No learner perfect!

- No Free Lunch Theorem
  - no single learning algorithm in any domain always inducing the most accurate learner.
  - Each learning algorithm dictates a certain model that comes with a set of assumptions.
    - This inductive bias leads to error if the assumptions do not hold for the data.
  - With finite data, each algorithm converges to a different solution and fails under different circumstances.
- A suitable combination of multiple base-learners should improve the accuracy.

Issues on combining learners

- Overhead of combining multiple learner.
  - Increase of space and time complexity.
  - Model combination may not increase accuracy.
- Two key issues:
  - How do we generate base-learners that complement each other?
  - How do we combine the outputs of base-learners for maximum accuracy?
- Maximizing individual accuracies and the diversity between learners.

Diversification: different techniques

- Different Algorithms
  - different learning algorithms to train different base-learners.
    - Parametric and non-parametric methods.
- Different Hyperparameters
  - the same learning algorithm but use it with different hyperparameters.
    - number of hidden units in a multilayer perceptron,
    - k in k-nearest neighbor,
    - error threshold in decision trees,
    - kernel function in support vector machines,
    - Initial weights of ANN.

Average of multiple base-learners trained with different hyperparameter values, to reduce variance, and therefore error.

Diversification: different techniques

- Different Input Representations
  - Integrating different types of sensors / measurements / modalities.
    - Sensor fusion: Audio and Video of lip movement to recognize speech.
    - Random sub-space: Use different feature subsets in learning.
      - Different learners will look from different points.
        - Random forest.
      - Reduce the curse of dimensionality.

Diversification: different techniques

- Different Training Sets
  - Different subsets of training samples:
    - Bagging.
  - trained serially so that instances on which the preceding base-learners are not accurate are given more emphasis in training later base-learners
    - boosting and cascading
    - actively try to generate complementary learners, instead of leaving this to chance.
  - Partitioning on locality of training space.
    - each base-learner trained on instances in a certain local part of the input space
      - Mixture of experts.

Diversity vs. Accuracy

- The base-learner to be simple
  - not chosen for its accuracy.
  - enough if performs with error rate less 50% for binary classification.
    - Operates marginally better than random guesses.
  - Final accuracy of combination should be high.
- the base-learners to be diverse
  - accurate on different instances, specializing in subdomains of the problem.

Model combination schemes

- Multi-expert combination
  - base-learners work in parallel.
  - Global approach:
    - All learners produce o/p given i/p and fusion of decisions.
      - Voting, Stacking
  - Local approach
    - Selected learners (mixture of experts) produce output.
      - A gating model for selecting experts by looking at input.

