Ensemble Learning
Introduction

In our daily life
- Asking different doctors’ opinions before undergoing a major surgery
- Reading user reviews before purchasing a product
- There are countless number of examples where we consider the decision of mixture of experts.

Ensemble systems follow exactly the same approach to data analysis.

Problem Definition
- Given
  - Training data set \( D \) for supervised learning
  - \( D \) drawn from common instance space \( \mathcal{X} \)
  - Collection of inductive learning algorithms
- Hypotheses produced by applying inducers to \( s(D) \)
  - \( s: \mathcal{X} \text{vector} \rightarrow \mathcal{X'} \text{vector} \) (sampling, transformation, partitioning, etc.)
- Return: new classification algorithm (not necessarily \( \in H \) for \( x \in \mathcal{X} \) that combines outputs from collection of classification algorithms

Desired Properties
- Guarantees of performance of combined prediction

Two Solution Approaches
- Train and apply each classifier; learn combiner function (s) from result
- Train classifier and combiner function (s) concurrently
Why We Combine Classifiers? [1]

- **Reasons for Using Ensemble Based Systems**
  - **Statistical Reasons**
    - A set of classifiers with similar training data may have different generalization performance.
    - Classifiers with similar performance may perform differently in field (depends on test data).
    - In this case, averaging (combining) may reduce the overall risk of decision.
    - In this case, averaging (combining) may or may not beat the performance of the best classifier.
  - **Large Volumes of Data**
    - Usually training of a classifier with a large volumes of data is not practical.
    - A more efficient approach is to
      - Partition the data into smaller subsets
      - Training different Classifiers with different partitions of data
      - Combining their outputs using an intelligent combination rule
  - **To Little Data**
    - We can use resampling techniques to produce non-overlapping random training data.
    - Each of training set can be used to train a classifier.
  - **Data Fusion**
    - Multiple sources of data (sensors, domain experts, etc.)
    - Need to combine systematically,
    - Example : A neurologist may order several tests
      - MRI Scan,
      - EEG Recording,
      - Blood Test
    - A single classifier cannot be used to classify data from different sources (heterogeneous features).
Why We Combine Classifiers? [2]

- **Divide and Conquer**
  - Regardless of the amount of data, certain problems are difficult for solving by a classifier.
  - Complex decision boundaries can be implemented using ensemble Learning.
Diversity

- **Strategy of ensemble systems**
  - Creation of many classifiers and combine their outputs in a such a way that combination improves upon the performance of a single classifier.

- **Requirement**
  - The individual classifiers must make errors on different inputs.
  - **If errors are different then strategic combination of classifiers can reduce total error.**

- **Requirement**
  - We need classifiers whose decision boundaries are adequately different from those of others.
  - Such a set of classifiers is said to be *diverse*.

- **Classifier diversity can be obtained**
  - Using different training data sets for training different classifiers.
  - Using unstable classifiers.
  - Using different training parameters (such as different topologies for NN).
  - Using different feature sets (such as random subspace method).

- **G. Brown, J. Wyatt, R. Harris, and X. Yao, “Diversity creation methods: a survey and categorization,” Information fusion, Vo. 6, pp. 5-20, 2005.**
Classifier diversity using different training sets
Diversity Measures (1)

- **Pairwise measures** (*assuming that we have T classifiers*)

<table>
<thead>
<tr>
<th></th>
<th>( h_j ) is correct</th>
<th>( h_j ) is incorrect</th>
</tr>
</thead>
<tbody>
<tr>
<td>( h_i ) is correct</td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>( h_i ) is incorrect</td>
<td>c</td>
<td>d</td>
</tr>
</tbody>
</table>

- **Correlation** (Maximum diversity is obtained when \( \rho = 0 \))

\[
\rho_{i,j} = \frac{ad - bc}{\sqrt{(a+b)(c+d)(a+c)(c+d)}} \quad 0 \leq \rho \leq 1
\]

- **Q-Statistics** (Maximum diversity is obtained when \( Q = 0 \)) \(|\rho| \leq |Q|\)

\[
Q_{i,j} = \frac{(ad - bc)}{(ad + bc)}
\]

