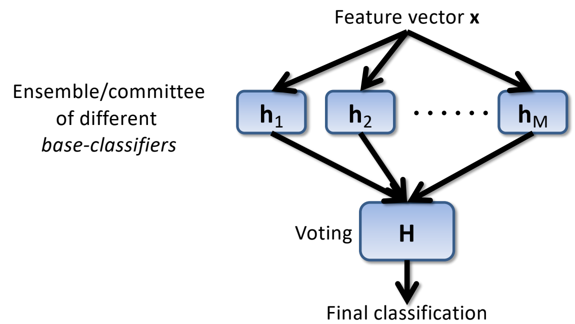
Neural Networks and Learning Systems
TBMI26 / 732A55
2019

Lecture 4

Ensemble Learning

Magnus Borga
magnus.borga@liu.se

Classifier ensemble

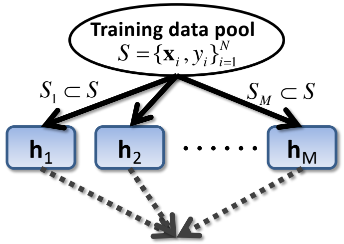


2

Bootstrap **Agg**regat**ing** (Bagging)

Breiman,1994

Train each base-classifier using a subset of the training data



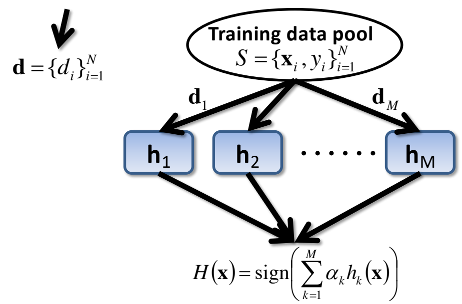
Reduced variance – reduced risk of overfitting

3

Boosting

Schapire and others, (1989-1990)

Train each base-classifier using all training data but with weights indicating how important each training sample is.



4

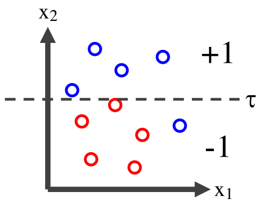
Simple/Weak classifiers

cf. "a rule of thumb"

$$\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix}$$

Example: Threshold ***one*** feature

$$h(x_2) = \begin{cases} +1 & x_2 \geq \tau \\ -1 & x_2 < \tau \end{cases}$$



5

Decision stump

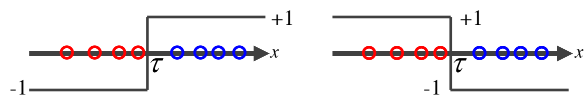
Polarity $p = \{-1,1\}$

$p = 1$

$$h(x) = \begin{cases} +1 & x \geq \tau \\ -1 & x < \tau \end{cases}$$

$p = -1$

$$h(x) = \begin{cases} +1 & x \leq \tau \\ -1 & x > \tau \end{cases}$$

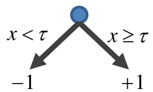


6

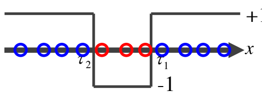
Classification and Regression Trees (CART)

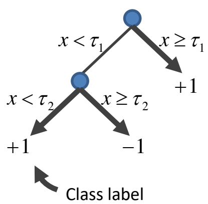
$$h(x) = \begin{cases} +1 & x \geq \tau \\ -1 & x < \tau \end{cases}$$

Decision stump



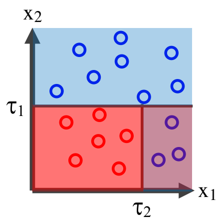
$$h(x) = \begin{cases} +1 & x \geq \tau_1 \\ -1 & x < \tau_1 \text{ and } x \geq \tau_2 \\ +1 & x < \tau_2 \end{cases}$$



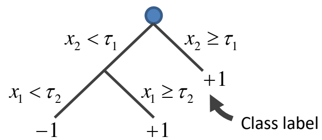


7

CART – 2D example



CART



Piecewise flat classification function

$$f(\mathbf{x}; \underbrace{w_1, \ldots, w_k}_{\text{Feature index and thresholds}}) \rightarrow \Omega$$

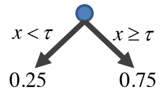
8

Regression Tree

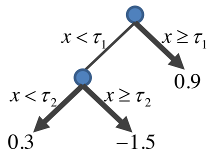
Same, but with real-valued output!

