COMS 4721: Machine Learning for Data Science
Lecture 13, 3/10/2015

Prof. John Paisley

Columbia University
**Algorithm: Bagging binary classifiers**

Given \((x_1, y_1), \ldots, (x_n, y_n), x \in \mathcal{X}, y \in \{-1, +1\}\)

- For \(b = 1, \ldots, B\)
  - Sample a bootstrap dataset \(\mathcal{B}_b\) of size \(n\) from \(D_0 = \sum_{i=1}^{n} \frac{1}{n} \delta_{x_i}\).
  - Learn a classifier \(f_b\) using data in \(\mathcal{B}_b\).

- Set the classification rule to be

\[
    f_{bag}(x_0) = \text{sign}\left(\sum_{b=1}^{B} f_b(x_0)\right).
\]

- With bagging, we saw how a *committee* of classifiers votes on a label.
- Each classifier is learned on a *bootstrap sample* from the data set.
- Learning a collection of classifiers is referred to as an *ensemble method*.
Boosting is another powerful method for ensemble learning. It is similar to bagging in that a set of classifiers are combined to make a better one.

Free to choose a classifier, but a “weak” one is usually picked – i.e., one with accuracy a little better than random guessing, but very fast to learn.

**Short history**

1984 : Leslie Valiant and Michael Kearns ask if “boosting” is possible.
1990 : Yoav Freund creates an optimal boosting algorithm.
Bagging vs Boosting (schematic)

Bagging

- Bootstrap sample $\rightarrow f_1(x)$
- Bootstrap sample $\rightarrow f_2(x)$
- Bootstrap sample $\rightarrow f_3(x)$

Training sample

Boosting

- Weighted sample $\rightarrow f_1(x)$
- Weighted sample $\rightarrow f_2(x)$
- Weighted sample $\rightarrow f_3(x)$

Training sample
Algorithm: Boosting a binary classifier

Given \((x_1, y_1), \ldots, (x_n, y_n), x \in \mathcal{X}, y \in \{-1, +1\}\), set \(w_1(i) = \frac{1}{n}\).

- For \(t = 1, \ldots, T\)

  1. Sample a bootstrap dataset \(B_t\) of size \(n\) from \(D_t = \sum_{i=1}^{n} w_t(i) \delta_{x_i}\).
  2. Learn a classifier \(f_t\) using data in \(B_t\).
  3. Set \(\epsilon_t = \sum_{i=1}^{n} w_t(i) \mathbb{1}\{y_i \neq f_t(x_i)\}\) and \(\alpha_t = \frac{1}{2} \ln \left( \frac{1 - \epsilon_t}{\epsilon_t} \right)\).
  4. Update \(\tilde{w}_{t+1}(i) = w_t(i) \exp\{-\alpha_t y_i f_t(x_i)\}\).
  5. Normalize \(w_{t+1}(i) = \frac{\tilde{w}_{t+1}(i)}{\sum_j \tilde{w}_{t+1}(j)}\).

- Set the classification rule to be

\[
    f_{boost}(x_0) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t f_t(x_0) \right).
\]

**Comment:** Step 1 can often be skipped and \(w_t\) used directly in Step 2, which is the way AdaBoost is normally presented.
### The AdaBoost Algorithm (Schematic)

**Training sample**

Weighted sample → Classify

B₁ ~ D₁ → {a₁, f₁(x)}

Classify

B₂ ~ D₂ → {a₂, f₂(x)}

Classify

B₃ ~ D₃ → {a₃, f₃(x)}

### Weighted sample

Boosting

\[
f_{\text{boost}}(x_0) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t f_t(x_0) \right)
\]
Boosting a decision stump (example 1)

Original data

Uniform data distribution
Learn weak classifier

Here: Use a decision stump

\[ x_1 > 1.7 \]

\[ \hat{y} = 1 \]

\[ \hat{y} = 3 \]
Boosting a Decision Stump (Example 1)

Round 1 classifier

Weighted error: $\epsilon_1 = 0.3$

Weight update: $\alpha_1 = 0.42$
Boosting a decision stump (example 1)

Weighted data

After round 1
Boosting a Decision Stump (Example 1)

Round 2 classifier

Weighted error: $\epsilon_2 = 0.21$
Weight update: $\alpha_2 = 0.65$
Boosting a Decision Stump (Example 1)

Weighted data

After round 2
Boosting a Decision Stump (Example 1)

Round 2 classifier

Weighted error: $\epsilon_3 = 0.14$
Weight update: $\alpha_3 = 0.92$
Boosting a Decision Stump (Example 1)

Classifier after three rounds:

- 0.42 x +
- 0.65 x +
- 0.92 x +
Boosting a Decision Stump (Example 2)

A Toy Problem

Random guessing
50% error

Decision stump
45.8% error

Full decision tree
24.7% error

Boosted stump
5.8% error
Point = one dataset. Location = error rate w/ and w/o boosting. The boosted version of the same classifier almost always produces better results.
(left) Boosting a bad classifier is often better than not boosting a good one.  
(right) Boosting a good classifier is often better (but can take more time).
**Boosting and feature maps**

**Q:** What makes boosting work so well?

**A:** This is a very well studied question. We will present one analysis later, but we can also give intuition by tying it in with what we’ve discussed.

