



Lecture Slides for

INTRODUCTION TO

# Machine Learning

## 2nd Edition

ETHEM ALPAYDIN

© The MIT Press, 2010

*alpaydin@boun.edu.tr*

*<http://www.cmpe.boun.edu.tr/~ethem/i2ml2e>*

CHAPTER 17:

# Combining Multiple Learners

# Rationale

- No Free Lunch Theorem: 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/Views
  - Training sets
  - Subproblems
- Diversity vs accuracy

# Voting

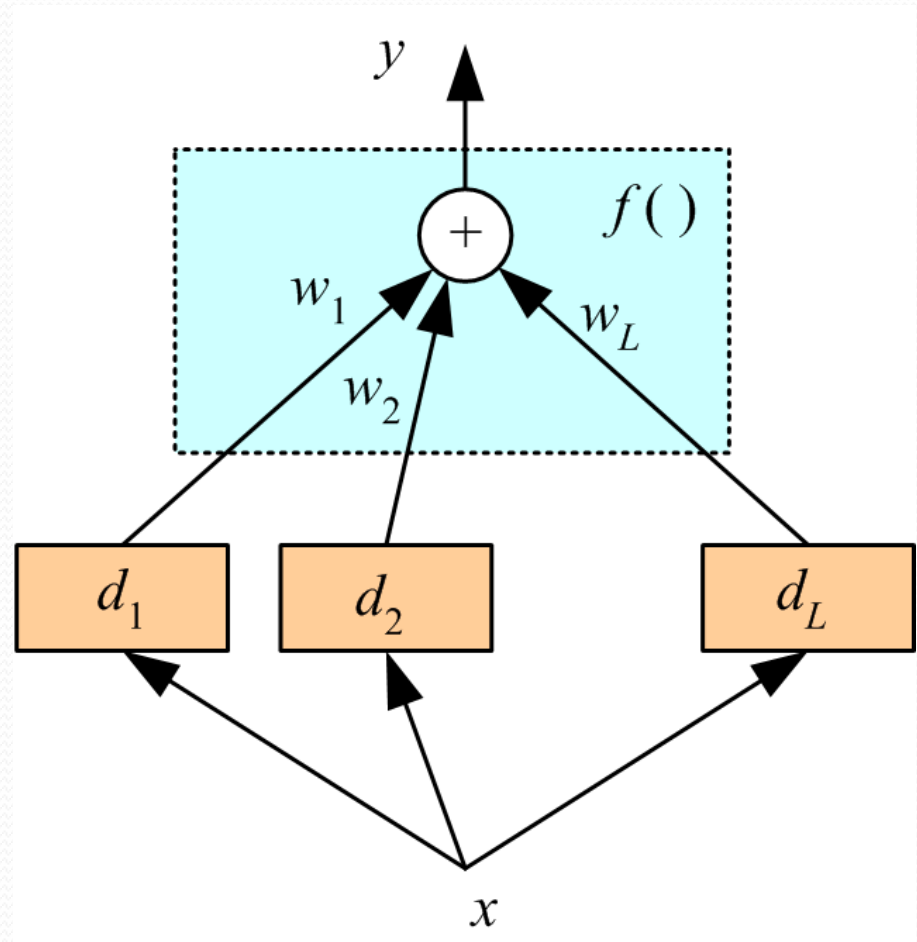
- Linear combination

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

$$w_j \geq 0 \text{ and } \sum_{j=1}^L w_j = 1$$

- Classification

$$y_i = \sum_{j=1}^L w_j d_{ji}$$



- Bayesian perspective:

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

If  $d_j$  are iid

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

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

Bias does not change, variance decreases by  $L$

- If dependent, error increase with positive correlation

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

# Fixed 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	<b>0.5</b>	0.3
Median	0.2	<b>0.5</b>	0.4
Minimum	0.0	<b>0.4</b>	0.2
Maximum	0.4	<b>0.6</b>	0.4
Product	0.0	<b>0.12</b>	0.032

# Error-Correcting Output Codes

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

- One per class

$$L=K$$

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

- 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 columns are maximized.
- Voting scheme

$$y_i = \sum_{j=1}^L w_j d_{ji}$$

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



# Bagging

- Use bootstrapping to generate  $L$  training sets and train one base-learner with each (Breiman, 1996)
- Use voting (Average or median with regression)
- Unstable algorithms profit from bagging

# 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  $j = 1, \dots, L$

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:

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

Testing:

