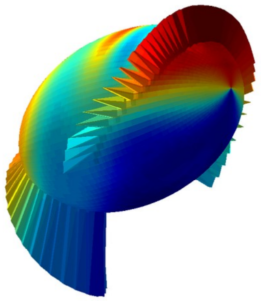
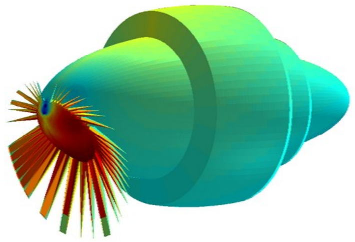
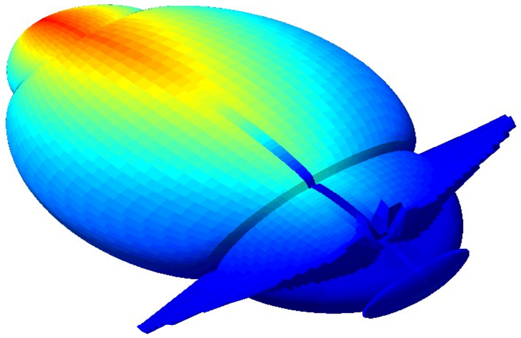
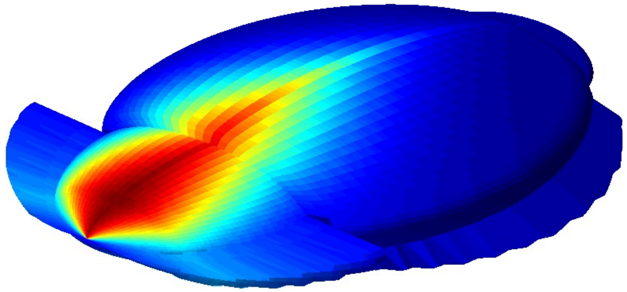




Lecture 10
Ensemble Systems





Three Fundamental Concepts

Redundancy: Strength in numbers
Train several similar systems on all the data and base the conclusion on their joint results.

Why it works: Failure in one system can be covered by others.

Degeneracy: Leveraging diversity
Train several different systems on all the data and base the conclusion on their joint results.

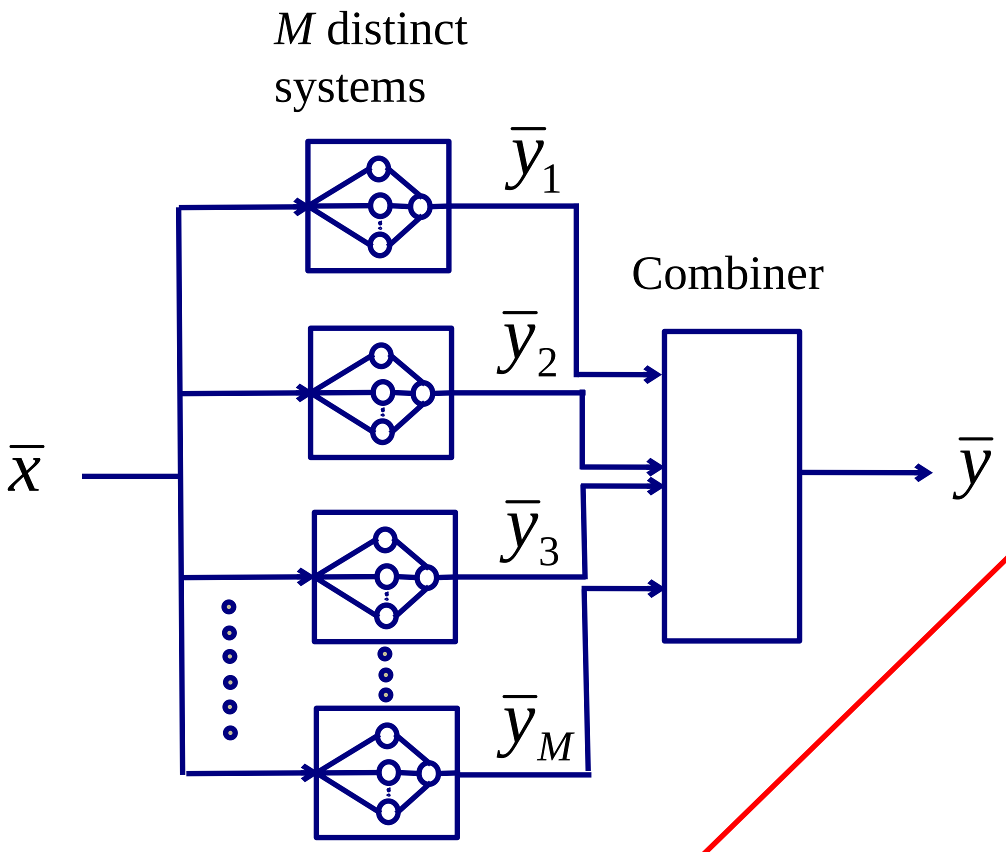
Why it works: Each system has its own strengths and weaknesses → each captures different aspects of the data. Combining them can pool their strengths and cancel their weaknesses.

