Lecture 8: Ensemble Learning.

In ensemble learning, the idea is to combine multiple classifiers into a single one. Ensemble learning usually works very well in practice.

Two methods (for this class):
1. Bagging
2. Boosting

**BAGGING:** (Bootstrap AGGregating)

1. **Input:** \( n \) labelled training examples \((x_1,y_1), \ldots, (x_n,y_n)\)

2. **Algorithm:**
   Repeat \( k \) times:
   (a) Select \( m \) samples out of \( n \) with replacement from the training set to get training set \( S_i \)
   (b) Train classifier \( h_i \) on \( S_i \) (usually, \( h_i \)'s are the same type of classifier)

3. **Output:** Classifiers \( h_1, \ldots, h_k \)

**Testing:** Given test example \( x \), output the majority of \( h_1(x), h_2(x), \ldots, h_k(x) \) (break ties at random as usual)

1. **How to pick \( k \)?**
   Higher \( k \) is better, but also increases training time, storage requirement and classification time. So pick a \( k \) which is feasible.

2. **How to pick \( m \)?**
Popular choice for \( m = n \).

But this is still very different from working with the entire training set!

\[
\Pr(S_i = s) = \frac{n!}{n^n} \quad \text{(# Ways of choosing } n \text{ samples)}
\]

\( \Rightarrow \) only \( n! \) of these ways give you the entire training set!

\( \Rightarrow \frac{n!}{n^n} \) is a very tiny number \( \ll 2^{-n/2} \)

For any \((x_j, y_j)\), \( \Pr((x_j, y_j) \text{ is not } \in S_i) = (1 - \frac{1}{n})^n \times \frac{1}{e} \) (for large \( n \))

\( \frac{1}{e} \approx 0.37 \); so about 37\% of the data is left out of \( S_i \).

Why does bagging work?

It can be shown that bagging decreases the variance of a classifier. (It doesn't help much with bias).

Thus it prevents overfitting.
Boosting

Sometimes it is:
- easy to come up with simple, easy to use, rules of thumb classifiers
- but hard to come up with a single highly accurate rule.

Examples:

(1) Spam classification, based on email text.
   Certain words, eg. "Nigeria", "Online Pharmacy", etc. typically are a good indicator of spam.
   Rule of thumb: Does email contain word "Nigeria"?

(2) Detect if an image has a face in it.
   On an average, pixels around the eyes are darker than those below.
   Rule of thumb: Is the (average darkness in the shaded region) - (average darkness in the white rectangular region below) > 0 ?

Boosting gives us a way to combine these weak rules into good classifiers.

Definitions:

1. Weak Learner: A simple rule of thumb that doesn't necessarily work very well.

2. Strong Learner: A good classifier (with high accuracy)
Boosting Procedure:

1. Design method to find a good rule of thumb.
2. Repeat:
   - Find a good rule of thumb
   - Modify training data to get a second data set
   - Apply method to new data set to get a good rule of thumb, and so on.

1. How to get a good rule of thumb? Application specific (more later)
2. How to modify training data set?
   - Give highest weight to the hardest examples — those that were misclassified more often by previous rules of thumb.
3. How to combine the rules of thumb into a prediction rule?
   Take a weighted majority of the rules.

Let \( D \) be a distribution over labelled examples, and let \( h \) be a classifier.

Error of \( h \) wrt \( D \) is:

\[
\text{err}_D(h) = \Pr \left[ h(x) \neq y \right] = \int_D h(x) \neq y \, \text{d}D
\]

Example: \( D \):

\[
\begin{array}{cccccc}
 & 0 & \frac{1}{4} & \frac{1}{2} & \frac{3}{4} & 1 \\
\hline
\text{Pr} & 0 & 1/4 & 1/2 & 1/4 & 0
\end{array}
\]

\( X \): takes values \( \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1 \), each w.p. \( \frac{1}{4} \).

\( Y = 1 \) if \( X \) has \( \text{Pr} \) value \( \geq \frac{1}{2} \), \( 0/w \) \( Y = 0 \).

Then if \( h \) is the rule:

\[
h(x) = 1 \text{ if } x > \frac{1}{4}
\]

Then, \( \text{err}(h) = \frac{1}{4} \).
→ h is called a weak learner if $\text{err}_D(h) < 0.5$
→ Error of random guessing is 0.5 (with 2 labels)

Given training examples $(x_1, y_1), \ldots, (x_n, y_n)$, we can assign weights $w_1, \ldots, w_n$ to these examples. If $\sum_{t=1}^n w_t = 1$, $w_t > 0$, we can think of these weights as a probability distribution over the examples.

Error of a classifier h wrt $w$ is:

$$\text{err}_w(h) = \sum_{i=1}^n w_t \cdot 1(h(x_t) \neq y_t)$$

1 is the indicator function, where $1(P) = 1$ if P is true = 0 otherwise.

