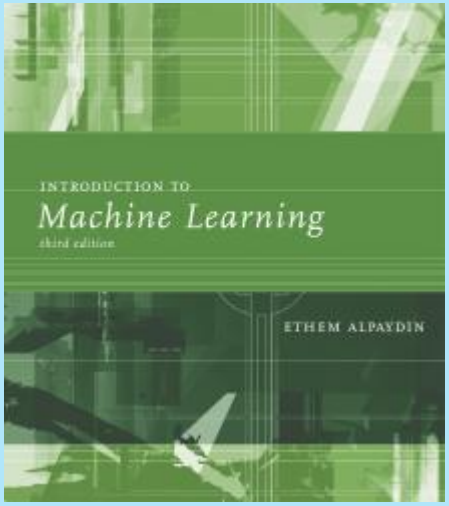


Lecture Slides for

INTRODUCTION TO MACHINE LEARNING

3RD EDITION

ETHEM ALPAYDIN

alpaydin@boun.edu.tr
http://www.cmpe.boun.edu.tr/~ethem/i2ml3e

CHAPTER 17:

COMBINING MULTIPLE LEARNERS

Rationale

- No Free Lunch Theorem: There is no algorithm that is always the most accurate
- Learning is an ill-posed problem and with finite data, each algorithm converges to a different solution and fails under different circumstances.
- Generate a group of base-learners which when combined has higher accuracy
- 1. How do we generate base-learners that complement each other?
- 2. How do we combine the outputs of base-learners for maximum accuracy?

Generating Diverse Learners

- Maximizing individual accuracies and the diversity between learners. Different ways to achieve this:
  - Different Algorithms
  - Different Hyperparameters
  - Different Input Representations
  - Different Training Sets: *bagging*, *boosting* and *cascading*, mixture of experts
  - Diversity vs. Accuracy

Model Combination Schemes

- *Multiexpert combination* methods: base-learners that work in *parallel*. These methods can in turn be divided into two:
  - *global* approach (*learner fusion*): all base-learners generate an output and all these outputs are used, examples are *voting* and *stacking*
  - *local* approach, or *learner selection*, for example, in *mixture of experts*, there is a *gating* model, which looks at the input and chooses one (or very few) of the learners as responsible for generating the output.
- *Multistage combination methods*: use a *serial* approach where the next base-learner is trained with or tested on only the instances where the previous base-learners are not accurate enough (cascading).

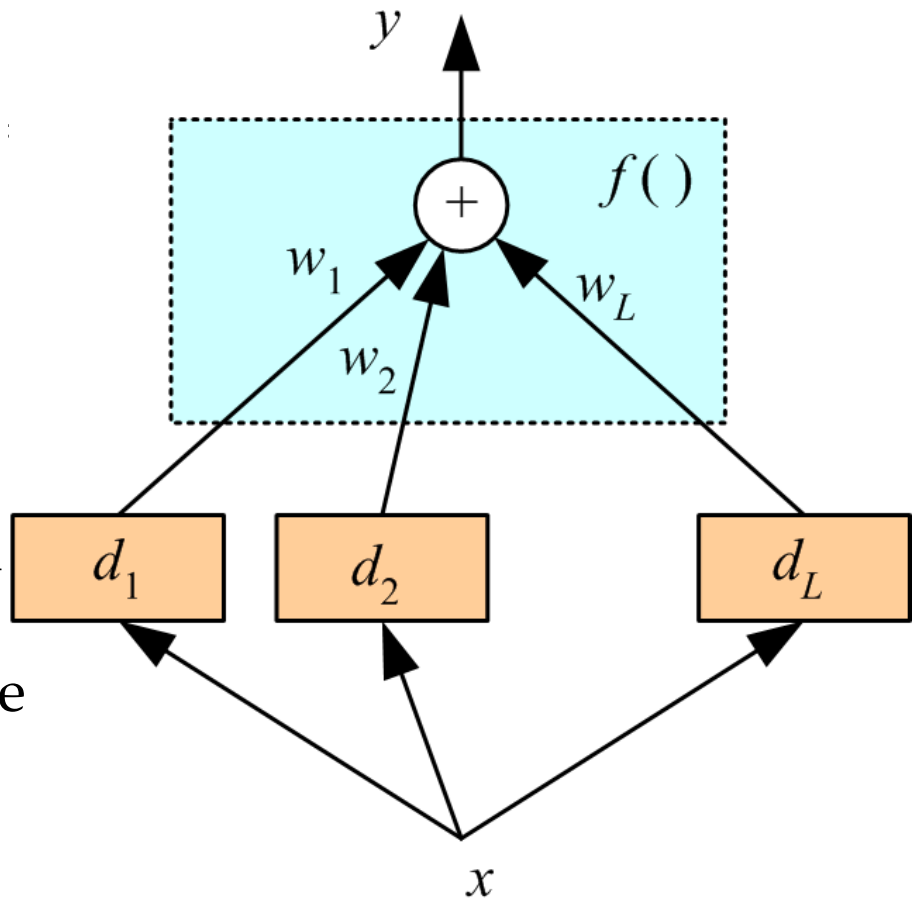
Voting

- Linear combination

$$y_i = \sum_{j=1}^{L} w_j d_{ji} \quad , w_j \geq 0 \quad \text{and} \quad \sum_{j=1}^{L} w_j$$