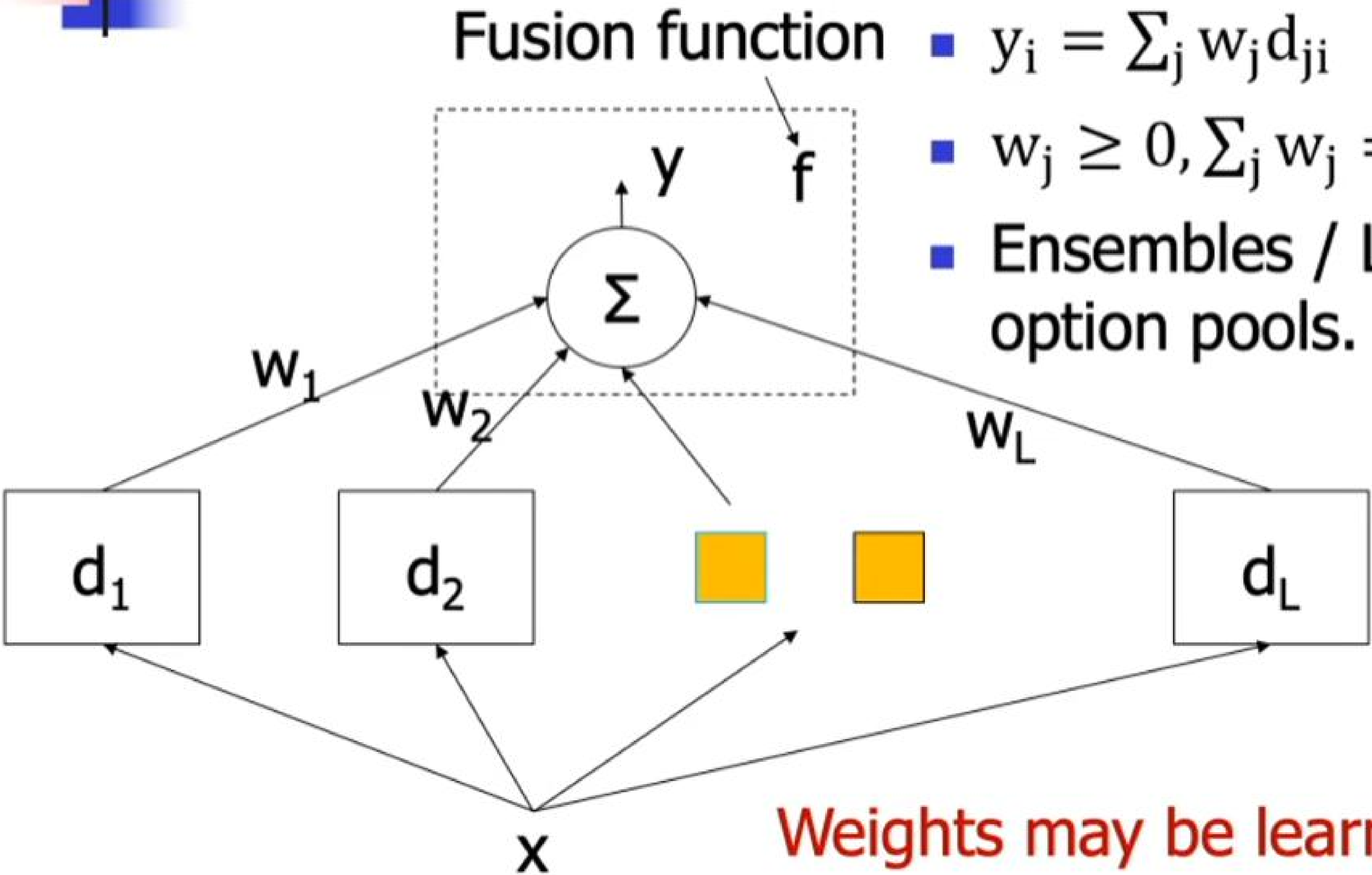
Model combination schemes

- Multi-stage combination
  - a serial approach
    - the next base-learner trained with or tested on only the instances where the previous base-learners are not accurate enough.
    - Base-learners sorted in complexity.
      - Complex learners used if preceding simpler learners not confident.
        - Cascading

Decision fusion

- L learners
- $d_j(x)$: decision for j th learner M_j.
- $y=f(d_1(x),d_2(x),\ldots,d_L(x)|\Phi)$
  - Φ is the set of parameters.
  - y : Final prediction of the combined learners.
- For k outputs from each j th learner
  - d_{ij}, $i=1,2,..k$ and $j=1,2,..L$
  - $y_i = f(d_{i1}(x),d_{i2}(x),\ldots,d_{iL}(x)|\Phi)$, $i=1,2,..,k$
    - Predict on y_i's. e.g. assign i th class if y_i is maximum.

Voting



- $y_i = \sum_j w_j d_{ji}$
- $w_j \geq 0, \sum_j w_j = 1$
- Ensembles / Linear option pools.

Weights may be learned too!

Classifier combination rules

- Different types of fusion functions.
  - Sum or average, weighted average, max, min, median, product, etc.
  - Sum rule most widely used in practice
    - majority (two-class) / plurality (multi-class) principle.
  - Median rule more robust to outliers.
  - Minimum and maximum rules respectively pessimistic and optimistic.
  - The product rule empowers each learner veto power.
    - 0/1 decision cases.
  - After the combination rules, y_i not necessarily sums up to 1.

Bayesian combination rule

- Weights approximating prior probabilities of models.
- Let $w_j = P(M_j)$, $d_{ij} = P(C_i|x, M_j)$

$$P(C_i|x) = \sum_{M_j} P(M_j) P(C_i|x, M_j)$$

- Instead of all models in the space, choose only those who have high $P(M_j)$.
- For each classifier if P(error)<1/2, with the increase of number of classifiers, accuracy increases by majority voting.

Hansen, L. K., and P. Salamon. 1990. "Neural Network Ensembles." IEEE Transactions on Pattern Analysis and Machine Intelligence 12: 993–1001.