Modularity: Divide and conquer
Train different networks for distinct subsets of the data (e.g. regions of input space) or different aspects of the task, and use the results of all networks appropriately.

Why it works: The learning problems corresponding to the subsets of data are often simpler than the full learning problem

Static Committee Machines

(use Redundancy or Degeneracy)



-Typical combiners:

- Average
- Majority
- Weighted average

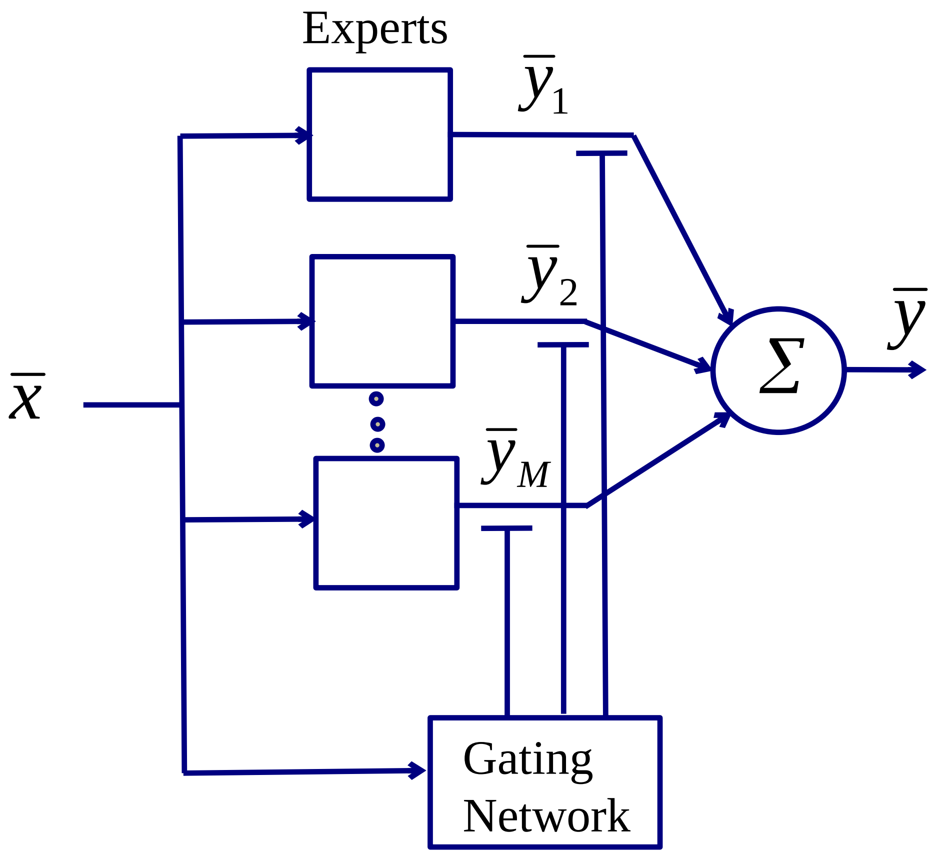
- Systems can have different architectures (indeed, can be different types)

- Individual systems can often be over-trained while preserving generalization of $\bar{y}$

This is degeneracy

Mixture of Experts: Domain Decomposition

(uses Modularity)



