CHAPTER 15:
Combining Multiple Learners
Rationale

- No Free Lunch thm: There is no algorithm that is always the most accurate
- Generate a group of base-learners which when combined has higher accuracy
- Different learners use different
  - Algorithms
  - Hyperparameters
  - Representations (Modalities)
  - Training sets
  - Subproblems

Voting

- Linear combination
  \[ y = \sum_{j=1}^{L} w_j d_j \]
  \[ w_j \geq 0 \text{ and } \sum_{j=1}^{L} w_j = 1 \]
- Classification
  \[ y_i = \sum_{j=1}^{L} w_j d_{ji} \]
Bayesian perspective:

\[ P(C_i | x) = \sum_{all\ models\ M_j} P(C_i | x, M_j) P(M_j) \]

- If \( d_j \) are iid

\[
E[y] = E\left[ \sum_{i=1}^{L} \frac{1}{L} \cdot d_i \right] = \frac{1}{L} \cdot E[d_i] = E[d_i] \\
Var(y) = Var\left( \sum_{i=1}^{L} \frac{1}{L} \cdot d_i \right) = \frac{1}{L^2} \cdot Var\left( \sum_{i=1}^{L} d_i \right) = \frac{1}{L^2} \cdot L \cdot Var(d_i) = \frac{1}{L} \cdot Var(d_i)
\]

Bias does not change, variance decreases by \( L \)

- Average over randomness

---

Bagging

- Use bootstrapping to generate \( L \) training sets and train one base-learner with each (Breiman, 1996)
- Draw \( L \) training sets at random with replacement.
- Use voting (Average or median with regression)
- Unstable algorithms profit from bagging
- Unstable algorithms: if small changes in the training set causes large difference in the generated learner: the algorithm has high variance. E.g., decision trees, multilayer perceptrons.
Boosting

- In bagging: generating complementary base-learner is left to chance and to the unstability of the learning methods
- In Boosting: actively try to generate complementary base-learner
- How: by training the next learner based on the mistakes of previous learners.
- Schapire 1990: combine three weak learners to generate a strong learner.
- Weak learner: error probability less than 1/2

**AdaBoost**

Adaptive Boosting:
Generate a sequence of base-learners each focusing on previous one’s errors
(Freund and Schapire, 1996)

Training:

For all $(x^i, r^i)_{i=1}^N \in \mathcal{X}$, initialize $p_j^i = 1/N$
For all base-learners $j = 1, \ldots, L$
Randomly draw $x^i_j$ from $\mathcal{X}$ with probabilities $p_j^i$
Train $d_j$ using $x^i_j$
For each $(x^i, r^i)$, calculate $y_j^i = d_j(x^i)$
Calculate error rate: $e_j = \sum_i p_j^i \cdot 1(y_j^i \neq r^i)$
If $e_j > 1/2$, then $L \leftarrow j - 1$; stop
Set $\beta_j = e_j / (1 - e_j)$
For each $(x^i, r^i)$, decrease probabilities if correct
If $y_j^i = r^i$, $p_j^i+1 = \beta_j p_j^i$ Else $p_j^i+1 = p_j^i$
Normalize probabilities:
$z_j = \sum_i p_j^i+1$; $p_j^i+1 \leftarrow p_j^i+1 / z_j$

Testing:

Given $x$, calculate $d_j(x), j = 1, \ldots, L$
Calculate class outputs, $i = 1, \ldots, K$:
$y_i = \sum_{j=1}^L \left( \log \frac{1}{\beta_j} \right) d_j(x)$
AdaBoost

- AdaBoost works because it increases the margin at each step as the sample probabilities change
- Not all algorithms will benefit from Boosting
- Base-learner has to be simple and not accurate (high variance)

Mixture of Experts

Voting where weight:

\[ y = \sum_{j=1}^{L} w_j d_j \]

(Jacobs et al., 1991)

Experts or gating can be nonlinear
Mixture of Experts

- In RBF, each local fit is a constant, $w_{ih}$, second layer weight.
- In MoE, each local fit is a linear function of $x$, a local expert:

$$w_{ih}^t = v_{ih}^t x^t$$

(Jacobs et al., 1991)

MoE as Models Combined

- Radial gating:

$$g_h^i = \frac{\exp\left[-\|x^t - m_i\|^2 / 2s_h^2\right]}{\sum_i \exp\left[-\|x^t - m_i\|^2 / 2s_i^2\right]}$$

- Softmax gating:

$$g_h^i = \frac{\exp[m_i^T x^t]}{\sum_i \exp[m_i^T x^t]}$$
Stacking

- Combiner $f()$ is another learner (Wolpert, 1992)

Cascading

Use $d_i$ only if preceding ones are not confident

Cascade learners in order of complexity