Ensemble Learning

CS4780/5780 – Machine Learning
Fall 2013
Igor Labutov
Cornell University

Ensemble Learning

A class of “meta” learning algorithms
Combining multiple classifiers to increase performance
Very effective in practice
Good theoretical guarantees
Easy to implement!

Ensemble

Problem: given T binary classification hypotheses \((h_1, \ldots, h_T)\), find a combined classifier:

\[ h_S(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right) \]

with better performance.

Teaser

BAGGING
Bagging

Bagging (Bootstrap aggregating), (Breiman, 1996)

\[ h_S(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right) \]

**Bagging**: Special case where we fix:

\[ \alpha_t = 1 \quad \text{and} \quad h_t = L(S_t) \]

\( L \) is some learning algorithm
\( S_t \) is a training set drawn from distribution \( P(x, y) \)

Bias-Variance Tradeoff

Generalization Error

**Classification**:

\[ \epsilon_{test} = \frac{1}{n} \sum_{i} \text{Zero-One-Loss}(y_i, h(x_i)) \]

**Regression**:

\[ \epsilon_{test} = \frac{1}{n} \sum_{i} (y_i - h(x_i))^2 \]
For the entire test set:

**CLAIM:**

\[
\bar{\epsilon}_{test}(x_i) = \frac{1}{T} \sum_{t=1}^{T} (y_i - h_t(x_i))^2
\]

OR, as an expectation:

\[
E_S [(y_i - h_S(x_i))^2]
\]

For the entire test set:

\[
E_{x,y}E_S [(y_i - h_S(x_i))^2]
\]

**Example**

( kNN )

**Democrat vs Republican party association**
CLAIM:
\[ \mathbb{E}_S [(y_i - h_S(x_i))^2] = \]

\[ \text{bias}^2 \qquad (y_i - \mathbb{E}_S[h_S(x_i)])^2 + \]

\[ \text{variance} \quad + \mathbb{E}_S[(h_s(x_i) - \mathbb{E}_S[(h_s(x_i))])^2] \]

USEFUL LEMMA:
\[ \mathbb{E}[(\alpha - \mathbb{E}[\alpha])^2] = \mathbb{E}[\alpha^2] + \mathbb{E}[\alpha]^2 \]

\[ y_i = f(x_i) + \mathcal{N}(0, \sigma^2) \]

\[ \mathbb{E}_S [(y_i - h_S(x_i))^2] = \]

\[ \text{bias}^2 \qquad (y_i - \mathbb{E}_S[h_S(x_i)])^2 + \]

\[ \text{variance} \quad + \mathbb{E}_S[(h_s(x_i) - \mathbb{E}_S[(h_s(x_i))])^2] \]

\[ \text{noise} \quad + \sigma^2 \]
BAGGING
revisited

Bagging

Bagging (Bootstrap aggregating).

Bagging(S = \{(x_1, y_1), \ldots, (x_n, y_n)\})
1 for t = 1 to T do
2 \text{S}_t = \text{BOOTSTRAP}(S) \text{ i.i.d. sampling with replacement from } S.
3 \text{h}_t = \text{TRAIN-CLASSIFIER}(\text{S}_t)
4 return \text{h}_y = x \rightarrow \text{MAJORITY VOTE}(\text{h}_1(x), \ldots, \text{h}_T(x))

Why does it work?

Bagging

Bagging Ensemble:

\[ h_S(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right) \]

What happens to bias and variance?

Bagging Ensemble (regression):

\[ h_S(x) = \frac{1}{T} \sum_{t=1}^{T} h_t(x) \]

What happens to bias and variance?

\[ \text{Bias}(h_S, x_i) = \frac{1}{T} \sum_{t=1}^{T} \text{Bias}(h_t, x_i) \]

\[ \text{Var}(h_S, x_i) \approx \frac{1}{T} \text{Var}(h_1, x_i) \]

Bagging has approximately the same bias, but reduces variance of individual classifiers!
Bagging as a "Training set manipulator"

Bagging

Bag as a "Training set manipulator"

Bagging as a "Training set manipulator"

Bag as a "Training set manipulator"

Bagging as a "Training set manipulator"
**Bagging as a “Training set manipulator”**

**WHAT IF I TOLD YOU**

**YOU CAN CHANGE THESE NUMBERS**

**Ensemble**

Problem: Given $T$ binary classification hypotheses ($h_1, \ldots, h_T$), find a combined classifier:

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

with better performance.