(ensembles and *linear opinion pools)*

- Classification



When there are K outputs, for each learner there are $d_{ji}(x)$, i = 1, . . . , K, j=1,. . . , L, and, combining them, we also generate K values, y_i, i =1,…,K

Choose C_i if $y_i = \max_{k=1}^{K} y_k$

Fixed Combination Rules

Classifier combination rules

Rule	Fusion function $f(\cdot)$
Sum	$y_i = \frac{1}{L}\sum_{j=1}^{L} d_{ji}$
Weighted sum	$y_i = \sum_j w_j d_{ji}, w_j \geq 0, \sum_j w_j = 1$
Median	$y_i = \text{median}_j d_{ji}$
Minimum	$y_i = \min_j d_{ji}$
Maximum	$y_i = \max_j d_{ji}$
Product	$y_i = \prod_j d_{ji}$

	C_1	C_2	C_3
d_1	0.2	0.5	0.3
d_2	0.0	0.6	0.4
d_3	0.4	0.4	0.2
Sum	0.2	**0.5**	0.3
Median	0.2	**0.5**	0.4
Minimum	0.0	**0.4**	0.2
Maximum	0.4	**0.6**	0.4
Product	0.0	**0.12**	0.032

Example of combination rules on three learners and three classes

- Bayesian perspective: $w_j \equiv P\left(\mathcal{M}_j\right), d_{ij} = P\left(C_i|x,\mathcal{M}_j\right)$

$$P\left(C_i|x\right) = \sum_{\text{all models } \mathcal{M}_j} P\left(C_i|x,\mathcal{M}_j\right) P\left(\mathcal{M}_j\right)$$

- If d_j are *iid*

$$E\left[y\right] = E\left[\sum_j \frac{1}{L} d_j\right] = \frac{1}{L} L \cdot E\left[d_j\right] = E\left[d_j\right]$$

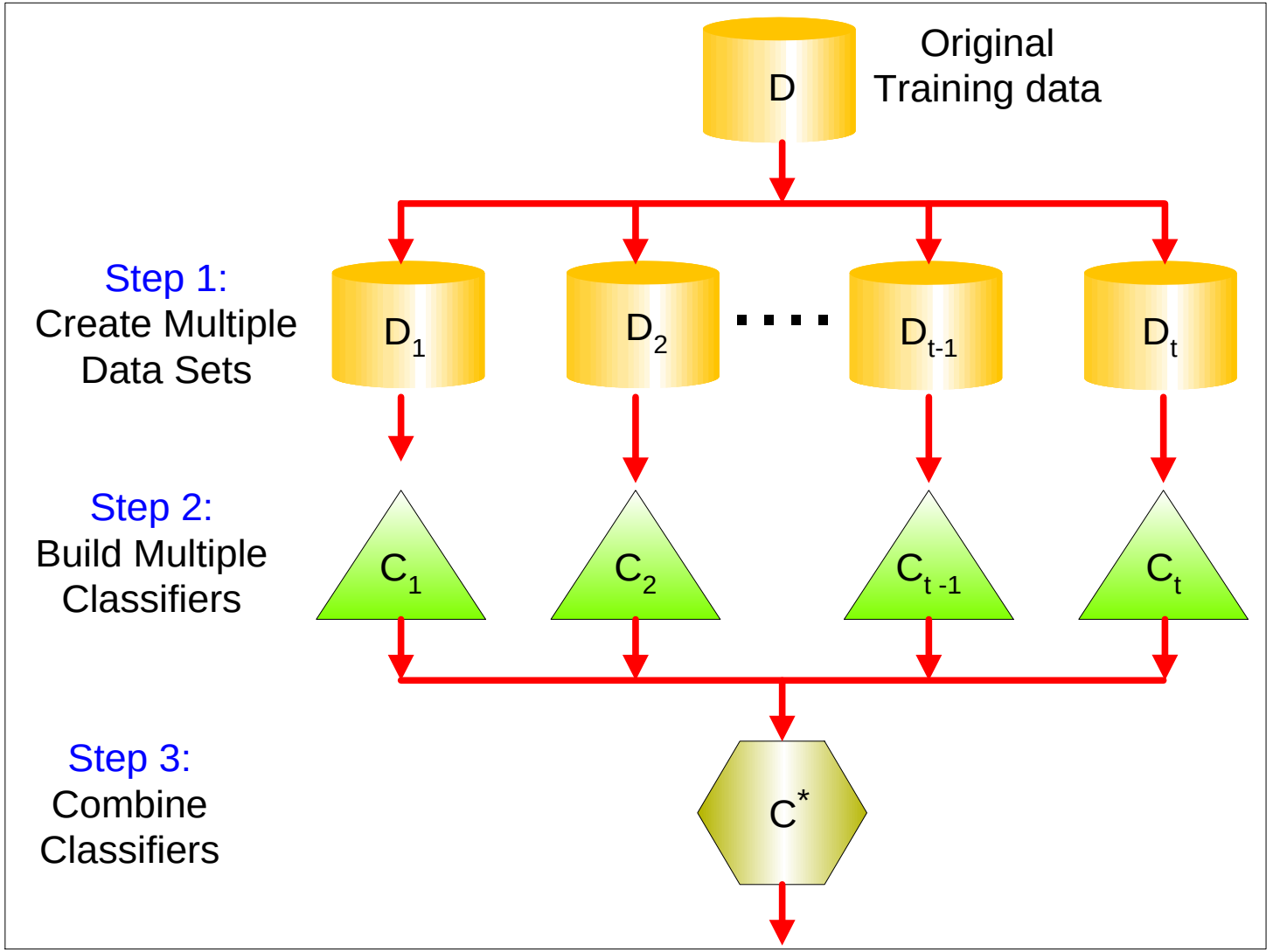
$$\text{Var}\left(y\right) = \text{Var}\left(\sum_j \frac{1}{L} d_j\right) = \frac{1}{L^2} \text{Var}\left(\sum_j d_j\right) = \frac{1}{L^2} L \cdot \text{Var}\left(d_j\right) = \frac{1}{L} \text{Var}\left(d_j\right)$$