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

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

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

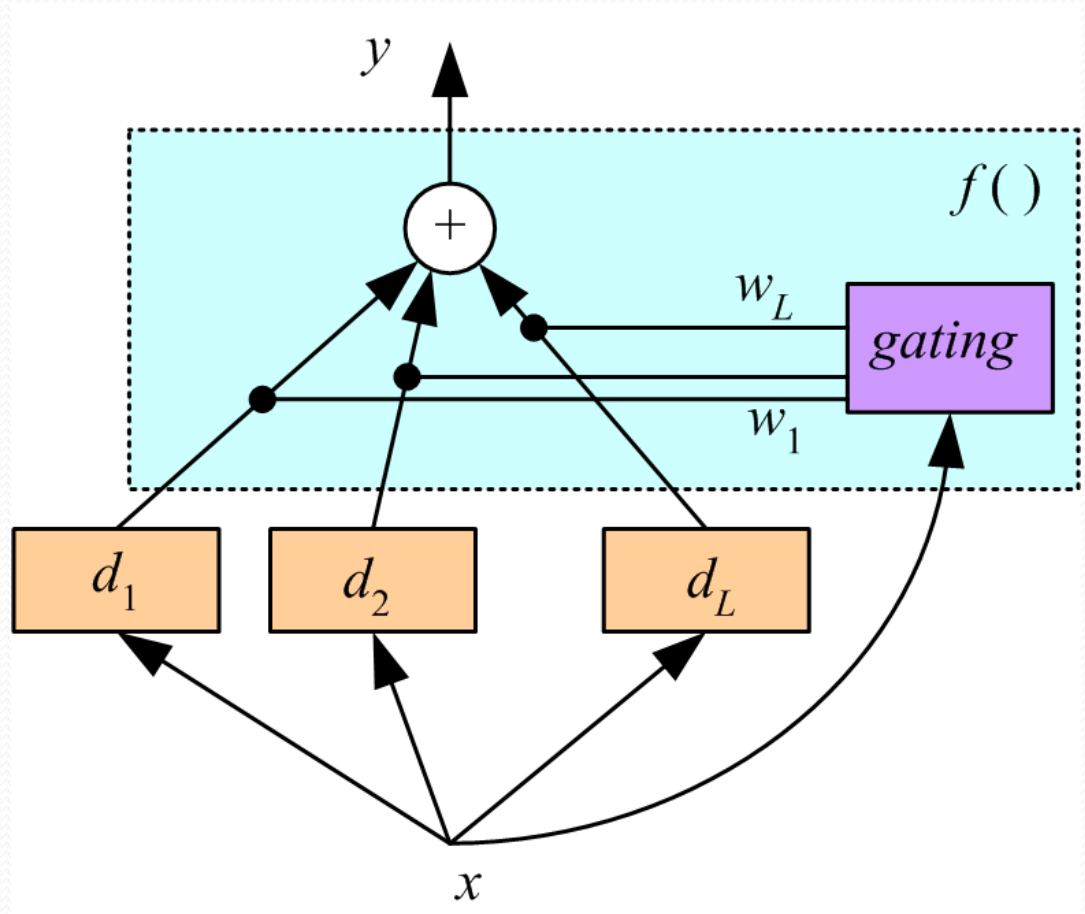
# Mixture of Experts

Voting where weights

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

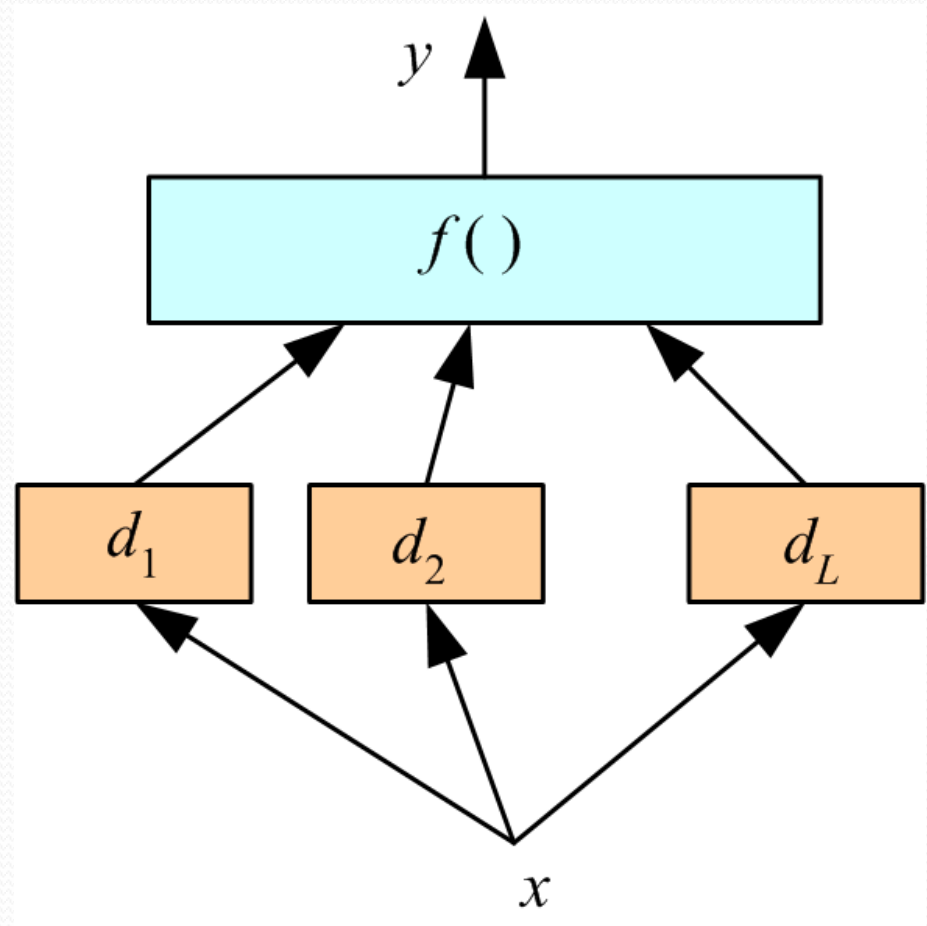
(Jacobs et al., 1991)

