Ensemble
Introduction

• We are almost at the end of the semester/final competition.
  • https://inclass.kaggle.com/c/ml2016-cyber-security-attack-defender/leaderboard
  • https://www.kaggle.com/c/outbrain-click-prediction/leaderboard
  • https://www.kaggle.com/c/transfer-learning-on-stack-exchange-tags/leaderboard

• You already developed some algorithms and codes. Lazy to modify them.

• Ensemble: improving your machine with little modification
Framework of Ensemble

• Get a set of classifiers
  • $f_1(x), f_2(x), f_3(x), \ldots$
  
    低  补  DD  They should be diverse.

• Aggregate the classifiers (*properly*)
  • 在打王時每個人都有該站的位置
Ensemble: Bagging
Review: Bias v.s. Variance

Large Bias
Small Variance

Overfitting
Underfitting

Large Variance
Small Bias

Error from bias
Error from variance
Error observed
A complex model will have large variance.

We can average complex models to reduce variance.

If we average all the $f^*$, is it close to $\hat{f}$

$$E[f^*] = \hat{f}$$
Bagging

N training examples

Sampling N’ examples with replacement (usually N=N’)

Set 1

Set 2

Set 3

Set 4

Function 1

Function 2

Function 3

Function 4
Bagging

This approach would be helpful when your model is complex, easy to overfit. E.g. decision tree
Decision Tree

Assume each object x is represented by a 2-dim vector $[x_1, x_2]$

Can have more complex questions

The questions in training:
- Number of branches,
- Branching criteria,
- Termination criteria,
- Base hypothesis
Experiment: Function of Miku

http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/theano/miku

(1st column: x, 2nd column: y, 3rd column: output (1 or 0))
Experiment: Function of Miku

Single Decision Tree

Depth = 5
Depth = 10
Depth = 15
Depth = 20
Random Forest

• Decision tree:
  • Easy to achieve 0% error rate on training data
  • If each training example has its own leaf ......

• Random forest: Bagging of decision tree
  • Resampling training data is not sufficient
  • Randomly restrict the features/questions used in each split

• Out-of-of-bag validation for bagging
  • Using RF = f_2+f_4 to test x^1
  • Using RF = f_2+f_3 to test x^2
  • Using RF = f_1+f_4 to test x^3
  • Using RF = f_1+f_3 to test x^4

Out-of-bag (OOB) error

Good error estimation of testing set

<table>
<thead>
<tr>
<th>train</th>
<th>f_1</th>
<th>f_2</th>
<th>f_3</th>
<th>f_4</th>
</tr>
</thead>
<tbody>
<tr>
<td>x^1</td>
<td>O</td>
<td>X</td>
<td>O</td>
<td>X</td>
</tr>
<tr>
<td>x^2</td>
<td>O</td>
<td>X</td>
<td>X</td>
<td>O</td>
</tr>
<tr>
<td>x^3</td>
<td>X</td>
<td>O</td>
<td>O</td>
<td>X</td>
</tr>
<tr>
<td>x^4</td>
<td>X</td>
<td>O</td>
<td>X</td>
<td>O</td>
</tr>
</tbody>
</table>
**Experiment:**

**Function of Miku**

Random Forest

(100 trees)
Ensemble: Boosting

Improving Weak Classifiers
Boosting

- Guarantee:
  - If your ML algorithm can produce classifier with error rate smaller than 50% on training data
  - You can obtain 0% error rate classifier after boosting.

- Framework of boosting
  - Obtain the first classifier $f_1(x)$
  - Find another function $f_2(x)$ to help $f_1(x)$
    - However, if $f_2(x)$ is similar to $f_1(x)$, it will not help a lot.
    - We want $f_2(x)$ to be complementary with $f_1(x)$ (How?)
  - Obtain the second classifier $f_2(x)$
  - ...... Finally, combining all the classifiers

- The classifiers are learned sequentially.

Training data: 
\{(x^1, \hat{y}^1), ..., (x^n, \hat{y}^n), ..., (x^N, \hat{y}^N)\}
\hat{y} = \pm 1 \text{ (binary classification)}
How to obtain different classifiers?