Bias does not change, variance decreases by L

- If dependent, error increase with positive correlation

$$\text{Var}\left(y\right) = \frac{1}{L^2} \text{Var}\left(\sum_j d_j\right) = \frac{1}{L^2}\left[\sum_j \text{Var}\left(d_j\right) + 2\sum_j \sum_{i<j} Cov(d_i, d_j)\right]$$

General Idea



Why does it work?

- Suppose there are 25 base classifiers
  - Each classifier has error rate, ε = 0.35
  - Assume classifiers are independent
  - Probability that the ensemble classifier makes a wrong prediction:

$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} = 0.06$$

What is the Main Challenge for Developing Ensemble Models?

- The main challenge is not to obtain highly accurate base models, but rather to obtain base models which make different kinds of errors.
- For example, if ensembles are used for classification, high accuracies can be accomplished if different base models misclassify different training examples, even if the base classifier accuracy is low. Independence between two base classifiers can be assessed in this case by measuring the degree of overlap in training examples they misclassify ($|A \cap B|/|A \cup B|$)—more overlap means less independence between two models.

Error-Correcting Output Codes

- K classes; L base-learners (Dietterich and Bakiri, 1995)
- Code matrix $\mathbf{W}$ codes classes in terms of learners

- One per class
  $L=K$

$$\mathbf{W} = \begin{bmatrix} +1 & -1 & -1 & -1 \\ -1 & +1 & -1 & -1 \\ -1 & -1 & +1 & -1 \\ -1 & -1 & -1 & +1 \end{bmatrix}_{K \times L}$$

- Pairwise
  $L=K(K-1)/2$

$$\mathbf{W} = \begin{bmatrix} +1 & +1 & +1 & 0 & 0 & 0 \\ -1 & 0 & 0 & +1 & +1 & 0 \\ 0 & -1 & 0 & -1 & 0 & +1 \\ 0 & 0 & -1 & 0 & -1 & -1 \end{bmatrix}$$

- Full code $L=2^{(K-1)}-1$

$$\mathbf{W}=\begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & +1 & +1 & +1 & +1 \\ -1 & +1 & +1 & -1 & -1 & +1 & +1 \\ +1 & -1 & +1 & -1 & +1 & -1 & +1 \end{bmatrix}$$

- With reasonable L, find $\mathbf{W}$ such that the Hamming distance btw rows and btw columns are maximized.
- Voting scheme

$$y_i = \sum_{j=1}^{L} w_{ij} d_j$$

and then we choose the class with the highest y_i .

- Subproblems may be more difficult than one-per-K

Bagging

- Bagging, short for bootstrap aggregating
- Use bootstrapping to generate L training sets and train one base-learner with each, using an unstable learning procedure (Breiman, 1996)
- Use voting (Average or median with regression)
- Unstable algorithms profit from bagging

Bagging

- Sampling with replacement

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

- Build classifier on each bootstrap sample

- Each sample has probability $(1 - 1/N)^N$ of being selected

Boosting

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
  - Initially, all *N* records are assigned equal weights
  - Unlike bagging, weights may change at the end of boosting round

Boosting (adaptive boosting)

- Records that are wrongly classified will have their weights increased
- Records that are classified correctly will have their weights decreased

Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

- Example 4 is hard to classify
- Its weight is increased, therefore it is more likely to be chosen again in subsequent rounds

AdaBoost

Generate a sequence of base-learners each focusing on previous one's errors

(Freund and Schapire, 1996)

Training:

For all $\{x^t, r^t\}_{t=1}^N \in \mathcal{X}$, initialize $p_1^t = 1/N$

For all base-learners $j = 1, \ldots, L$

Randomly draw $\mathcal{X}_j$ from $\mathcal{X}$ with probabilities p_j^t

Train d_j using $\mathcal{X}_j$

For each (x^t, r^t), calculate $y_j^t \leftarrow d_j(x^t)$

Calculate error rate: $\epsilon_j \leftarrow \sum_t p_j^t \cdot 1(y_j^t \neq r^t)$