- Each expert is trained on a distinct subset of data

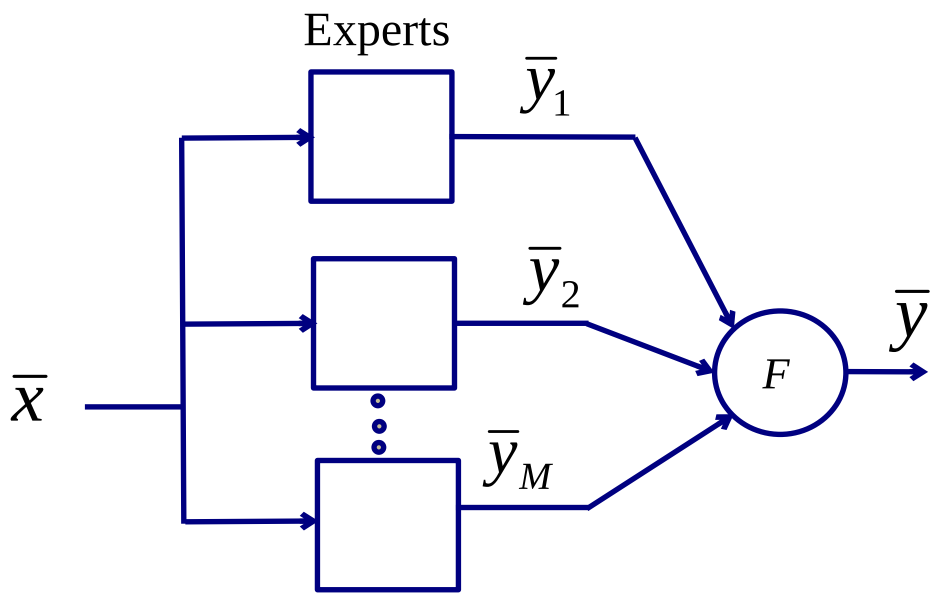
- The gating network ensures that the output from the right expert gets to the summation node.

- The gating can be “soft” if the modularity of expertise is not sharp.

- There is some danger of poor generalization

Mixture of Experts: Problem Decomposition

(uses Modularity)



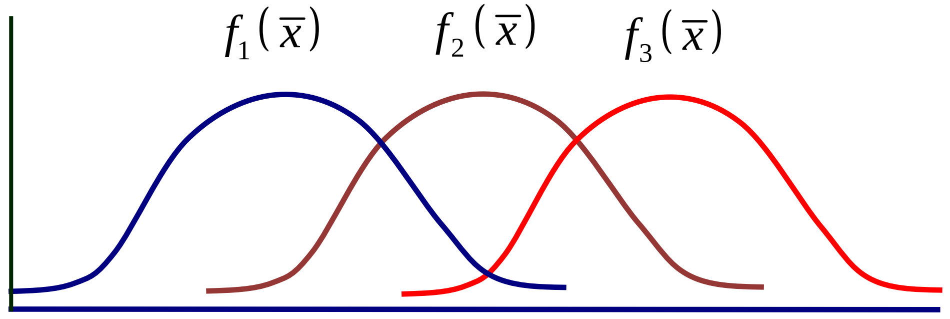
- Each expert is trained on a distinct component of the task.

- The F function combines the outputs of all experts to make a choice.

e.g., each expert classifies data into Class *j* yes/no

Example of Soft Gating

- Select training data for each expert q according to a localized Gaussian distribution $f_q(\bar{x})$



Input Space

- For a new input pattern $\bar{x}$ let each expert q produce an output,

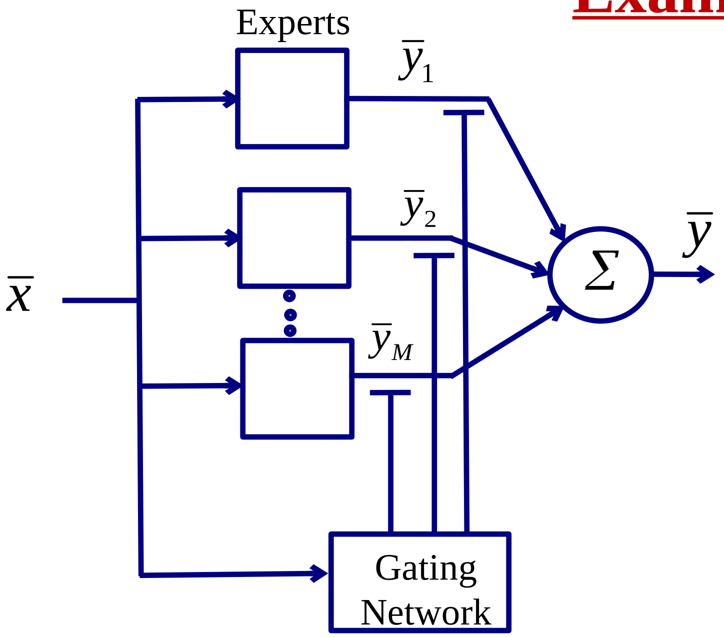
- Combine $\bar{y}_q$ s using:

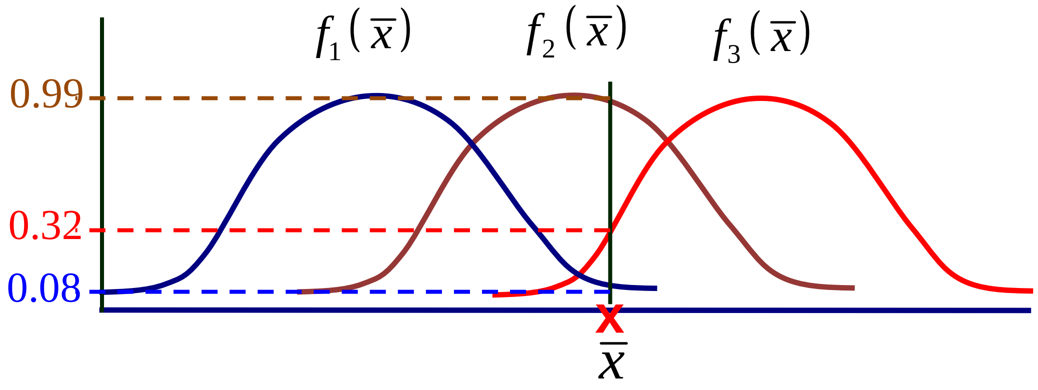
$$y = \sum_q \frac{f_q(\bar{x})}{\bar{f}(\bar{x})} y_q$$

where $\bar{f}(\bar{x}) = \sum_q f_q(\bar{x})$

In principle, the distributions $f_q(\bar{x})$ could be chosen adaptively, and do not have to be unimodal.

Example of Soft Gating





$$\bar{f}(\bar{x}) = \sum_q f_q(\bar{x}) = 0.08 + 0.99 + 0.32 = 1.39$$

$$\bar{y} = \sum_q \frac{f_q(\bar{x})}{\bar{f}(\bar{x})} y_q = \frac{1}{1.39}\left(0.08 y_1 + 0.99 y_2 + 0.32 y_3\right)$$

Bagging (Bootstrap Aggregation)

L. Breiman (1996) Bagging predictors, *Machine Learning* 24 (2): 123–140.

- Consider a training dataset, ***D***, with N samples.
- Uniformly sample this set ***with replacement*** to get m datasets, $\boldsymbol{D}_j$, $j = 1, 2, \ldots, m$, of N_1 samples each (N_1 could be $\geq N$)
- Train separate classifiers on each $\boldsymbol{D}_j$.
- For any new data point, combine the result of all classifiers (e.g., majority, plurality, average + threshold, etc)

Note that:

- Because of sampling with replacement, each dataset may include multiple copies of the same data point in ***D***.
- Not all data points in ***D*** are necessarily included in the m datasets.

➕ Bagging can improve performance and reduce variance of error (e.g., neural networks, decision trees).

➖ Sometimes, bagging can slightly degrade performance (e.g., k-NN).

Strong and Weak Learning

Consider a set, ***A***, for all objects of interest (e.g., faces).

A concept, *C*, is defined as a Boolean function on this space:

$$C : \boldsymbol{A} \rightarrow \{0, 1\}$$

Samples are drawn from ***A*** with a probability distribution *D*

Strong Learner: A system that can learn the unknown concept *C* with an arbitrarily small error, ε, with probability 1 - δ over all distributions *D*.

Weak Learner: A system that can learn the unknown concept *C* with an error slightly less than ½ with probability 1 - δ over all distributions *D*.

Q: Can we get strong learning from weak learners?
A: Yes, through boosting.

Boosting

Boosting is a structured way to use mixtures of experts.

Basic Idea:

- Use a set of three weak experts
- Choose training sets for each expert by filtering sequentially,
  Expert 2's training set depends on the performance of Expert 1,
  and Expert 3's set on the performance of Experts 1 and 2.
- Combine the decision of the three experts to arrive at the final decision.

The goal is to create experts with ***complementary*** but ***not mutually exclusive*** expertise, which is an example of soft modularity.

The experts may be very different – even of different types – which is an example of degeneracy.

Classical Boosting

1. Consider a 2-class problem, N data points, and three learners: First, Second, Third.

2. Train First on a dataset of $N_1 < N$ samples (e.g. randomly chosen, or first N_1).

3. Use First to filter the training set for Second as follows:
   - Flip a fair coin.
   - If heads: Have First try out a random data point. If it classifies correctly, discard the data point; else add it to the training set for Second.
   - If tails: Try a random data point on First. If classified correctly, add it to the training set for Second; else discard it.
   - Continue this process until a training set of size N_1 is obtained for Second.

4. Train Second on the generated training set of N_1 points.

5. Use First and Second to filter the training set for Third as follows:
   - Choose a random data point and pass it through First and Second.
   - If they agree, discard the point; else add it to the training set for Third.
   - Continue this process until N_1 points are obtained for training Third.