• Training on different training data sets
• How to have different training data sets
  • Re-sampling your training data to form a new set
  • Re-weighting your training data to form a new set
  • In real implementation, you only have to change the cost/objective function

\[
L(f) = \sum_n l(f(x^n), \hat{y}^n)
\]

\[
L(f) = \sum_n u^n l(f(x^n), \hat{y}^n)
\]
Idea of Adaboost

- Idea: **training** $f_2(x)$ **on the new training set that fails** $f_1(x)$
- How to find a new training set that fails $f_1(x)$?

$\varepsilon_1$: the error rate of $f_1(x)$ on its training data

\[
\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_1}
\]

$Z_1 = \sum_n u_1^n \quad \varepsilon_1 < 0.5$

Changing the example weights from $u_1^n$ to $u_2^n$ such that

\[
\frac{\sum_n u_2^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_2} = 0.5
\]

The performance of $f_1$ for new weights would be random.

Training $f_2(x)$ based on the new weights $u_2^n$
Re-weighting Training Data

• Idea: training $f_2(x)$ on the new training set that fails $f_1(x)$
• How to find a new training set that fails $f_1(x)$?

$(x^1, \hat{y}^1, u^1) \quad u^1 = 1$ ✔

$(x^2, \hat{y}^2, u^2) \quad u^2 = 1$ ✗

$(x^3, \hat{y}^3, u^3) \quad u^3 = 1$ ✔

$(x^4, \hat{y}^4, u^4) \quad u^4 = 1$ ✔

$\varepsilon_1 = 0.25$

$0.5$

$f_1(x)$

$\varepsilon_2 < 0.5$

$f_2(x)$

$u^1 = 1/\sqrt{3}$ ✔

$u^2 = \sqrt{3}$ ✗

$u^3 = 1/\sqrt{3}$ ✔

$u^4 = 1/\sqrt{3}$ ✔
Re-weighting Training Data

• Idea: training $f_2(x)$ on the new training set that fails $f_1(x)$
• How to find a new training set that fails $f_1(x)$?

\[
\begin{cases}
\text{If } x^n \text{ misclassified by } f_1 \ (f_1(x^n) \neq \hat{y}^n) & \Rightarrow u_2^n \leftarrow u_1^n \text{ multiplying } d_1 \text{ increase} \\
\text{If } x^n \text{ correctly classified by } f_1 \ (f_1(x^n) = \hat{y}^n) & \Rightarrow u_2^n \leftarrow u_1^n \text{ devided by } d_1 \text{ decrease}
\end{cases}
\]

$f_2$ will be learned based on example weights $u_2^n$

What is the value of $d_1$?
Re-weighting Training Data

\[
\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_1}
\]

\[
Z_1 = \sum_n u_1^n
\]

\[
\sum_n u_2^n \delta(f_1(x^n) \neq \hat{y}^n) = 0.5
\]

\[
f_1(x^n) \neq \hat{y}^n \quad u_2^n \leftarrow u_1^n \text{ multiplying } d_1
\]

\[
f_1(x^n) = \hat{y}^n \quad u_2^n \leftarrow u_1^n \text{ devided by } d_1
\]

\[
= \sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1 = \sum_{f_1(x^n) \neq \hat{y}^n} u_2^n + \sum_{f_1(x^n) = \hat{y}^n} u_2^n
\]

\[
= \sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1 + \sum_{f_1(x^n) = \hat{y}^n} u_1^n / d_1
\]

\[
\sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1 + \sum_{f_1(x^n) = \hat{y}^n} u_1^n / d_1
\]

\[
\sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1
\]

\[
= 2
\]
Re-weighting Training Data

\[ \varepsilon_1 = \frac{\sum u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_1} \]

\[ Z_1 = \sum u_1^n \]

\[ \sum u_2^n \delta(f_1(x^n) \neq \hat{y}^n) \]

\[ \frac{Z_2}{Z_2} = 0.5 \]

\[ f_1(x^n) \neq \hat{y}^n \quad u_2^n \leftarrow u_1^n \text{ multiplying } d_1 \]

\[ f_1(x^n) = \hat{y}^n \quad u_2^n \leftarrow u_1^n \text{ divided by } d_1 \]

\[ \frac{\sum f_1(x^n) \neq \hat{y}^n u_1^n d_1 + \sum f_1(x^n) = \hat{y}^n u_1^n / d_1}{\sum f_1(x^n) \neq \hat{y}^n u_1^n d_1} = 2 \]