- **Disagreement measure** (the prob. that two classifiers disagree)

\[
D_{i,j} = b + c
\]

- **Double fault measure** (the prob. that two classifiers are incorrect)

\[
DF_{i,j} = d
\]

- **For a team of T classifiers, the diversity measures are averaged over all pairs**:

\[
D_{avg} = \frac{2}{T(T-1)} \sum_{i=1}^{T-1} \sum_{j=1}^{T} D_{i,j}
\]
Diversity Measures (2)

- Non-Pairwise measures (assuming that we have $T$ classifiers)
  - Entropy Measure:
    - Makes the assumption that the diversity is highest if half of the classifiers are correct and the remaining ones are incorrect.
  - Kohavi-Wolpert Variance
  - Measure of difficulty

- Comparison of different diversity measures

<table>
<thead>
<tr>
<th>Name</th>
<th>$\uparrow$ / $\downarrow$</th>
<th>P</th>
<th>S</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q-statistic</td>
<td>$Q$</td>
<td>$\downarrow$</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Correlation coefficient</td>
<td>$\rho$</td>
<td>$\downarrow$</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Disagreement measure</td>
<td>$D$</td>
<td>$\uparrow$</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Double-fault measure</td>
<td>$DF$</td>
<td>$\downarrow$</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Kohavi-Wolpert variance</td>
<td>$kw$</td>
<td>$\uparrow$</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Interrater agreement</td>
<td>$\kappa$</td>
<td>$\downarrow$</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Entropy measure</td>
<td>$Ent$</td>
<td>$\uparrow$</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Measure of difficulty</td>
<td>$\theta$</td>
<td>$\downarrow$</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Generalised diversity</td>
<td>$GD$</td>
<td>$\uparrow$</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Coincident failure diversity</td>
<td>$CFD$</td>
<td>$\uparrow$</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

*Note: The arrow specifies whether diversity is greater if the measure is lower ($\downarrow$) or greater ($\uparrow$). ‘P’ stands for ‘Pairwise’ and ‘S’ stands for ‘Symmetrical’.*
Diversity Measures (3)

- **No Free Lunch Theorem**: No classification algorithm is universally correlates with the higher accuracy.
  - Conclusion: There is no diversity measure that consistently correlates with the higher accuracy.
  - Suggestion: In the absence of additional information, the Q statistics is suggested because of its intuitive meaning and simple implementation.

- **Reference**:
Design of Ensemble Systems

- Two key components of an ensemble system
  - Creating an ensemble by creating *weak learners*
    - Bagging
    - Boosting
    - Stacked generalization
    - Mixture of experts
  - Combination of classifiers' outputs
    - Majority Voting
    - Weighted Majority Voting
    - Averaging

- What *is* a weak classifier?
  - One not guaranteed to do better than random guessing (1 / number of classes)
  - Goal: combine multiple weak classifiers, get one at least as accurate as strongest

- Combination Rules
  - Trainable vs. Non-Trainable
  - Labels vs. Continuous outputs
In ensemble learning, a rule is needed to combine outputs of classifiers.

- **Classifier Selection**
  - Each classifier is trained to become an expert in some local area of feature space.
  - Combination of classifiers is based on the given feature vector.
  - Classifier that was trained with the data closest to the vicinity of the feature vector is given the highest credit.
  - One or more local classifiers can be nominated to make the decision.

- **Classifier Fusion**
  - Each classifier is trained over the entire feature space.
  - Classifier Combination involves merging the individual *weak* classifier design to obtain a single *strong* classifier.
Combination Rule [2] : Majority Voting

- **Majority Based Combiner**
  - **Unanimous voting**: All classifiers agree the class label
  - **Simple majority**: At least one or more than half of the classifiers agree the class label
  - **Majority voting**: Class label that receives the highest number of votes.