6. Train Third using the generated training set.

7. For any new data point, try First and Second. If they agree, accept that answer; else accept the answer given by Third.

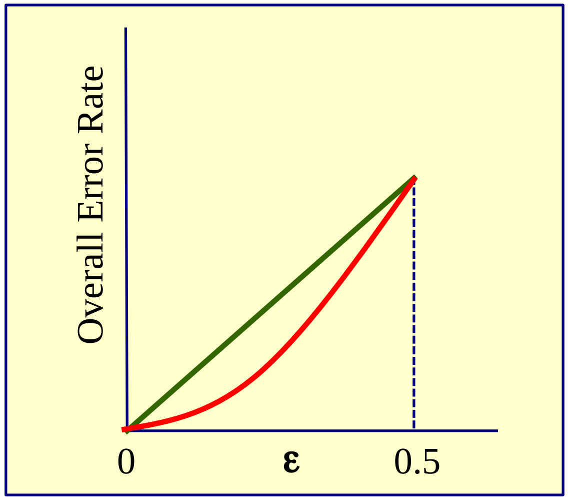
- Second is trained on a set for which the performance of First is ½ by design (i.e., First knows nothing about it).
- Third is trained on data points that need a tie-breaker.
- If selecting the N_1 points for Second required going through N_2 points and selecting N_1 points for Third required going through N_3 points:

  Total number of data points used = $N_1 + N_2 + N_3$

  but training cost is only for $3N_1$ points.

- If each expert has an error rate of $\varepsilon < ½$, Schapire (1990) proved that the error of the committee is bounded by: $g(\varepsilon) = 3\varepsilon^2 - 2\varepsilon^3$
- The error could be brought down further by using boosting recursively

Committees of weak learners can become arbitrarily strong



AdaBoost

AdaBoost is an Adaptive Boosting algorithm

- Given a data set $(\mathbf{x}^k, y^k)$ of N points for a 2-class problem and a weak learner:
- Set the initial sampling distribution to $D_1(k) = 1/N \quad \forall\, k = 1, \dots, N$

All points equally likely

- For $j = 1$ to T
  - Use distribution D_j to train the learner and get weak hypothesis (rule) F_j
  - Calculate error ε_j of F_j using D_j

$D_j(k)$ is the probability of x^k being chosen in distribution D_j

Total probability weight of all the data points on which F_j is wrong

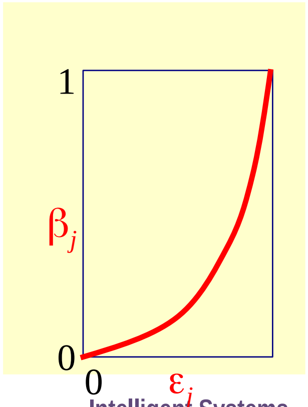
$$\varepsilon_j = \sum_{k:F_j(x^k)\neq y^k} D_j(k)$$

  - If $\varepsilon_j > \tfrac{1}{2}$, set $j = j-1$ and abort the loop (repeat from previous step)

$$\beta_j = \frac{\varepsilon_j}{1-\varepsilon_j}$$

Note: This is a scaled error

Note: $\beta_j = 1$ at $\varepsilon_j = 0.5$, and decreases convexly with decreasing ε_j



  - Set

- Set sampling distribution D_{j+1} as:

$$D_{j+1}(k) = \frac{D_j(k)}{Z_{j+1}} \times \begin{cases} \beta_j & \text{if } x^k \text{ was correctly classified by } F_j \\ 1 & \text{else} \end{cases}$$

Normalizing factor

Note: So if a data-point was correctly classified by the previous learner, the current one gives it lower weight. Otherwise, the full weight.

This also means that the weight of each data-point comes to depend on how many previous learners classified it correctly: If many learners did so, it gets much less weight; if only a few did, it has more weight but still less than full; if none did, it gets full weight.

- Given test point *x*, choose the class *y* that maximizes $\sum_{j:F_j(x)=y} \log \frac{1}{\beta_j}$

Note: log $1/\beta_j$ is a measure of accuracy, and the sum is the total accuracy of all learners voting for class *y*. Thus, we're choosing that *y* for which the most accurate learners gave the most vote.

- It can be shown that if all the hypotheses have $\varepsilon_j < ½$, the error of the overall system drops exponentially fast with T.

- This does not say anything about generalization, but generalization performance with AdaBoost is often excellent.

The above discussion of boosting and AdaBoost is based on:

Haykin, S. (1999) Neural Networks: A Comprehensive Foundation. Prentice-Hall. pp 357-363.

The AdaBoost Algorithm given here is AdaBoost.M1 from Freund & Schapire (1996)