If $\epsilon_j > 1/2$, then $L \leftarrow j - 1$; stop

$\beta_j \leftarrow \epsilon_j / (1 - \epsilon_j)$

For each (x^t, r^t), decrease probabilities if correct:

If $y_j^t = r^t$ $p_{j+1}^t \leftarrow \beta_j p_j^t$ Else $p_{j+1}^t \leftarrow p_j^t$

Normalize probabilities:

$$Z_j \leftarrow \sum_t p_{j+1}^t; \quad p_{j+1}^t \leftarrow p_{j+1}^t / Z_j$$

Testing:

Given x, calculate $d_j(x), j = 1, \ldots, L$

Calculate class outputs, $i = 1, \ldots, K$:

$$y_i = \sum_{j=1}^{L} \left(\log \frac{1}{\beta_j} \right) d_{ji}(x)$$

Basic AdaBoost Loop

D_1= initial dataset with equal weights

FOR i=1 to k DO

Learn new classifier $C_{i;}$

Compute α_j (classifier's importance);

Update example weights;

Create new training set D_{i+1} (using weighted sampling)

END FOR

Construct Ensemble which uses C_i weighted by α_i (i=1,k)

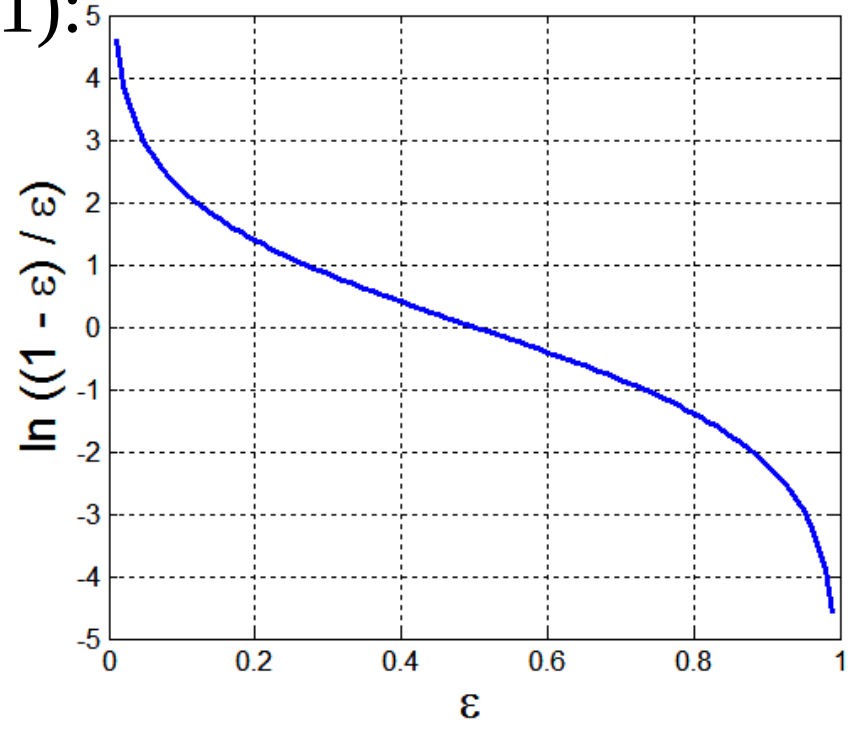
Example: AdaBoost

- Base classifiers: $C_1, C_2, \ldots, C_L$
- Error rate (weights add up to 1):

$$\varepsilon_i = \frac{1}{N}\sum_{j=1}^{N} w_j \delta\left(C_i(x_j) \neq y_j\right)$$

- Importance of a classifier:

$$\alpha_i = \frac{1}{2}\ln\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$$



Example: AdaBoost

Increase weight if misclassification; Increase is proportional to classifiers Importance.

- Weight update:

$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \begin{cases} \exp^{-\alpha_j} & \text{if } C_j(x_i) = y_i \\ \exp^{\alpha_j} & \text{if } C_j(x_i) \neq y_i \end{cases}$$

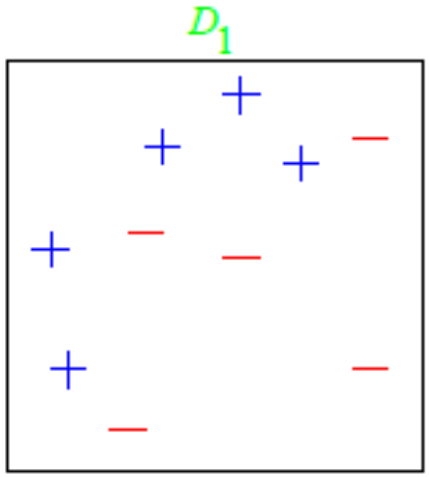
where Z_j is the *normalization* factor

- If any intermediate rounds produce error rate higher than 50%, the weights are reverted back to $1/N$ and the re-sampling procedure is repeated

- Classification (α_j is a classifier's importance for the whole dataset):