- **Weight-Based Combiner**
  - Collect votes from pool of classifiers for each training example
  - Decrease weight associated with each classifier that guessed wrong
  - Combiner predicts weighted majority label

- **How we do assign the weights?**
  - Based on Training Error
  - Using Validation set
  - Estimate of the classifier’s future performance

- **Other combination rules**
  - Behavior knowledge space, Borda count
  - Mean rule, Weighted average
Bagging [1]

- **Application of bootstrap sampling**
  - Given: set $D$ containing $m$ training examples
  - Create $S[i]$ by drawing $m$ examples at random with replacement from $D$
  - $S[i]$ of size $m$ expected to leave out 75%-100% of examples from $D$

- **Bagging**
  - Create $k$ bootstrap samples $S[1], S[2], ..., S[k]$
  - Train distinct inducer on each $S[i]$ to produce $k$ classifiers
  - Classify new instance by classifier vote (majority vote)

- **Variations**
  - **Random forests**
    - Can be created from decision trees, whose certain parameters vary randomly.
  - **Pasting small votes (for large datasets)**
    - RVotes: Creates the data sets randomly
    - IVotes: Creates the data sets based on the importance of instances, easy to hard!
Bagging [2]

**Algorithm: Bagging**

**Input:**
- Training data $S$ with correct labels $\omega_i \in \Omega = \{\omega_1, \ldots, \omega_C\}$ representing $C$ classes
- Weak learning algorithm **WeakLearn**
- Integer $T$ specifying number of iterations.
- Percent (or fraction) $F$ to create bootstrapped training data

**Do** $t = 1, \ldots, T$

1. Take a bootstrapped replica $S_t$ by randomly drawing $F$ percent of $S$.
2. Call **WeakLearn** with $S_t$ and receive the hypothesis (classifier) $h_t$.
3. Add $h_t$ to the ensemble, $E$.

**Test: Simple Majority Voting** – Given unlabeled instance $x$

1. Evaluate the ensemble $E = \{h_1, \ldots, h_T\}$ on $x$.

2. Let $v_{t,j} = \begin{cases} 
1, & \text{if } h_t \text{ picks class } \omega_j \\ 
0, & \text{otherwise} 
\end{cases}$ (8)

be the vote given to class $\omega_j$ by classifier $h_t$.

3. Obtain total vote received by each class

$$V_j = \sum_{t=1}^{T} v_{t,j}, \ j = 1, \ldots, C$$ (9)

4. Choose the class that receives the highest total vote as the final classification.
**Bagging: Pasting small votes (IVotes)**

**Algorithm: Pasting Small Votes (IVotes)**

**Input:**
1. Training data $S$ with correct labels $\omega_i \in \Omega = \{\omega_1, \ldots, \omega_C\}$ representing $C$ classes;
2. Weak learning algorithm **WeakLearn**;
3. Integer $T$ specifying number of iterations;
4. *Bitesize* $M$, indicating the size of individual training subsets to be created.

**Initialize**

1. Choose a random subset $S_0$ of size $M$ from $S$.
2. Call **WeakLearn** with $S_0$, and receive the hypothesis (classifier) $h_0$.
3. Evaluate $h_0$ on a validation dataset, and obtain error $\varepsilon_0$ of $h_0$.
4. If $\varepsilon_0 > \frac{1}{2}$, return to step 1.

**Do** $t=1, \ldots, T$

1. Randomly draw an instance $x$ from $S$ according to uniform distribution.
2. Evaluate $x$ using majority vote of out-of-bag classifiers in the current ensemble $E_t$.
3. If $x$ is misclassified, place $x$ in $S_t$. Otherwise, place $x$ in $S_t$ with probability $p$

$$p = \frac{\varepsilon_{t-1}}{(1-\varepsilon_{t-1})}.$$  \hspace{1cm} (10)

Repeat Steps 1-3 until $S_t$ has $M$ such instances.

4. Call **WeakLearn** with $S_t$ and receive the hypothesis $h_t$.
5. Evaluate $h_t$ on a validation dataset, and obtain error $\varepsilon_t$ of $h_t$. If $\varepsilon_t > \frac{1}{2}$, return to step 4.
6. Add $h_t$ to the ensemble to obtain $E_t$.

**End**
Schapire proved that a weak learner, an algorithm that generates classifiers that can merely do better than random guessing, can be turned into a strong learner that generates a classifier that can correctly classify all but an arbitrarily small fraction of the instances.

- In boosting, the training data are ordered from easy to hard.
- Easy samples are classified first, and hard samples are classified later.