Expectation, bias and variance

- Assume d_j's are iids with the expected value $E(d_j)$ and variance $var(d_j)$.
- For simple average ($w_j=1/L$):

$$\mathrm{E}(\mathrm{y}) = \mathrm{E}(\tfrac{1}{\mathrm{L}}\textstyle\sum_{\mathrm{j}} \mathrm{d_j}) = \tfrac{1}{\mathrm{L}} \mathrm{LE}(\mathrm{d_j}) = \mathrm{E}(\mathrm{d_j})$$

$$var(y) = var\left(\frac{1}{L}\sum_j d_j\right) = \frac{1}{L^2} L.\, var(d_j) = \frac{1}{L} var(d_j)$$

- If they are not *independent*,

$$var(y) = var\left(\frac{1}{L}\sum_j d_j\right) = \frac{1}{L^2} var\left(\sum_j d_j\right) = \frac{1}{L^2}\left(\sum_j var(d_j) + 2\sum_j \sum_{i<j} cov(d_i, d_j)\right)$$

-ve correlation may improve variance, but difficult to satisfy both accuracy more than 50% but negatively correlated.

Error correcting output codes (ECOC)

- For each class a set of binary classification tasks predefined.
- Coded in K x L matrix for K classes and L classifiers.
- Each row represents the signature of a class.
- Each column defines partitioning of classes into two sets labeled by either +1 or -1.
- The codes corresponding to class should follow error correcting codes principle
  - by keeping sufficient distance (Hamming distance) between any pair of them.

An example of ECOC codes L=7, K=4

- columns of W to be as different as possible
  - the tasks to be learned by the base-learners to be as different from each other as possible
  - Predefined tasks may not be simple to learn.

$$W = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & +1 & +1 & +1 & +1 \\ -1 & +1 & +1 & -1 & -1 & +1 & +1 \\ +1 & -1 & +1 & -1 & +1 & -1 & +1 \end{bmatrix} \qquad y_i = \sum_j w_{ij} d_j$$

Choose the class with the highest y_j.

Given posterior p_j (in [0,1]), we make d_j in [-1,1]:

$$d_j = 2p_j - 1$$

Bagging (Bootstrap Aggregating)

- A voting method whereby each base-learner trained over slightly different training sets.
  - of similar structure and mathematical form, but with different set of parameters.
  - Sampling with replacement.
    - Possible to have repeated samples in the training set.
  - Used for both classifications and regression.
    - For regression median is used to make the estimation more robust to outliers.
  - Small change in data, if causes large variation in model, learning is unstable.
    - E.g. Decision trees, ANNs are unstable.

Boosting

- Generating complementary base learners.
  - Training the next base learner from the mistakes of the previous learners.
  - Bagging: left to the chance factor and instability of training algorithm.
- Series of weak learners.
  - Weak learner: Error prob. < ½
  - Strong learner: Error prob. as small as possible.
- Original boosting algorithm.
  - A combination of 3 weak learners.

Boosting by three weak learners in tandem

- Randomly divide training samples into 3 sets, X_1,X_2 and X_3.
- Train d_1 with $X_{1,}$ and test d_1 with $X_{2.}$
- Form a training set X_2' for training d_2.
  - with misclassified samples of X_2 and as many as correctly classified samples by d_1.
- Train d_2 with X_2', and test X_3 with d_1 and d_2.
- Train d_3 with instances disagreed by d_1 and d2.
- Testing:
  - If a sample X has same labels by d_1 and d_2, accept it, else accept the result from d_3.

AdaBoost (Adaptive Boosting)

- Uses the same training set over and over
  - training set need not be large.
  - the classifiers should be simple so that they do not overfit.
  - Each should perform with error rate < ½.
  - Combines an arbitrary number of base learners, not just three.
- Many variants exist.
  - Randomly draw samples to form a training set each with varying probability.
    - Easier to classify, smaller the probability.

Freund, Y., and R. E. Schapire. 1996. "Experiments with a New Boosting Algorithm." In Thirteenth International Conference on Machine Learning, ed. L. Saitta, 148–156. San Mateo, CA: Morgan Kaufmann.

AdaBoost.M1: The original algorithm (Training)

- At each iteration i train with the sample set and compute the training error e of classification.
  - If $e>1/2$, stop (no more classifier required in the set).
  - Else
    - include the model d_i in the list
    - update the sampling prob. of each t th training sample,
      - by decreasing which are classified correctly with the weight $w^{(i)}$: $p^{(t)}=w^{(i)} \cdot p^{(t)}$, where $w^{(i)}=e/(1-e)$. <1
      - Normalize probabilities of samples at each iteration.
      - $\log(1/w^{(i)})$ is taken as the weight of the decision from that model during voting.