**Hypothetical Algorithm**

Given $x_i \in X, y_i \in Y = \{-1, 1\}$ where $(x_1, y_1), \ldots, (x_n, y_n)$

Initialize $W_1(i) = 1/n$

Initialize set $H = \{h_1, \ldots, h_T\}$

For $t = 1, \ldots, T$:
- Pick hypothesis $h_t$ of the set $H$
- Compute error rate of $h_t$
- Assign new weights $W_0 h_t X$
- Compute new weight of $h_t$

Output $h_S(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$
Hypothetical Algorithm

Given $x_i \in X, y_i \in Y = \{-1, 1\}$ where $(x_1, y_1), \ldots, (x_n, y_n)$
Initialize $W_1(i) = 1/n$
Learning algorithm $L$
For $t = 1, \ldots, T$:
  - Generate hypothesis $h_t^{\text{with}}$
  - Compute error rate $e_t^{q_f}$
  - Assign new weights $W_t \propto X$
  - Compute new weight $h_t^{\text{for}}$
Output $h_S(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$

Hypothetical Algorithm

Given $x_i \in X, y_i \in Y = \{-1, 1\}$ where $(x_1, y_1), \ldots, (x_n, y_n)$
Initialize $W_1(i) = 1/n$
Initialize set $H = \{h_1, \ldots, h_T\}$
For $t = 1, \ldots, T$:
  - Pick hypothesis $h_t^{\text{out}}$ of the set $H$
  - Compute error rate $e_t^{q_f}$
  - Assign new weights $W_t \propto X$
  - Compute new weight $h_t^{\text{for}}$
Output $h_S(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$

Toy Example

- Positive examples
- Negative examples
- 2-Dimensional plane
- Weak hyps: linear separators
- 3 iterations

X > 4?
Questions

• Which hypothesis do we choose at every iteration?
• How should we weight the hypotheses?
• How should we weight the examples?

Answers

Choose \( h_t \) that maximizes \( W_{\text{correct}} \)

Choose \( \alpha_t \) according to:

\[
\alpha_t = \frac{1}{2} \log \left( \frac{1 - \epsilon_t}{\epsilon_t} \right)
\]

Update the weight of instance \( i \) as follows:

\[
w_t(i) = w_{t-1}(i) * e^{-\alpha_t} \quad \text{if} \quad y_i = h_t(x_i)
\]

\[
w_t(i) = w_{t-1}(i) * e^{\alpha_t} \quad \text{if} \quad y_i \neq h_t(x_i)
\]
AdaBoost

Hypothetical Algorithm

Given $x_i \in X, y_i \in Y = \{-1, 1\}$
where $(x_1, y_1), \ldots, (x_n, y_n)$
Initialize $W_1(i) = 1/n$
Initialize set $H = \{h_1, \ldots, h_T\}$
For $t = 1, \ldots, T$:
  • Pick hypothesis $h_t$ of the set $H$
  • Compute error rate of $h_t$
  • Assign new weights $X$
  • Compute new weight $\alpha_t$
Output $h_\delta(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$

Training Error for AdaBoost

Write for some weighted error $\mathcal{E}$:

$\epsilon_t = \frac{1}{2} - \gamma_t$

We can then bound the training error:

Training error $\leq \exp(-2T\gamma^2)$

For some $\gamma$ such that:

$\gamma_t \geq \gamma > 0$

What about Generalization Error?
Why?

Margin

\[
\text{margin}_f(x, y) = \frac{yf(x)}{\sum \alpha_i} = \frac{y \sum \alpha_i h(x)}{\sum \alpha_i}
\]

Margins

Viola Jones Classifier
Image Features

“Rectangle filters”

\[
\text{Value} = \sum \text{(pixels in white area)} - \sum \text{(pixels in black area)}
\]

Fast computation with integral images

- The integral image computes a value at each pixel \((x,y)\) that is the sum of the pixel values above and to the left of \((x,y)\), inclusive.
- This can quickly be computed in one pass through the image.

Computing sum within a rectangle

- Let \(A, B, C, D\) be the values of the integral image at the corners of a rectangle.
- Then the sum of original image values within the rectangle can be computed as:
  \[
  \text{sum} = A - B - C + D
  \]
- Only 3 additions are required for any size of rectangle!
  - This is now used in many areas of computer vision.

Example

- Integral Image
  \[
  \begin{array}{c}
  \text{Integral Image} \\
  \hline
  -1 & \text{(x,y)} \leftarrow \text{+1} \\
  +2 & \text{+1} \\
  -1 & \text{-2} \\
  \end{array}
  \]
“Rectangle filters”

Similar to Haar wavelets

Papageorgiou, et al.

\[ h_i(x_i) = \begin{cases} \alpha_i & \text{if } f_i(x_i) > \varrho_i \\ \beta_i & \text{otherwise} \end{cases} \]

\[ C(x) = \delta \left( \sum h_i(x_i) + b \right) \]

60,000 features to choose from