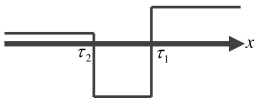
Decision stump

$$h(x)=\begin{cases}0.75 & x\ge\tau\\ 0.25 & x<\tau\end{cases}$$



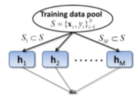
$$h(x)=\begin{cases}0.9 & x\ge\tau_1\\ -1.5 & x<\tau_1 \text{ and } x\ge\tau_2\\ 0.3 & x<\tau_2\end{cases}$$



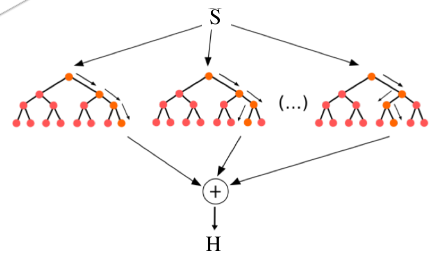


9

Random Forest



Bagging + Decision Trees



1. For each tree, select a random subset of the training samples
2. For each split, select a random subset of the features

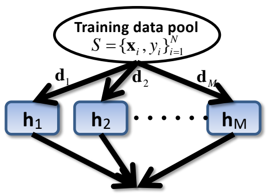
} Make the trees "independent"

Image from https://towardsdatascience.com/random-forest-learning-essential-understanding-1ca856a963cb

10

General boosting algorithm

Train weak classifiers sequentially!



$$H(\mathbf{x})=\operatorname{sign}\left(\sum_{k=1}^{M}\alpha_k h_k(\mathbf{x})\right)$$

1. Set weights d_1=1/N
2. Train weak classifier $h_1(\mathbf{x})$ using weights $\mathbf{d}_1$
3. Increase and decrease weight for wrongly and correctly classified training examples respectively -> $\mathbf{d}_2$
4. Train weak classifier $h_2(\mathbf{x})$ using weights $\mathbf{d}_2$
5. Repeat until $h_M(\mathbf{x})$

11

Training a decision stump

Find best split threshold τ!

Class label {-1,+1}

Training input: $\{x_i, y_i, d_i\}_{i=1}^M$ Normalized weights: $\sum_{i=1}^{M} d_i = 1$

Consider only one feature

Weight

Threshold function: $h(x;\tau,p)=\begin{cases}+1 & px\ge p\tau\\ -1 & px<p\tau\end{cases}$

Polarity {-1,1}

0-1 cost function: $\min_{\tau,p}\varepsilon(\tau,p)=\sum_{i=1}^{M} d_i\, I\big(y_i\ne h(x_i;\tau,p)\big)$

1 for false classifications

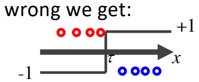
12

Training a decision stump, cont.

$$\min_{\tau,p} \varepsilon(\tau,p) = \sum_{i=1}^{M} d_i \, I(y_i \neq h(x_i;\tau,p)) \text{ is always} \leq 0.5!$$

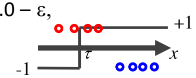
Why? If we classify all training samples wrong we get:

$$\varepsilon(\tau,p) = \sum_{i=1}^{M} d_i = 1$$



But we can then just change polarity/sign and get all samples correct, i.e., ε = 0!

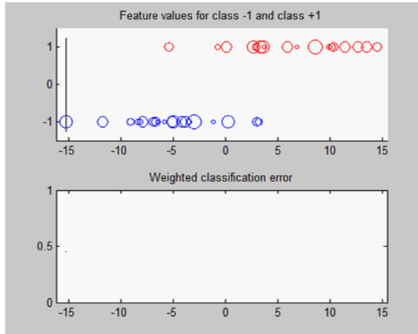
In general, if we obtain an error ε between 0.5 and 1.0, we can switch polarity and get the error 1.0 – ε, which is smaller than 0.5.



Brute force optimization

$$\min_{\tau,p} \varepsilon(\tau,p) = \sum_{i=1}^{N} d_i \, I(y_i \neq h(x_i;\tau,p))$$

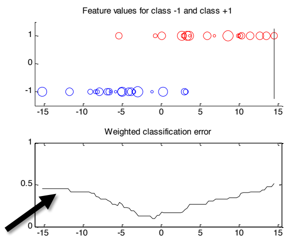
Movie!



Brute force optimization

$$\min_{\tau,p} \varepsilon(\tau,p) = \sum_{i=1}^{M} d_i \, I(y_i \neq h(x_i;\tau,p))$$

Cost-function jumps at the x_i:s. Enough to test all thresolds in the set $\tau \in \{x_i\}_{i=1}^{N}$ and see which one gives the smallest error.



Cost function is piece-wise constant!