AdaBoost.M1: The original algorithm (Testing)

- Given x calculate $d_j(x)$, j=1,2,..,L
- Calculate class outputs y_i, i=1,2,..,K

$$y_i = \sum_j log\left(\frac{1}{w^{(j)}}\right) d_j(x)$$

- Assign the class with maximum y.

- Use simple classifier so that error is not low.
- Decision tree grown up to one or two levels (Decision stump).
- Linear discriminant classifiers not useful.
  - Low variance

Mixture of experts

- In voting weights are fixed for each classifier (expert).
- In mixture of experts, depending upon inputs these weights would vary. Ideally local experts (on the locality of input) would have weight close to 1 and the rest close to 0.
  - Voting by gating system.
  - Final classification score for each class the weighted means of votes.

Stacked generalization

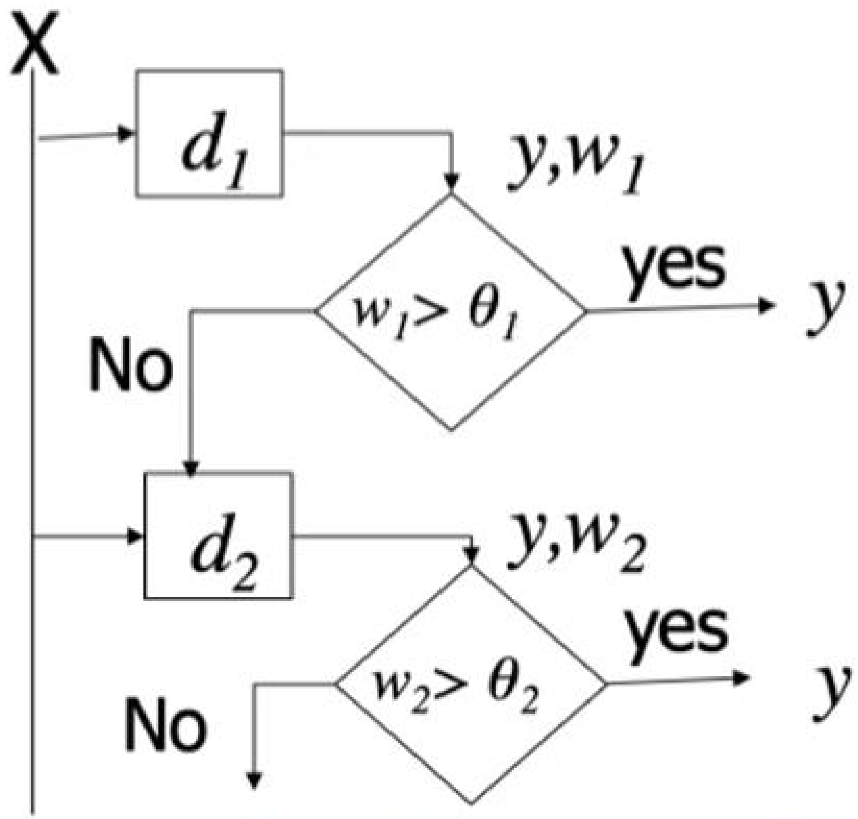
- Instead of linear combination it could be any general functional forms with parameters Φ, which are also learned.
  - $f(d_1, d_2, \ldots, d_L | \Phi)$
  - It could be a multilayer perceptron
    - Input d_j's and output y.

Cascading

d_1 is less costly than d_2 and so on.

w_j is confidence (e.g. posterior prob.) of decision for d_j.

- Base learners (d_i's) ordered in terms of complexity.
- Each learner produces output (y) with a confidence (w).
  - The next base leaner used if previous learners' decisions lack confidence.



Courtesy: "Introduction to Machine Learning" by Ethem Alpaydin (Chapter 17, Fig. 17.5)

Summary

- No learner perfect.
- Simple but diverse set of learners.
- Decision fusion
  - Voting
  - Bayesian combination rule.
- Error correcting output codes.
- Bagging (Bootstrap Aggregating).
- Boosting
  - AdaBoost,
- Mixture of experts.
- Stacked generalization
- Cascading.