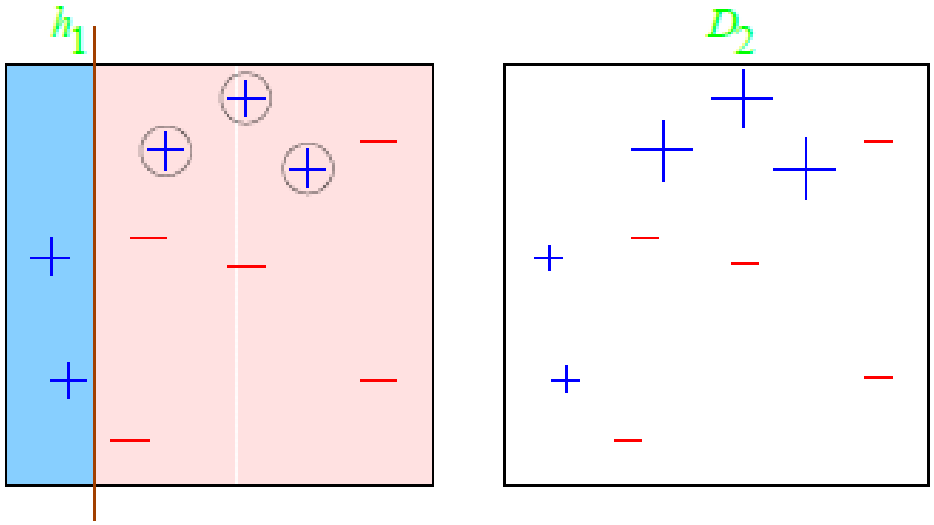
$$C^*(x) = \arg\max_y \sum_{j=1}^{L} \alpha_j \delta\left(C_j(x) = y\right)$$