\[ \frac{\sum f_1(x^n) = \hat{y}^n u_1^n / d_1}{\sum f_1(x^n) \neq \hat{y}^n u_1^n d_1} = 1 \]

\[ \sum f_1(x^n) = \hat{y}^n \quad \sum f_1(x^n) \neq \hat{y}^n \]

\[ \frac{1}{d_1} \sum u_1^n d_1 \quad \sum u_1^n = d_1 \quad \sum u_1^n = \sum f_1(x^n) = \hat{y}^n \frac{u_1^n}{d_1} = \sum f_1(x^n) \neq \hat{y}^n \]

\[ \varepsilon_1 = \frac{\sum f_1(x^n) \neq \hat{y}^n u_1^n}{Z_1} \]

\[ \sum u_1^n = Z_1 \varepsilon_1 \]

\[ Z_1(1 - \varepsilon_1) \]

\[ Z_1 \varepsilon_1 \]

\[ Z_1(1 - \varepsilon_1)/d_1 = Z_1 \varepsilon_1 d_1 \]

\[ d_1 = \sqrt{(1 - \varepsilon_1)/\varepsilon_1} > 1 \]
Algorithm for AdaBoost

• Giving training data
  \{ (x^1, \hat{y}^1, u_1^1), \ldots, (x^n, \hat{y}^n, u_1^n), \ldots, (x^N, \hat{y}^N, u_1^N) \} 
  \hat{y} = \pm 1 \text{ (Binary classification), } u_1^1 = 1 \text{ (equal weights) }

• For \( t = 1, \ldots, T \):
  \begin{itemize}
    \item Training weak classifier \( f_t(x) \) with weights \( \{u_t^1, \ldots, u_t^N\} \)
    \item \( \varepsilon_t \) is the error rate of \( f_t(x) \) with weights \( \{u_t^1, \ldots, u_t^N\} \)
  \end{itemize}

• For \( n = 1, \ldots, N \):
  \begin{itemize}
    \item If \( x^n \) is misclassified
      \begin{align*}
        \hat{y}_t^n = f_t(x^n)
        u_{t+1}^n &= u_t^n \times d_t = u_t^n \times \exp(\alpha_t) \\
        d_t &= \sqrt{\frac{1 - \varepsilon_t}{\varepsilon_t}} \\
        \alpha_t &= \ln \sqrt{\frac{1 - \varepsilon_t}{\varepsilon_t}}
      \end{align*}
    \item Else:
      \begin{align*}
        u_{t+1}^n &= \frac{u_t^n}{d_t} = u_t^n \times \exp(-\alpha_t) \\
        \alpha_t &= \ln \sqrt{\frac{1 - \varepsilon_t}{\varepsilon_t}}
      \end{align*}
  \end{itemize}

\[
\begin{align*}
  u_{t+1}^n &\leftarrow u_t^n \times \exp(-\alpha_t) \\
\end{align*}
\]
Algorithm for AdaBoost

• We obtain a set of functions: \( f_1(x), \ldots, f_t(x), \ldots, f_T(x) \)

• How to aggregate them?
  • Uniform weight:
    • \( H(x) = sign(\sum_{t=1}^{T} f_t(x)) \)
  • Non-uniform weight:
    • \( H(x) = sign(\sum_{t=1}^{T} \alpha_t f_t(x)) \)

\[
\alpha_t = \ln \sqrt{\frac{1 - \varepsilon_t}{\varepsilon_t}} \quad \varepsilon^t = 0.1 \quad \varepsilon^t = 0.4
\]
\[
u_{t+1}^n = u_t^n \times \exp\left(-\hat{y}^n f_t(x^n)\alpha_t\right) \quad \alpha^t = 1.10 \quad \alpha^t = 0.20
\]
Toy Example

T=3, weak classifier = decision stump

• t=1

\[ f_1(x) \]

\[ \varepsilon_1 = 0.30 \]
\[ d_1 = 1.53 \]
\[ \alpha_1 = 0.42 \]
Toy Example

\( f_1(x): \)

- \( t=2 \)
  - \( \alpha_1 = 0.42 \)

\( f_2(x) \)

- \( \varepsilon_2 = 0.21 \)
- \( d_2 = 1.94 \)
- \( \alpha_2 = 0.66 \)