Brute force optimization

Training samples $\mathbf{x}_i = \begin{pmatrix} x_{1,i} \\ \vdots \\ x_{N,i} \end{pmatrix}, i = 1 \cdots M$

training samples

feature components

Pseudo code:

ε_{min} = inf
for all feature components k = 1:N
for all thresholds $\tau \in \{x_{k,i}\}_{i=1}^{M}$
$\varepsilon(\tau, p=1) = \sum_{i=1}^{M} I(y_i \neq h(x_{k,i};\tau,p=1))$
if $\varepsilon > 0.5$
p = -1
$\varepsilon = 1 - \varepsilon$
end
if $\varepsilon < \varepsilon_{min}$ ...... **end**
end
end

Discrete AdaBoost

Freund & Schapire, 1995

Training data: $\{\mathbf{x}_i, y_i\}_{i=1}^{M}$, $y_i \in \{-1,+1\}$

Initialization: $d_1(i) = \frac{1}{M}$, $T = \#$ base classifiers

for t = 1 to T

Find weak classifier $h_t(\mathbf{x}) = \{-1,+1\}$ that minimizes the weighted classification error:

$$\varepsilon_t = \sum_{i=1}^{M} d_t(i) I(y_i \neq h_t(\mathbf{x}_i))$$

Update weights:

$$d_{t+1}(i) = d_t(i) e^{-\alpha_t y_i h_t(x_i)}, \text{ where } \alpha_t = \frac{1}{2}\ln\frac{1-\varepsilon_t}{\varepsilon_t}$$

and renormalize so that $\sum_{i=1}^{M} d_{t+1}(i) = 1$

end

Final strong classifier: $H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t \text{h}_t(\mathbf{x})\right)$

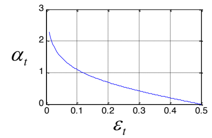
17

Discrete AdaBoost

- <u>Discrete</u> output from the weak classifier $h_t(\mathbf{x}) = \{-1,+1\}$
- Weight update

$$d_{t+1}(i) = d_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)} = \begin{cases} d_t(i) e^{-\alpha_t} & \text{If } \mathbf{x}_i \text{ correctly classified} \\ d_t(i) e^{\alpha_t} & \text{If } \mathbf{x}_i \text{ wrongly classified} \end{cases}$$

- Performance of weak classifier: $\alpha_t = \frac{1}{2}\ln\frac{1-\varepsilon_t}{\varepsilon_t}$

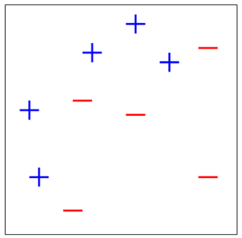


Final strong classifier: $H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t \text{h}_t(\mathbf{x})\right)$

18

Discrete AdaBoost – Toy example

Initial weights $d_1(i) = \frac{1}{10}$, $\text{T} = 3$



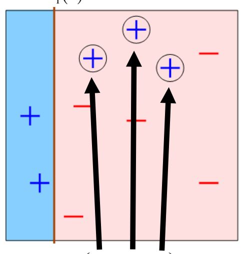
10 training samples $\mathbf{x}_i$, $\text{i} = 1 \cdots 10$

Images borrowed from R. Schapire, "A Boosting Tutorial"

19

Discrete AdaBoost – Toy example

$h_1(\mathbf{x})$

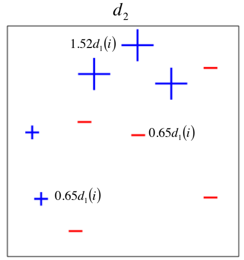


$d_1(i) = \frac{1}{10}$

$$\varepsilon_1 = \left(\frac{1}{10} + \frac{1}{10} + \frac{1}{10}\right) = 0.3$$

$$\alpha_1 = \frac{1}{2}\ln\left(\frac{1-0.3}{0.3}\right) = 0.42$$

d_2

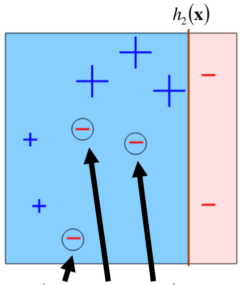


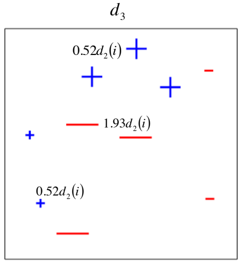
$$d_2(i) = \begin{cases} d_1(i) e^{-0.42} = 0.65 d_1(i) & \text{If } \mathbf{x}_i \text{ correctly classified} \\ d_1(i) e^{0.42} = 1.52 d_1(i) & \text{If } \mathbf{x}_i \text{ wrongly classified} \end{cases}$$

Normalize d_2!

