# ADABOOST

Shingchern D. You

# Ensemble learning

- Stacking
  - Voting
  - Training a post classifier
- Bagging
  - Well known: Random forest
  - Mentioned before
- Boosting
  - Well known: AdaBoost (Adaptive Boosting)
  - Build a strong classifier from many weak classifiers

# Numerical illustration of voting

- Given the following 1-D example (not linearly separable)

| X = | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| d = | 1 | 1 | 1 | -1 | -1 | -1 | 1 | 1 | 1 | -1 |

- 1st classifier

| X = | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| h1 = | 1 | 1 | 1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |

# Numerical illustration of voting

- 2nd classifier

| X = | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| h2 = | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -1 |

- 3rd classifier

| X = | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| h3 = | -1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 | 1 | 1 |

# Numerical illustration of voting

- Perform majority vote

| X= | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| h1 = | 1 | 1 | 1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| h2 = | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -1 |
| h3 = | -1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 | 1 | 1 |
| H= | 1 | 1 | 1 | -1 | -1 | -1 | 1 | 1 | 1 | -1 |

- All samples are correctly classified

# Numerical illustration of voting

- Although each classifier is a linear weak classifier (i.e., low accuracy), combined classifier is a strong **nonlinear** classifier
- Explain why (**where do we introduce nonlinearity**?)
- AdaBoost follows the same idea, but with weighted sum instead of voting

# AdaBoost algorithm

- Symbol definition
  - Samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in R^p$
  - Desired output $d_1, \ldots, d_n \in \{-1, +1\}$
  - Initial weights $w_{1,1}, \ldots, w_{n,1}$ set to $\frac{1}{n}$ (note: 2nd index is time)
  - Weak classifiers $h: \boldsymbol{x}_k \to \{-1, +1\}$

# AdaBoost algorithm

- For t = 1 … T
  - Find and save weak classifier $h_t(\boldsymbol{x})$ minimize $\epsilon_t = \sum_{k=1}^{n} w_{k,t} \ell(h_t(\boldsymbol{x}_k) \neq d_k)$ (Note: $\epsilon_t$ sometimes could be very small)
  - Update $\alpha_t \leftarrow \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
  - Update weights: $w_{k,t+1} \leftarrow w_{k,t} \exp(-\alpha_t h_t(\boldsymbol{x}_k) d_k)$
  - For k = 1 …n : $H(\boldsymbol{x}_k) = \operatorname{sign}((\sum_{z=1}^{t} \alpha_z h_z(\boldsymbol{x}_k))$
  - Stop condition: (1) No error on classifying training data

Or (2) Upper limit of iterations reached

# AdaBoost algorithm

- Two classifiers in use
  - Use current weak classifier $h_t(\boldsymbol{x}_k)$ to update weights
    $$w_{k,t+1} \leftarrow w_{k,t} \exp(-\alpha_t h_t(\boldsymbol{x}_k) d_k)$$
  - Use combined strong classifier $H(\boldsymbol{x}_k)$ to check error samples (but **cannot** be used for weights updating)
    $$H(\boldsymbol{x}_k) = \mathrm{sign}\left(\sum_{z=1}^{t} \alpha_z h_z(\boldsymbol{x}_k)\right) == d_k?$$

# AdaBoost algorithm

- For classification after training, use

$$H(\boldsymbol{x}) = \text{sign}(\left(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})\right)$$

- There are several variations on AdaBoost
- The version given here is from **Machine Learning in Action (a good book for engineers)**
- You can compare this algorithm with the one in textbook (original AdaBoost.M1)

# AdaBoost

- AdaBoost has solid theories behind it, but not to be mentioned here
- Some key points in algorithm
  - Weak classifier (should not be too strong)
  - Best decision for weighted error in weak classifier
  - Numerical issues (could be bad)
  - Very sensitive to noise and outliers

# Weak classifier

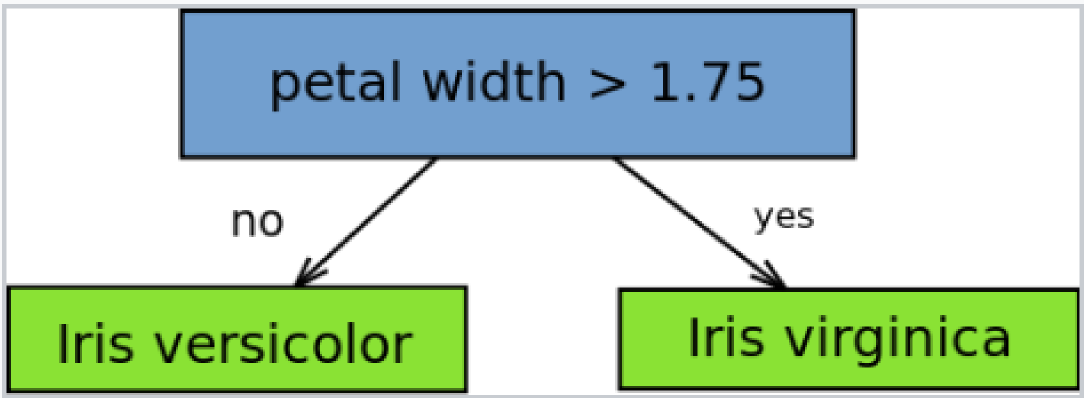
- In the algorithm, we need to search over all possible combinations of parameters to find optimal weighted error

$$\epsilon_t = \sum_{k=1}^{n} w_{k,t} \ell(h_t(\boldsymbol{x}_k) \neq y_k)$$

- Not easy with many classifiers (such as SVM)
- One widely used classifier is **decision stump**: making a decision on one feature only

# Weak classifier

- Decision stump example (from wiki)




An example of a decision stump that discriminates between two of three classes of Iris flower data set: *Iris versicolor* and *Iris virginica*. The petal width is in centimetres. This particular stump achieves 94%

# Weak classifier

- To find $\epsilon_t$, we need to check all possible threshold values for all features
- Consider the following small example:
  - P1 = (1, 2.1), C = +1
  - P2 = (2, 1.1), C = +1
  - P3 = (1.3 1) ,C = -1
  - P4 = (1, 1), C = -1
  - P5 = (2, 1), C = +1

# Weak classifier

- For 1st feature, we need to check (for example)

Threshold = {0.9, 1.1, 1.4, 2.1}  (other values OK, too)

- For 2nd feature, we need to check

Threshold = { 0.9, 1.05, 1.2, 2.2}

- We also need to know if $h(\boldsymbol{x}_k) > 0$ means C=1 or C=-1
- Finally, pick the threshold with lowest $\epsilon_t$

# Weak classifier

- For example, we set thr = 1.4 in 1st feature: if 1st feature > thr, C =1, else C = -1
- We have only one error in 1st iteration (t = 1)
- Therefore, $\epsilon_1 = 0.2, \alpha_1 = 0.6931,$

$$\boldsymbol{w}_{.,1} = [0.5, 0.125, 0.125, 0.125, 0.125]^T$$

- We can do more steps with same approach

# XOR experiment

- Use 100 samples in XOR as training samples:
- If (feature 1) * (feature 2) > 0

  then C = 1, else C = -1
- Feature 1 and 2 are random numbers
- No error in training set at around 400 iterations (i.e., 400 weak classifiers)

# Using AdaBoost

- **Keep in mind: AdaBoost is very sensitive to noise and outliers (i.e., training samples with wrong classification)**
- Need to use weak classifiers for best performance
- Theories show that AdaBoost also widens the "margin" as SVM does