The classification for a new $x_0$ from boosting is

$$f_{boost}(x_0) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t f_t(x_0) \right).$$

Define $\phi(x) = [f_1(x), \ldots, f_T(x)]^T$, where each $f_t(x) \in \{-1, +1\}$.

- We can think of $\phi(x)$ as a high dimensional feature map of $x$.
- The vector $\alpha = [\alpha_1, \ldots, \alpha_T]^T$ correspond to hyperplane.
- So the classifier can be written $f_{boost}(x_0) = \text{sign}(\phi(x_0)^T \alpha)$.
- Boosting learns the feature mapping and hyperplane simultaneously.
APPLICATION: FACE DETECTION
**Face Detection** (Viola & Jones, 2001)

**Problem**: Locate the faces in an image or video.

**Processing**: Divide image into patches of different scales, e.g., $24 \times 24$, $48 \times 48$, etc. Extract features from each patch.

**Classify** each patch as face or no face using a *boosted decision stump*. This can be done in real-time, for example by your digital camera (at 15 fps).

- One patch from a larger image. Mask it with many “feature extractors.”
- Each pattern gives one number, which is the sum of all pixels in black region minus sum of pixels in white region (total of 45,000+ features).

\begin{figure}
  \centering
  \includegraphics[width=\textwidth]{example.png}
  \caption{The first and second features selected by AdaBoost. The two features are shown in the top row and then overlayed on a typical training face in the bottom row. The first feature measures the difference in intensity between the region of the eyes and a region across the upper cheeks. The feature capitalizes on the observation that the eye region is often darker than the cheeks. The second feature compares the intensities in the eye regions to the intensity across the bridge of the nose.}
  \end{figure}
6. Conclusions

We have presented an approach for face detection which minimizes computation time while achieving high detection accuracy. The approach was used to construct a face detection system which is approximately 15 times faster than any previous approach. Preliminary experiments, which will be described elsewhere, show that highly efficient detectors for other objects, such as pedestrians or automobiles, can also be constructed in this way.

This paper brings together new algorithms, representations, and insights which are quite generic and may well have broader application in computer vision and image processing.

The first contribution is a new technique for computing a rich set of image features using the integral image. In order to achieve true scale invariance, almost all face detection systems must operate on multiple image scales. The integral image, by eliminating the need to compute a multi-scale image pyramid, reduces the initial image processing required for face detection.
ANALYSIS OF BOOSTING
Training error theorem

We can use analysis to make a statement about the predictive accuracy of boosting on the training data.

**Theorem:** Under the AdaBoost framework, if $\epsilon_t$ is the weighted error of classifier $f_t$, then for the classifier $f_{boost}(x_0) = \text{sign}(\sum_{t=1}^{T} \alpha_t f_t(x_0))$, 

$$\text{training error} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\{y_i \neq f_{boost}(x_i)\} \leq \exp\left(-2 \sum_{t=1}^{T} \left(\frac{1}{2} - \epsilon_t\right)^2\right).$$

Even if each $\epsilon_t$ is only a little better than random guessing, the accumulation of them over $T$ classifiers can lead to a substantial value in the exponent.

For example:

$\epsilon_t = 0.45, \ T = 1000 \rightarrow \text{training error} \leq 0.0067.$
PROOF OF THEOREM

Setup

We break the proof into three steps. It is an application of the fact that

\[
\text{if } a < b \quad \text{and} \quad b < c \quad \text{then} \quad a < c
\]

Step 1 allows us to know what \( b \) is above.

Steps 2 and 3 correspond to the two inequalities.

Also recall the following step from AdaBoost:

\[
\text{Update } \tilde{w}_{t+1}(i) = w_t(i)e^{-\alpha_t y_i f_t(x_i)} \quad \text{and normalize } \tilde{w}_{t+1}(i) = \frac{\tilde{w}_{t+1}(i)}{\sum_j \tilde{w}_{t+1}(j)}.
\]

We define \( Z_t = \sum_j \tilde{w}_{t+1}(j) \) for use in the proof.
**Proof of Theorem**

**Step 1**

We first want to expand the equation of the weights to show that

$$w_{T+1}(i) = \frac{1}{n} \exp\{-y_i \sum_{t=1}^{T} \alpha_t f_t(x_i)\} \times \frac{\exp\{-\alpha_1 y_i x_i\}}{Z_1} \times \cdots \times \frac{\exp\{-\alpha_T y_i x_i\}}{Z_T}. $$

**Derivation of Step 1:**

To do so, first notice the recurrence: $w_{t+1}(i) = w_t(i) e^{-\alpha_t y_i x_i} / Z_t$.