**Boosting Algorithm:**

**Input:** Training set $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, $y_i = \pm 1$

$D_1(i) = 1/n$ for all $i = 1, \ldots, n$

For $t = 1, 2, 3, \ldots$

$h_t = \text{weak-learner wrt } D_t$. (so, $\text{err}_{D_t}(h_t) < 0.5$)

$\varepsilon_t = \text{err}_{D_t}(h_t)$

$\alpha_t = \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}$ (so, $\alpha_t$ is high when $\varepsilon_t$ is low, and almost 0 when $\varepsilon_t$ is close to 0.5)

$D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$ (D_{t+1} goes ↑ if i is misclassified by $h_t$; so higher $D_t$ means harder example.

where $Z_t$ is a normalization constant to ensure that

$$\sum_{i} D_{t+1}(i) = 1.$$ 

**Final classifier:** $H(x) = \text{sign} \left( \sum_{t=1}^T \alpha_t h_t(x) \right)$ (weighted majority of $h_t(x)$'s)
Example of Weighted Error:

Suppose training data is: \((0,0), 1), ((1, 0), 1), ((0,1), -1)\)

Weights \(W:\)

\[ \frac{1}{2} \quad \frac{1}{4} \quad \frac{1}{4} \]

classification rule: Predict 1 if \(x_1 \leq \frac{1}{2}\), -1 otherwise.

\[ \text{err}_W(h) = \frac{1}{2} \times 0 + \frac{1}{4} \times 1 + \frac{1}{4} \times 0 = \frac{1}{4} \]

(The usual (unweighted) error would be \(\frac{2}{3}\)).

Boosting Algorithm Example:

Training data:

\((1, 1), +\) \((2, 1), -\) \((4, 1), -\) \((1, 2), +\) \((2, 2), -\) \((3, 2), -\) \((2, 3), +\) \((3, 3), +\) \((4, 3), -\) \((2, 4), +\)

Initially: \(D_1(i) = 0.1\) (for all \(i\))

\[\text{Suppose } h_1\]

Weak Learners: Set of vertical and horizontal thresholds.

\[\text{Suppose we pick } h_1(x) = + \text{ if } x_1 \leq 1.5\]

\(- \text{ otherwise}\)

Name the points: \(a, b, \ldots, j\) (for ease of understanding)

Then:

\[\text{err}_{D_1}(h_1) = \varepsilon_1 = 0.3 \quad \alpha_1 = 0.42\]

Weights of \(a, b, c, d, e, f, g\): \(D_2 = 0.07\)

Weights of \(h, i, j\): \(D_2 = 0.17\)

\[\varepsilon_2 = 7 \times 0.42 - 0.1 + 3 \times 0.1 \times 0.42\]

\[= 0.92\]

Note: Calculations rounded to 2 decimal places.
In Round 2, suppose we pick
\[ h_2(x) = + \text{ if } x_2 > 2.5 \]
\[ = - \text{ otherwise.} \]
\[ \text{err}_{D_2}(h_2) = \varepsilon_2 = 0.21 \]
\[ \alpha_2 = 0.66 \]
Weights of \( a, b \): \[ D_3 := 0.07 \times e^{0.66} / Z_3 = 0.17 \]
Weights of \( c, d, e, f \): \[ D_3 := 0.07 \times e^{0.66} / Z_3 = 0.04 \]
Weights of \( h, i, j \): \[ D_3 := 0.17 \times e^{-0.66} / Z_3 = 0.11 \]
Weight of \( g \): \[ D_3 := 0.07 \times e^{0.66} / Z_3 = 0.17 \]
\[ Z_3 = 0.81 \]

In Round 3, suppose we pick:
\[ h_3(x) = + \text{ if } x_1 \leq 3.5 \]
\[ = - \text{ otherwise.} \]
\[ \text{err}_{D_3}(h_3) = \varepsilon_3 = 0.12 \]
\[ \alpha_3 = 0.99 \]
Weights of \( a, b \): \[ D_4 := 0.17 \times e^{-0.99} / Z_4 = 0.1 \]
" " c, d, e: \[ D_4 := 0.04 \times e^{0.99} / Z_4 = 0.17 \]
" " h, i, j: \[ D_4 := 0.11 \times e^{-0.99} / Z_4 = 0.06 \]
" " f: \[ D_4 := 0.04 \times e^{-0.99} / Z_4 = 0.02 \]
" " g: \[ D_4 := 0.17 \times e^{-0.99} / Z_4 = 0.1 \]
\[ Z_4 = 0.65 \]
Final classifier: \[ \text{sign}(\alpha_4 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x)) \]
\[ = \text{sign}(0.42 h_1(x) + 0.66 h_2(x) + 0.99 h_3(x)) \]
When to stop boosting? Use a validation dataset to find a stopping time.
Stop when validation error does not improve.

**Boosting and Overfitting:**

Overfitting can happen with boosting, but often does not.

Typical boosting run:

Intuitively, margin of classification measures how far the + labels are from the - labels.

\[
\begin{array}{c c}
\begin{array}{c c c c c}
+++ & +++ & ++ + & + +
\end{array} & \begin{array}{c c c c c}
--- & -- -- & -- -- & -- --
\end{array}
\end{array}
\]

Small Margin \quad Large Margin

For boosting:
- think of each \( h_t(x) \) as a feature
- Feature space is:
  \[
  \left[ h_1(x), h_2(x), \ldots, h_T(x) \right]
  \]
- Margin of example \( x \) is:
  \[
  \left| \sum_{t=1}^{T} \alpha_t h_t(x) \right|
  \]
- If you have large margin data, then classifiers need less training examples to avoid overfitting. (This is also why kernels work, even if they are very high dimensional feature space.)

Reason is that the margin of classification often increases with boosting.
Applications of Boosting:

1. Boosted Decision Trees:

Weak learners are single node decision trees of the form:

- **Is Feature $f \leq t$**
  - Yes → Predict 1
  - No → Predict -1

- **Is Feature $f > t$**
  - Yes → Predict 1
  - No → Predict -1

2. Face detection: Viola and Jones: see slides.