20

Discrete AdaBoost – Toy example





$$\varepsilon_2 = \left(d_2(j) + d_2(k) + d_2(l)\right) = 0.21$$

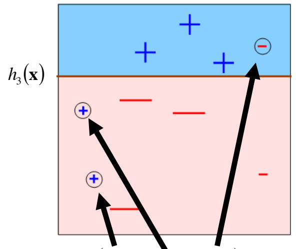
$$\alpha_2 = \frac{1}{2}\ln\left(\frac{1-0.21}{0.21}\right) = 0.66$$

$$d_3(i) = \begin{cases} d_2(i)e^{-0.66} = 0.52d_2(i) & \text{If } \mathbf{x}_i \text{ correctly classified} \\ d_2(i)e^{0.66} = 1.93d_2(i) & \text{If } \mathbf{x}_i \text{ wrongly classified} \end{cases}$$

Normalize d_3!

21

Discrete AdaBoost – Toy example



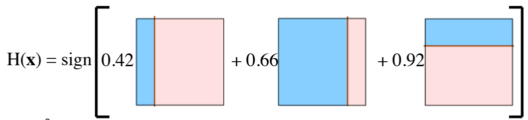
$$\varepsilon_3 = \left(d_3(p) + d_3(q) + d_3(r)\right) = 0.14$$

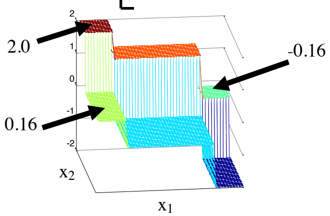
$$\alpha_3 = \frac{1}{2}\ln\left(\frac{1-0.14}{0.14}\right) = 0.92$$

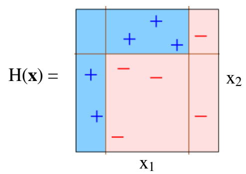
22

Discrete AdaBoost – Toy example

Final strong classifier: $H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$

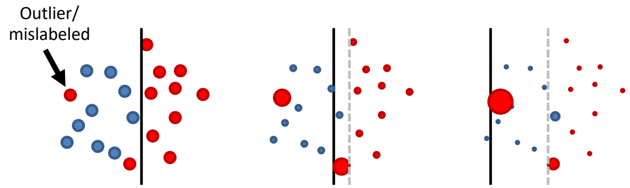






23

Problem - Outliers



- Outliers gain weight exponentially
- Will eventually result in bad weak classifiers
- May ruin the final ensemble classifier

24

Outlier strategies

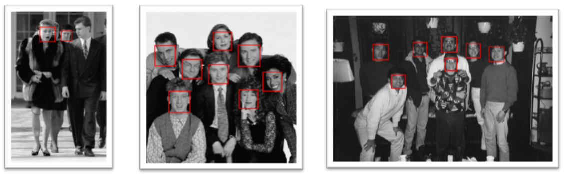
- Keep an eye on the weights (plot them!)
- Weight trimming
  – Don’t allow weights larger than a certain threshold
  – Disregard training samples with too large weights
- Use alternative weight update schemes with less aggressive increases for misclassified training data
  – LogitBoost
  – GentleBoost

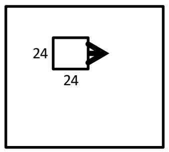
Summary Ensemble Learning
(using decision trees)

- Nonlinear classifiers that are easy to implement
- Easy to use - just one or a few parameters (T)
- Inherent feature selection
- Slow to train – fast to classify (real-time)
- Look out for outlier problems (AdaBoost)

Real-time object detection

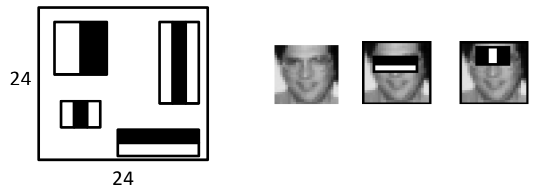
P. Viola and M. Jones “*Rapid Object Detection using a Boosted Cascade of Simple Features*”, 2001





Sweep a sub-window over the image.
for each position, determine if the sub-window contains a face or not.

Haar-features
Rectangle filters



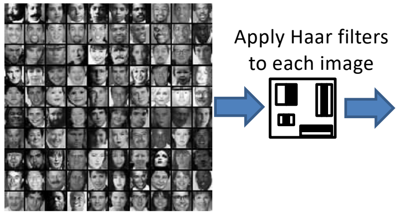
From the sub-window, calculate contrast features (Haar features) at different locations and scales, and in different orientations.
LOTS of different combinations, can be several 100,000 features!

Train detector with AdaBoost

As described previously!

24 x 24 pixels face images

Apply Haar filters to each image



$$\mathbf{x}_i = \begin{pmatrix} x_{1,i} \\ \vdots \\ x_{N,i} \end{pmatrix}, i = 1 \ldots M$$

Similar with non-face images to obtain negative examples.

29