T=3, weak classifier = decision stump

\( \varepsilon = 0.21 \)
\( d = 1.94 \)
\( \alpha = 0.66 \)
Toy Example

T=3, weak classifier = decision stump

- $t=3$
  - $f_1(x)$: $\alpha_1 = 0.42$
  - $f_2(x)$: $\alpha_2 = 0.66$
  - $f_3(x)$: $\alpha_3 = 0.95$

$\epsilon_3 = 0.13$
$d_3 = 2.59$
$\alpha_3 = 0.95$
Toy Example

- Final Classifier: \( H(x) = \text{sign}(\sum_{t=1}^{T} \alpha_t f_t(x)) \)

\[
\text{sign}( 0.42 + 0.66 + 0.95 )
\]
Warning of Math

\[ H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t f_t(x) \right) \quad \alpha_t = \ln \sqrt{\frac{1 - \varepsilon_t}{\varepsilon_t}} \]

As we have more and more \( f_t \) (T increases), \( H(x) \) achieves smaller and smaller error rate on training data.
Error Rate of Final Classifier

- Final classifier: $H(x) = \text{sign}(\sum_{t=1}^{T} \alpha_t f_t(x))$
- $\alpha_t = \ln \sqrt{(1 - \varepsilon_t)/\varepsilon_t}$

Training Data Error Rate

$$= \frac{1}{N} \sum_n \delta(H(x^n) \neq \hat{y}^n)$$

$$= \frac{1}{N} \sum_n \delta(\hat{y}^n g(x^n) < 0)$$

$$\leq \frac{1}{N} \sum_n \exp(-\hat{y}^n g(x^n))$$

\[\hat{y}^n g(x^n)\]
Training Data Error Rate

\[ \leq \frac{1}{N} \sum_n \exp(-\hat{y}^n g(x^n)) \]

\[ = \frac{1}{N} Z_{T+1} \]

\[ g(x) = \sum_{t=1}^{T} \alpha_t f_t(x) \]

\[ \alpha_t = \ln \sqrt{(1 - \varepsilon_t)/\varepsilon_t} \]

\[ Z_t: \text{the summation of the weights of training data for training } f_t \]

What is \( Z_{T+1} =? \)

\[ Z_{T+1} = \sum_n u_{T+1}^n \]

\[ u_1^n = 1 \]

\[ u_{t+1}^n = u_t^n \times \exp(-\hat{y}^n f_t(x^n)\alpha_t) \]

\[ = \prod_{t=1}^{T} \exp(-\hat{y}^n f_t(x^n)\alpha_t) \]

\[ Z_{T+1} = \sum_n \prod_{t=1}^{T} \exp(-\hat{y}^n f_t(x^n)\alpha_t) \]

\[ = \sum_n \exp \left( -\hat{y}^n \sum_{t=1}^{T} f_t(x^n)\alpha_t \right) \]
Training Data Error Rate

$$\leq \frac{1}{N} \sum_n \exp(-\hat{y}^n g(x^n)) = \frac{1}{N} Z_{T+1}$$

$$Z_1 = N \quad \text{(equal weights)}$$

$$Z_t = Z_{t-1} \epsilon_t \exp(\alpha_t) + Z_{t-1} (1 - \epsilon_t) \exp(-\alpha_t)$$

Misclassified portion in $Z_{t-1}$  
Correctly classified portion in $Z_{t-1}$

$$= Z_{t-1} \epsilon_t \sqrt{(1 - \epsilon_t)/\epsilon_t} + Z_{t-1} (1 - \epsilon_t) \sqrt{\epsilon_t/(1 - \epsilon_t)}$$

$$= Z_{t-1} \times 2\sqrt{\epsilon_t(1 - \epsilon_t)}$$

$$Z_{T+1} = N \prod_{t=1}^{T} 2\sqrt{\epsilon_t(1 - \epsilon_t)}$$

Training Data Error Rate $\leq \prod_{t=1}^{T} 2\sqrt{\epsilon_t(1 - \epsilon_t)} < 1$

Smaller and smaller

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

$$\alpha_t = \ln \sqrt{(1 - \epsilon_t)/\epsilon_t}$$
End of Warning
Even though the training error is 0, the testing error still decreases?

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

\[ g(x) \]

Margin = \( \hat{y} g(x) \)
Large Margin?