We can break down $w_t(i)$ in exactly the same way, and continue until $w_1(i)$,

$$w_{T+1}(i) = w_1(i) \frac{\exp\{-\alpha_1 y_i x_i\}}{Z_1} \times \cdots \times \frac{\exp\{-\alpha_T y_i x_i\}}{Z_T} = \frac{1}{n} \frac{\exp\{-y_i \sum_{t=1}^{T} \alpha_t f_t(x_i)\}}{\prod_{t=1}^{T} Z_t}.$$
**Step 2**

We next need to show that the training error of the classifier after $T + 1$ steps is not greater than $\prod_{t=1}^{T} Z_t$.

**Derivation of Step 2:**

From Step 1: $w_{T+1}(i) = \frac{1}{n} \exp\{-y_i f_{boost}(x_i)\} \prod_{t=1}^{T} Z_t \rightarrow w_{T+1}(i) \prod_{t=1}^{T} Z_t = \frac{1}{n} e^{-y_i f_{boost}(x_i)}$

\[
\frac{1}{n} \sum_{i=1}^{n} 1\{y_i \neq f_{boost}(x_i)\} \leq \frac{1}{n} \sum_{i=1}^{n} \exp\{-y_i f_{boost}(x_i)\} = \sum_{i=1}^{n} w_{T+1}(i) \prod_{t=1}^{T} Z_t = \prod_{t=1}^{T} Z_t
\]
**Proof of Theorem**

**Step 3**

The final step is to calculate an upper bound on $Z_t$, and therefore of $\prod_{t=1}^{T} Z_t$.

Since $\prod_{t=1}^{T} Z_t$ is an upper bound on the training error, the upper bound from Step 3 is also of the training error.

**Derivation of Step 3:**

This step is slightly more involved. It also shows why $\alpha_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$.

$$Z_t = \sum_{i=1}^{n} w_t(i) \exp \{-\alpha_t y_i f_t(x_i)\}$$

$$= \sum_{i : y_i = f_t(x_i)} e^{-\alpha_t} w_t(i) + \sum_{i : y_i \neq f_t(x_i)} e^{\alpha_t} w_t(i)$$

$$= e^{-\alpha_t} \left( 1 - \epsilon_t \right) + e^{\alpha_t} \epsilon_t$$
Proof of Theorem

Derivation of Step 3 (continued):

We’re currently at $Z_t = e^{-\alpha_t}(1 - \epsilon_t) + e^{\alpha_t}\epsilon_t$. Remember from Step 2 that

$$\text{training error} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\{y_i \neq f_{\text{boost}}(x_i)\} \leq \prod_{t=1}^{T} Z_t.$$  

We want the training error to be small. We therefore pick $\alpha_t$ to minimize $Z_t$. This minimum is independent for each $t$ and occurs at

$$\alpha_t = \frac{1}{2} \ln \left( \frac{1 - \epsilon_t}{\epsilon_t} \right).$$

Plugging this value in gives $Z_t = 2\sqrt{\epsilon_t(1 - \epsilon_t)}$. 
**Proof of Theorem**

**Derivation of Step 3 (conclusion):**

Thus \( Z_t = 2\sqrt{\epsilon_t(1 - \epsilon_t)} \)

We re-write this as

\[
Z_t = \sqrt{1 - 4\left(\frac{1}{2} - \epsilon_t\right)^2}.
\]

We use the general inequality \( 1 - x \leq e^{-x} \) to conclude that

\[
Z_t = \left(1 - 4\left(\frac{1}{2} - \epsilon_t\right)^2\right)^{\frac{1}{2}} \leq \left(e^{-4\left(\frac{1}{2} - \epsilon_t\right)^2}\right)^{\frac{1}{2}} = e^{-2\left(\frac{1}{2} - \epsilon_t\right)^2}.
\]
Putting it all together

Step 3 showed that $Z_t \leq e^{-2(\frac{1}{2} - \epsilon_t)^2}$. Because both sides are positive, the product over $t$ doesn’t change this inequality,

$$\prod_{t=1}^{T} Z_t \leq \prod_{t=1}^{T} e^{-2(\frac{1}{2} - \epsilon_t)^2} = e^{-2 \sum_{t=1}^{T} \left(\frac{1}{2} - \epsilon_t\right)^2}.$$ 

From the earlier steps we showed $\prod_{t=1}^{T} Z_t$ was also an upper bound,

training error \(= \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\{y_i \neq f_{boost}(x_i)\} \leq \prod_{t=1}^{T} Z_t \leq e^{-2 \sum_{t=1}^{T} \left(\frac{1}{2} - \epsilon_t\right)^2}.

The two ends of this chain is what we set out to prove.
Q: Driving the training error to zero leads one to ask, does boosting overfit?
A: Sometimes, but very often it doesn’t!