- Create the first classifier same as Bagging
- The second classifier is trained on training data only half of which is correctly classified by the first one and the other half is misclassified.
- The third one is trained with data that two first disagree.

- Variations
  - AdaBoost.M1
  - AdaBoost.R
Boosting

**Algorithm: Boosting**

**Input:**
- Training data $S$ of size $N$ with correct labels $\omega_i \in \Omega = \{\omega_1, \omega_2\}$;
- Weak learning algorithm `WeakLearn`.

**Training**

1. Select $N_1 < N$ patterns without replacement from $S$ to create data subset $S_1$.
2. Call `WeakLearn` and train with $S_1$ to create classifier $C_1$.
3. Create dataset $S_2$ as the most informative dataset, given $C_1$, such that half of $S_2$ is correctly classified by $C_2$, and the other half is misclassified. To do so:
   a. Flip a fair coin. If Head, select samples from $S$, and present them to $C_1$ until the first instance is misclassified. Add this instance to $S_2$.
   b. If Tail, select samples from $S$, and present them to $C_1$ until the first one is correctly classified. Add this instance to $S_2$.
   c. Continue flipping coins until no more patterns can be added to $S_2$.
4. Train the second classifier $C_2$ with $S_2$.
5. Create $S_3$ by selecting those instances for which $C_1$ and $C_2$ disagree. Train the third classifier $C_3$ with $S_3$.

**Test** – Given a test instance $x$

1. Classify $x$ by $C_1$ and $C_2$. If they agree on the class, this class is the final classification.
2. If they disagree, choose the class predicted by $C_3$ as the final classification.
Algorithm AdaBoost.M1
Input:
- Sequence of $N$ examples $S = \{(x_i, y_i), i = 1, \ldots, N\}$ with labels $y_i \in \Omega, \Omega = \{\omega_1, \ldots, \omega_C\}$;
- Weak learning algorithm WeakLearn;
- Integer $T$ specifying number of iterations.

Initialize $D_1(i) = \frac{1}{N}, i = 1, \ldots, N$ \hspace{1cm} (11)

Do for $t = 1, 2, \ldots, T$:
1. Select a training data subset $S_t$, drawn from the distribution $D_t$.
2. Train WeakLearn with $S_t$, receive hypothesis $h_t$.
3. Calculate the error of $h_t$: $\varepsilon_t = \sum_{i: h_t(x_i) \neq y_i} D_t(i)$. \hspace{1cm} (12)
   If $\varepsilon_t > \frac{1}{2}$, abort.
4. Set $\beta_t = \varepsilon_t / (1 - \varepsilon_t)$. \hspace{1cm} (13)
5. Update distribution
   $$D_t: D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t, & \text{if } h_t(x_i) = y_i \\ 1, & \text{otherwise} \end{cases}$$ \hspace{1cm} (14)
   where $Z_t = \sum_i D_t(i)$ is a normalization constant chosen so that $D_{t+1}$ becomes a proper distribution function.

Test – Weighted Majority Voting: Given an unlabeled instance $x$,
1. Obtain total vote received by each class
   $$V_j = \sum_{i: h_t(x_i) = \omega_j} \log \frac{1}{\beta_t}, \hspace{0.5cm} j = 1, \ldots, C.$$ \hspace{1cm} (15)
2. Choose the class that receives the highest total vote as the final classification.
Stacked Generalization (Stacking)

Intuitive Idea

- Train multiple learners
  - Each uses subsample of $D$
  - May be ANN, decision tree, etc.
- Train combiner on validation segment
Mixture Models

Intuitive Idea

- Train multiple learners
  - Each uses subsample of $D$
  - May be ANN, decision tree, etc.
- Gating Network usually is NN

![Diagram of Mixture Model]

Machine Learning
Cascading

Use \( d_j \) only if preceding ones are not confident

Cascade learners in order of complexity
T. G. Dietterich, “Machine Learning Research: four current directions”, Department of computer science, Oregon State University

T. G. Dietterich, “Ensemble Methods in Machine Learning”, Department of computer science, Oregon State University

Ron Meir, Gunnar Ratsch, “An introduction to Boosting and Leveraging”, Australian National University