Training Data Error Rate =

\[ \frac{1}{N} \sum_{n} \delta(H(x^n) \neq \hat{y}^n) \]

\[ \leq \frac{1}{N} \sum_{n} \exp(-\hat{y}^n g(x^n)) \]

\[ = \prod_{t=1}^{T} 2\sqrt{\epsilon_t (1 - \epsilon_t)} \]

Getting smaller and smaller as T increase
Experiment:
Function of Miku

Adaboost + Decision Tree
(depth = 5)

T = 10
T = 20
T = 50
T = 100
To learn more ...

• Introduction of Adaboost:
  • Freund; Schapire (1999). "A Short Introduction to Boosting“

• Multiclass/Regression

• Gentle Boost
  • Schapire, Robert; Singer, Yoram (1999). "Improved Boosting Algorithms Using Confidence-rated Predictions".
General Formulation of Boosting

• Initial function $g_0(x) = 0$
• For $t = 1$ to $T$:
  • Find a function $f_t(x)$ and $\alpha_t$ to improve $g_{t-1}(x)$
    • $g_{t-1}(x) = \sum_{i=1}^{t-1} \alpha_i f_i(x)$
    • $g_t(x) = g_{t-1}(x) + \alpha_t f_t(x)$
• Output: $H(x) = \text{sign}(g_T(x))$

What is the learning target of $g(x)$?

Minimize $L(g) = \sum_{n} l(\hat{y}^n, g(x^n)) = \sum_{n} \exp(-\hat{y}^n g(x^n))$
Gradient Boosting

- Find $g(x)$, minimize $L(g) = \sum_n \exp(-\hat{y}^n g(x^n))$
- If we already have $g(x) = g_{t-1}(x)$, how to update $g(x)$?

Gradient Descent:

\[
g_t(x) = g_{t-1}(x) - \eta \left. \frac{\partial L(g)}{\partial g(x)} \right|_{g(x) = g_{t-1}(x)}
\]

\[
\sum_n \exp(-\hat{y}^n g_{t-1}(x^n))(\hat{y}^n)
\]

\[
g_t(x) = g_{t-1}(x) + \alpha_t f_t(x)
\]
Gradient Boosting

\[ f_t(x) \leftarrow \sum_n \exp(-\hat{y}^n g_t(x^n))(\hat{y}^n) \]

We want to find \( f_t(x) \) maximizing

\[ \sum_n \exp(-\hat{y}^n g_{t-1}(x^n))((\hat{y}^n) f_t(x^n)) \]

Same sign

\[ u_t^n = \exp(-\hat{y}^n g_{t-1}(x^n)) = \exp\left(-\hat{y}^n \sum_{i=1}^{t-1} \alpha_i f_i(x^n)\right) \]

Exactly the weights we obtain in Adaboost
Gradient Boosting

- Find $g(x)$, minimize $L(g) = \sum_n \exp(-\hat{y}^n g(x^n))$

$$g_t(x) = g_{t-1}(x) + \alpha_t f_t(x)$$  \hspace{1cm} \alpha_t \text{ is something like learning rate}

Find $\alpha_t$ minimizing $L(g_{t+1})$

$$L(g) = \sum_n \exp(-\hat{y}^n (g_{t-1}(x) + \alpha_t f_t(x)))$$

$$= \sum_n \exp(-\hat{y}^n g_{t-1}(x)) \exp(-\hat{y}^n \alpha_t f_t(x))$$

$$= \sum_{\hat{y}^n \neq f_t(x)} \exp(-\hat{y}^n g_{t-1}(x^n)) \exp(\alpha_t)$$

$$+ \sum_{\hat{y}^n = f_t(x)} \exp(-\hat{y}^n g_{t-1}(x^n)) \exp(-\alpha_t)$$

Find $\alpha_t$ such that

$$\frac{\partial L(g)}{\partial \alpha_t} = 0$$

$$\alpha_t = \frac{\ln \sqrt{(1 - \varepsilon_t)/\varepsilon_t}}{\varepsilon_t}$$

Adaboost!
Cool Demo

- http://arogozhnikov.github.io/2016/07/05/gradient_boosting_playground.html
Ensemble: Stacking
Voting

X

小明’s system  →  y

老王’s system  →  y

老李’s system  →  y

小毛’s system  →  y

{ Majority Vote }
Stacking

- 小明’s system
- 老王’s system
- 老李’s system
- 小毛’s system

Training Data → Val Data → Testing Data

```
x
```

```
y
```

Final Classifier as new feature
2017
新年快樂
Happy New Year