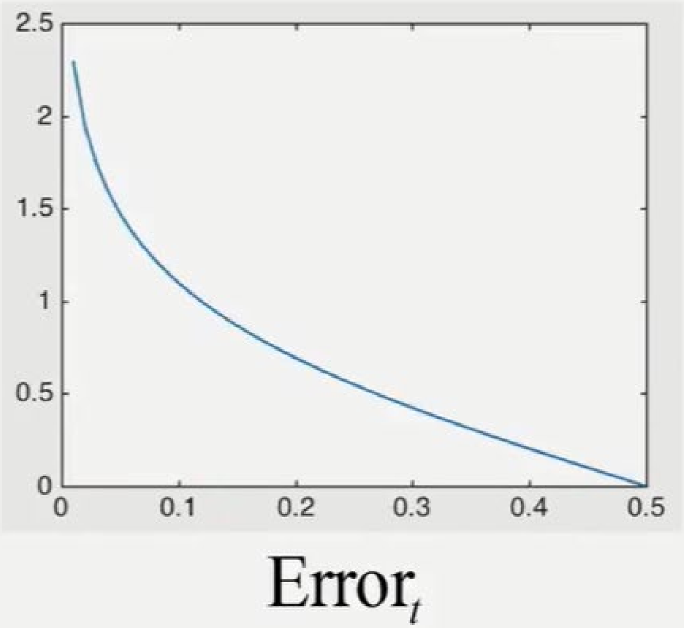


Why red?

Alpha value (1)

$$\text{Error}_t = \sum_{i:h_t(x_i)\neq y_i} d_t = \text{sum of weights of misclassified points}$$

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\text{Error}_t}{\text{Error}_t}\right)$$



Big error -> small alpha → less influence of this classifier on the final classifier result (at the end).

Alpha value (2)

- $$d_{t+1,i} = \frac{d_{t,i}\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

  - If a sample miss-classified, then the weight will be larger since we get d * e^(alpha)
    - Big error → small alpha →
  - If a sample is correctly classified, then the weight will be smaller since we get d * e^(-alpha).
  - Hence, in the next step we focus on the misclassed examples.

AdaBoost Algorithm (2)

Notes:

- This algorithm only works for binary classification
- If our classifiers do not take weighted samples, we can sample from the training data, according to the distribution of $d_t(x)$
- Since each weak classifier is better than random, the error rate ε_t is less than ½
- It can be shown that the training error drops exponentially fast

$$Error_{train} \leq \exp\left(-2\sum_t \left(\varepsilon_t - \frac{1}{2}\right)^2\right)$$

AdaBoost: Advantages

- Relatively fast
- Simple
- Has only one parameter to tune: the number of classifiers T
- Flexible – can be combined with any classifier(s)
- Can find outliers
  - Which are often the hardest examples
- Effective
- Robust for overfitting.
- Test set error decreases even after training error is zero.
- Maximizes distance from margin, as function of T.

epoch	5	100	1000
training error	0.0	0.0	0.0
test error	8.4	3.3	3.1
%margins≤0.5	7.7	0.0	0.0
Minimum margin	0.14	0.52	0.55

AdaBoost: Disadvantages

- Depends on the weak learner
  - If the learner is too weak => it can fail
  - If the learner is too strong/complex => can lead to overfitting
- Empirically, it seems especially susceptible to noise
  - Tries to classify all points and if there is a lot of noise, will fail..

Question 2

True/False:

1. If a weak classifier has a weighted error rate of $\varepsilon \leq \frac{1}{3}$, it can only misclassify up to 1/3 of the training points
2. When you update weights, the training point with the smallest weight in the previous round will always increase in weight.
3. AdaBoost accounts for outliers by lowering the weights of the training points that are repeatedly misclassified.

Question 2: Solution

1. **False**. Note that the error rate is weighted.
2. **False.** It will increase weight only if it is misclassified in the current round.
3. **False**. AdaBoost will increase the weights of training points that are repeatedly misclassified

Question 3

True/False:

Assume we found classifier f_t at stage t, with weight α_t, error ε_t and sample weights distribution d_t.

It is possible that at stage t+1 **the same classifier** f_t will be chosen again.

באיטרציה t מסויימת, נסתכל על $f_t(x)$ ונסתכל על $d_{t+1}(x)$, אז אם היינו מפעילים את $f_t(x)$ על קבוצת האימון במשקלים $d_{t+1}(x)$, היינו מקבלים שהשגיאה היא בדיוק 0.5. לכן, באלגוריתם Ada Boost לא ייצא מצב שנבחר פעמיים ברצף את אותו המסווג.

AdaBoost: Code example

Once again, we'll use the drugs dataset.

```
from sklearn.ensemble import AdaBoostClassifier

clf = AdaBoostClassifier(n_estimators=100)
clf.fit(X_trainset, y_trainset)
clf.score(X_testset, y_testset)
```

```
0.8166666666666667
```

The score method computes the mean accuracy on the given test data and labels

In this case, we have a lower accuracy than the other models

Thank you ☺