Toy Example

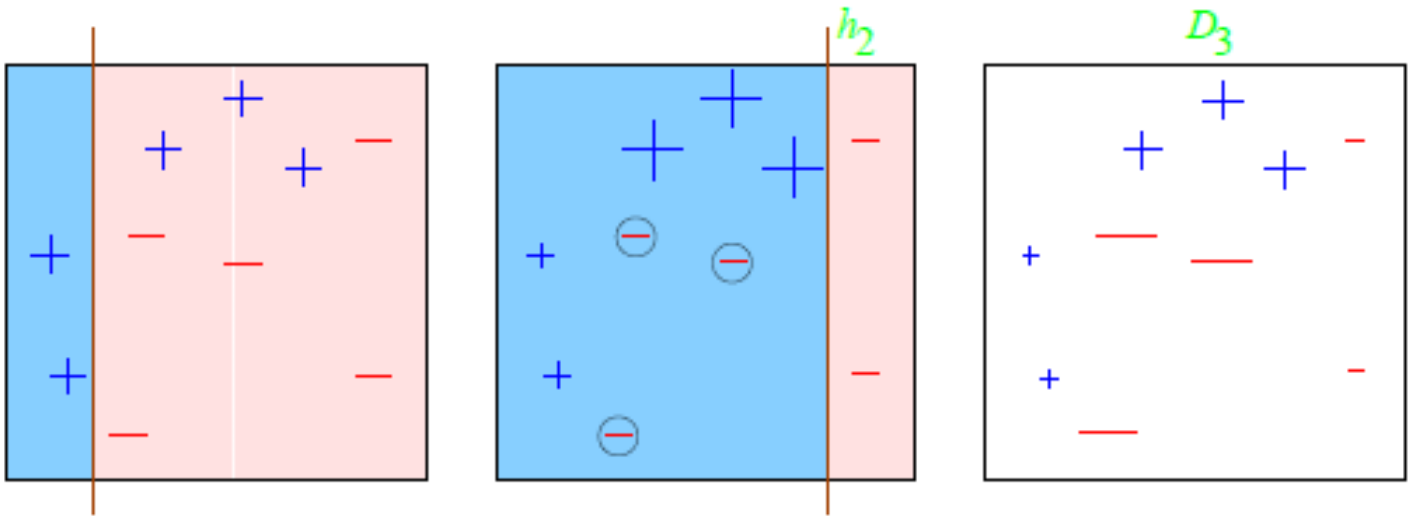


weak classifiers = vertical or horizontal half-planes

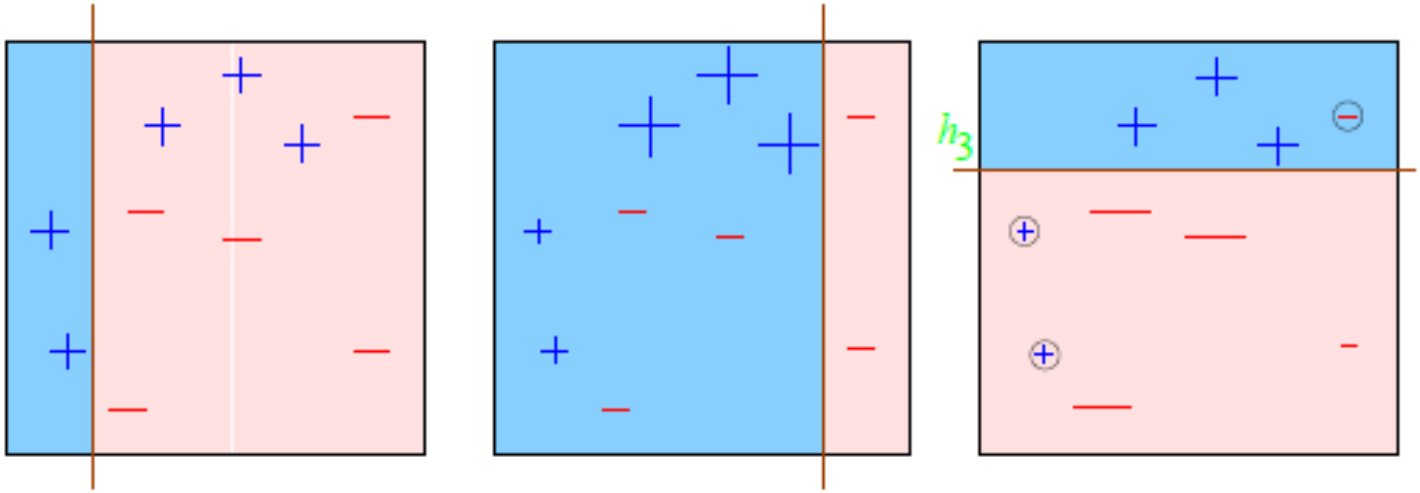
Round 1



Round 2



Round 3

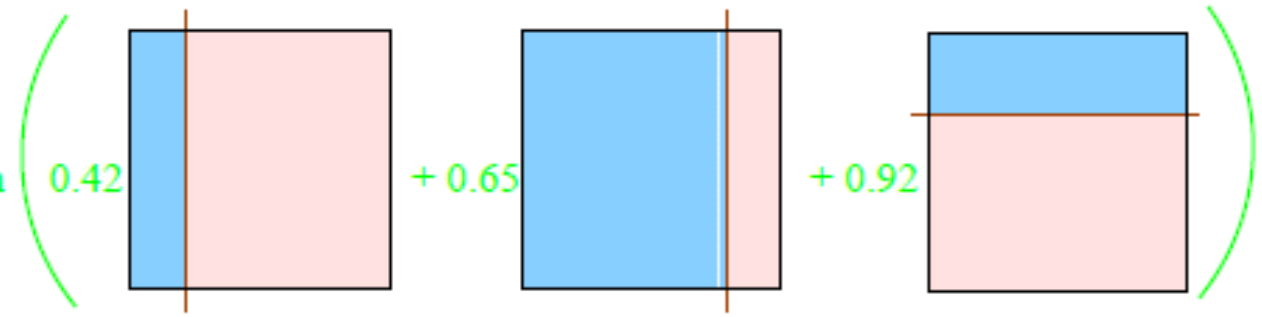


$\varepsilon_3 = 0.14$

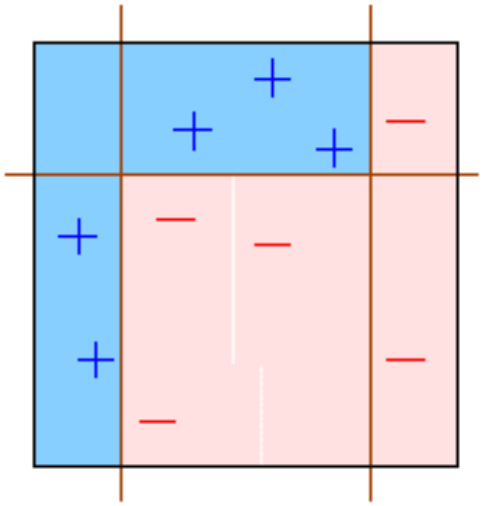
$\alpha_3 = 0.92$

Final Classifier

$H_{\text{final}} = \text{sign}\left(0.42 \; + 0.65 \; + 0.92 \right)$



$=$



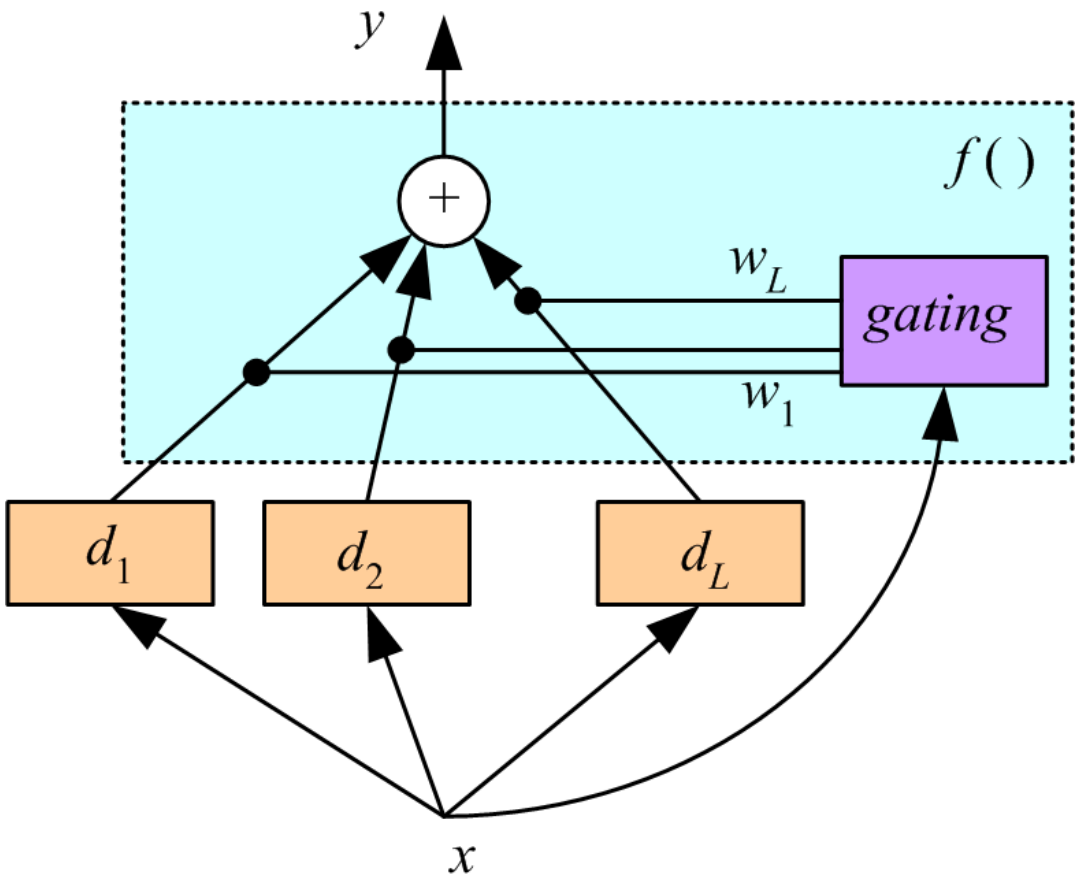
Mixture of Experts

- Voting where weights are input-dependent (gating)

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