Experts or gating  
can be nonlinear



# Stacking

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



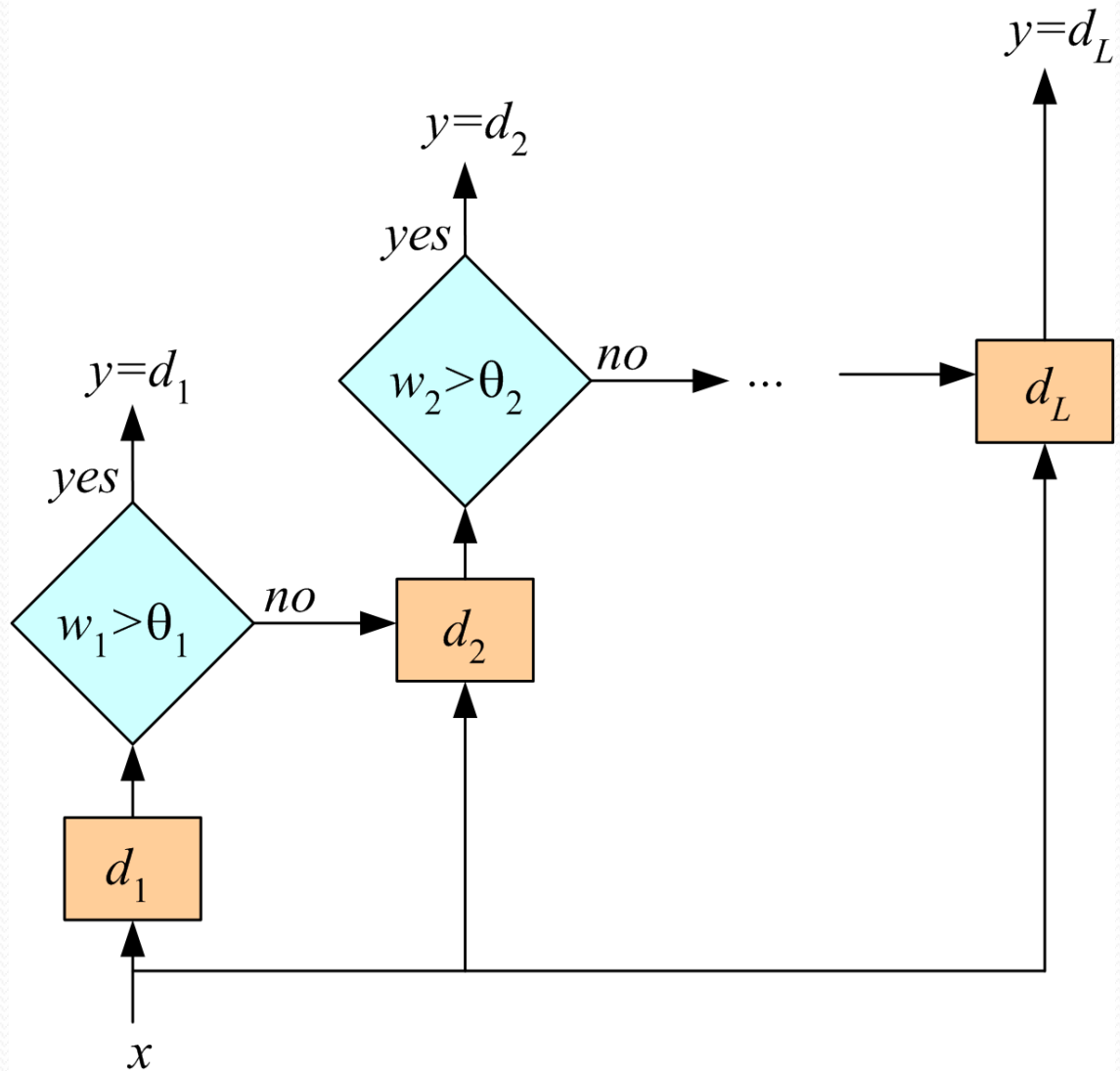
# Fine-Tuning an Ensemble

- 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

Use  $d_j$  only if preceding ones are not confident

Cascade learners in order of complexity



# Combining Multiple Sources

- 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