(Jacobs et al., 1991)

- Experts or gating can be nonlinear
- Biased but are negatively correlated



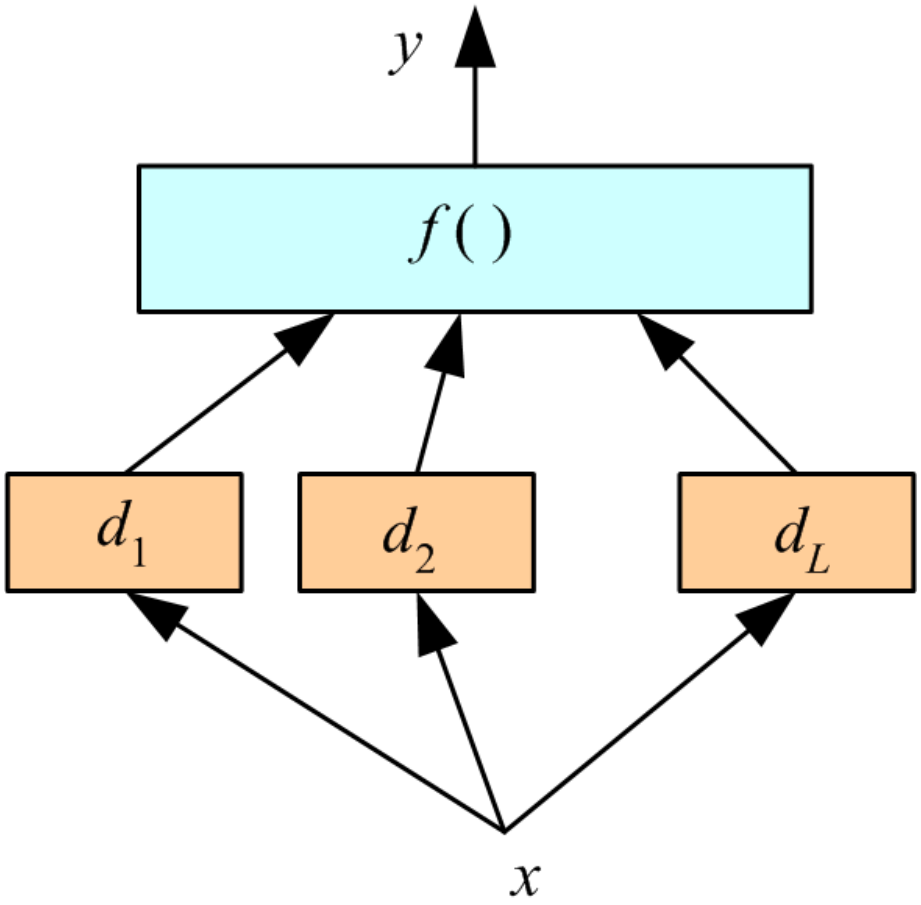
Dynamic Classifier Selection

- There is first a system which takes a test input and estimates the competence of base-classifiers in the vicinity of the input.
- It then picks the most competent to generate output and that output is given as the overall output.
- Woods et al find the *k* nearest training points of the test input, look at the accuracies of the base classifiers on those, and choose the one that performs the best on them.

Stacking

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

$$y = f\left(d_1, d_2, \ldots, d_L \mid \Phi\right)$$



Fine-Tuning an Ensemble

- Given an ensemble of dependent classifiers, do not use it as is, try to get independence

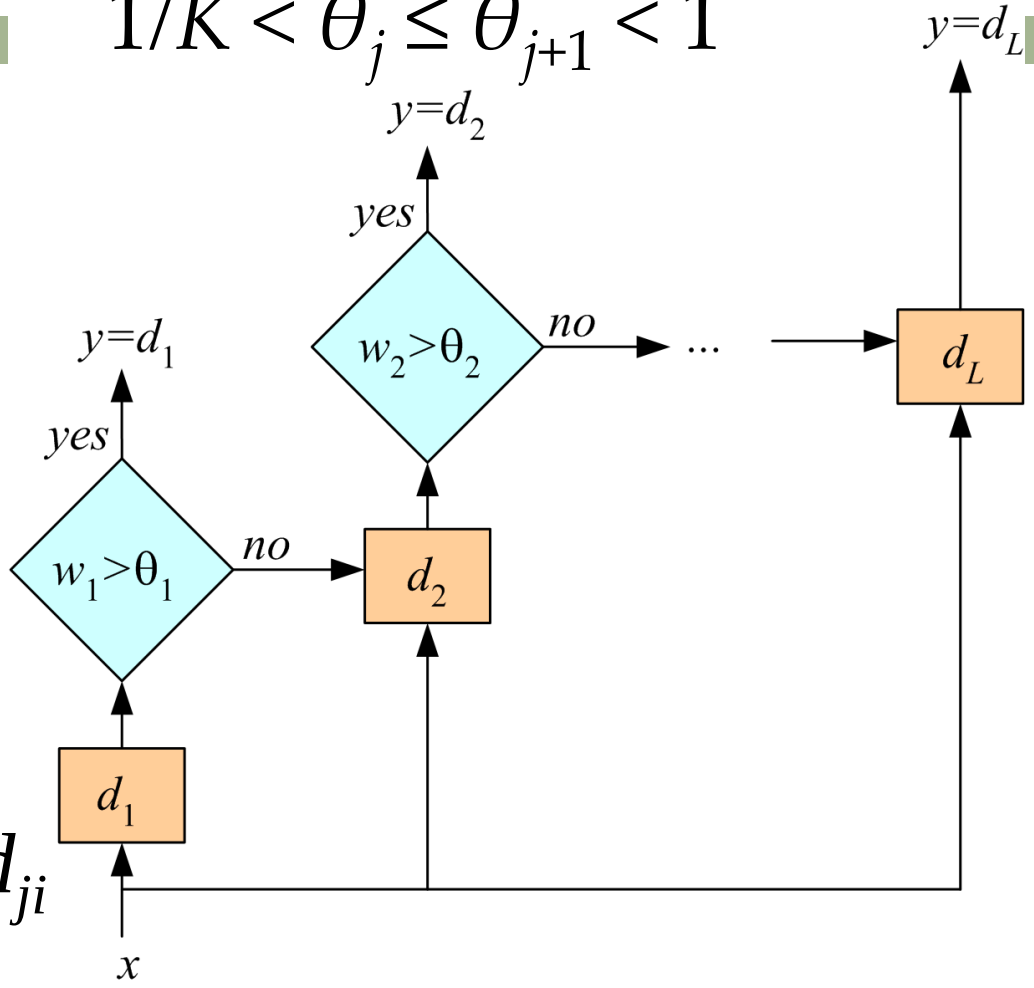
1. Subset selection: Forward (growing)/Backward (pruning) approaches to improve accuracy/diversity/independence
2. Train metaclassifiers: From the output of correlated classifiers, extract new combinations that are uncorrelated. Using PCA, we get "eigenlearners."

- Similar to feature selection vs feature extraction

Cascading

Confidence threshold

$$1/K < \theta_j \leq \theta_{j+1} < 1$$



Use d_j only if preceding ones are not confident

Cascade learners in order of complexity

confidence $w_j \equiv max_i d_{ji}$

$$y_i = d_{ji} \text{ if } w_j > \theta_j \text{ and } \forall k < j,\ w_k < \theta_k$$

Combining Multiple Sources/Views

- Early integration: Concat all features and train a single learner
- Late integration: With each feature set, train one learner, then either use a fixed rule or stacking to combine decisions
- Intermediate integration: With each feature set, calculate a kernel, then use a single SVM with multiple kernels
- Combining features vs decisions vs kernels

Summary Ensemble Learning

- Ensemble approaches use multiple models in their decision making. They frequently accomplish high accuracies, are less likely to over-fit and exhibit a low variance. They have been successfully used in the Netflix contest and for other tasks. However, some research suggest that they are sensitive to noise (http://www.phillong.info/publications/LS10_potential.pdf).
- The key of designing ensembles is diversity and not necessarily high accuracy of the base classifiers: Members of the ensemble should vary in the examples they misclassify. Therefore, most ensemble approaches, such as AdaBoost, seek to promote diversity among the models they combine.

Summary Ensemble Learning

- The trained ensemble represents a single hypothesis. This hypothesis, however, is not necessarily contained within the hypothesis space of the models from which it is built. Thus, ensembles can be shown to have more flexibility in the functions they can represent. Example: http://www.scholarpedia.org/article/Ensemble_learning
- Current research on ensembles centers on: more complex ways to combine models, understanding the convergence behavior of ensemble learning algorithms, parameter learning, understanding over-fitting in ensemble learning, characterization of ensemble models, sensitivity to noise.