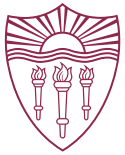
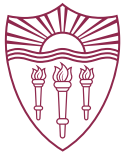


# ENSEMBLE METHODS AND RANDOM FORESTS

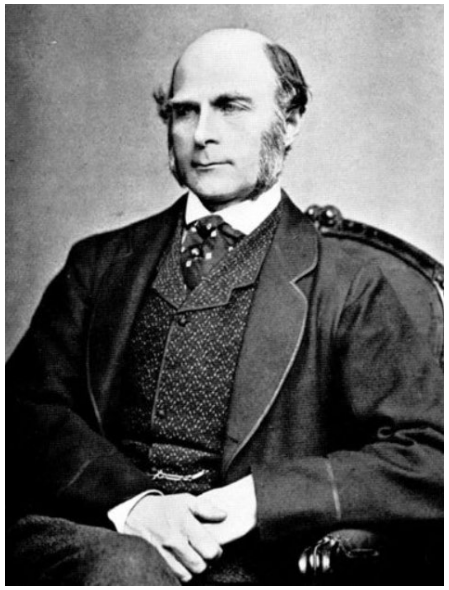
Kristina Lerman
USC Information Sciences Institute
DSCI 552 – Spring 2021
March 24, 2021

# Topics this week

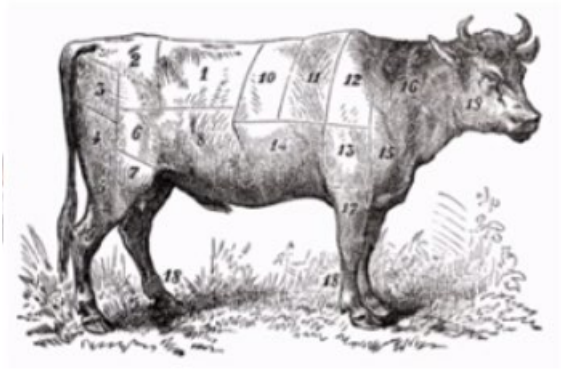


- Decision trees
  - Classification trees
  - Regression trees
- Ensemble methods
  - Boosting and bagging
  - Random forest

# Wisdom of the crowds

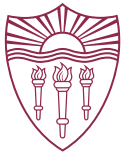


Sir Francis Galton, 1906



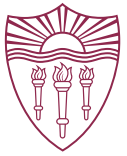
*average of 800 guesses = 1,197*
*actual weight of the ox = 1,198*

# Intuition



- No-Free-Lunch Theorem
  - There is no algorithm that is always the most accurate
- Objective
  - Generate a group of (less accurate) base-learners which when combined has higher accuracy
- Different learners use different
  - Algorithms
  - Hyperparameters
  - Representations /Modalities/Views
  - Training sets
  - Subproblems
- Diversity vs Accuracy

# Rationale for Multi Classifier Systems

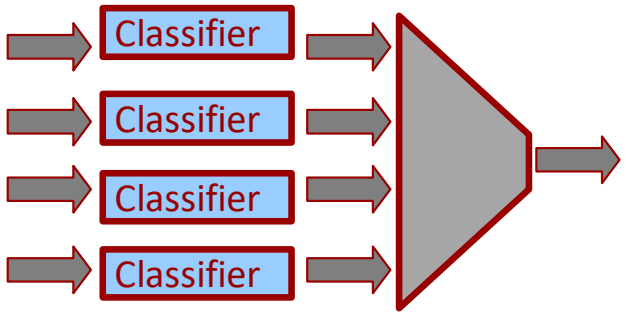


- Even through validation and parameter tuning you will not achieve a perfectly working classifier and all of the classifiers will converge to a different optimum
- This might be a good thing… Maybe a combination of *base learners* will outperform each single base learner and it might be possible to learn from mistakes and correct for them
- Through increase in computational power and distributed learning approaches it became possible to learn many classifiers
- But that alone doesn't help us solve the problem and two basic questions need to be answered (often experimentally):
  - How do we generate base learners that **complement** each other?
  - How do we **combine** the outputs of base learners for maximum accuracy?

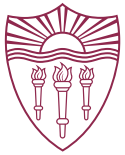
http://www.cmpe.boun.edu.tr/~ethem/i2ml2e/index.html

# Challenges

- Two basic questions need to be answered (often experimentally):
  - How do we **combine** the outputs of base learners for maximum accuracy?
  - How do we generate base learners that **complement** each other?



# Diversity



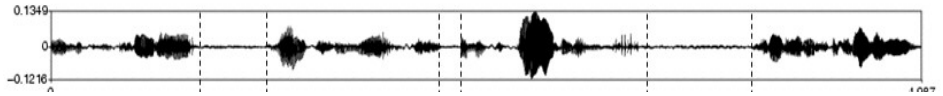
- *Analogy:* In order to learn from someone else it is important that the person does not have exactly the same knowledge as you do
- The same holds for classifiers: If two classifiers always make the same mistakes there is no way a combination of the two can improve performance
  - At the same time the classifiers or learners should be accurate in their own domain of expertise
- We have two goals
  - **maximize individual accuracies and**
  - **diversity between learners**

# Example: Audio-Visual Speech Recognition



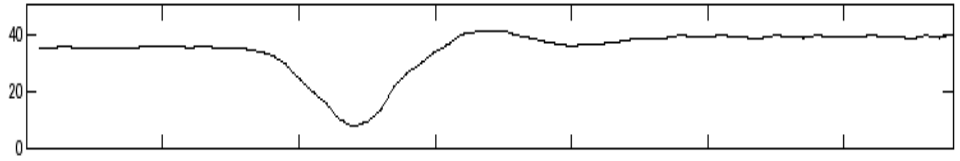
**Diversity?**

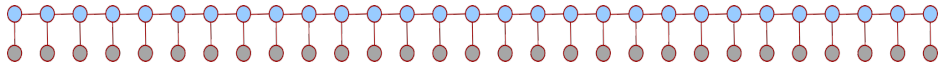
**Audio**




**Video and Lip tracking**

**Lip width**



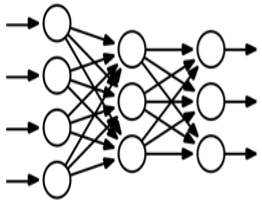



**Words**

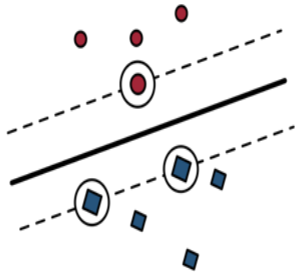
*... on the computer* | *way too much* | *and it's bothering her* | *so she's telling ...*

# Achieving Diversity

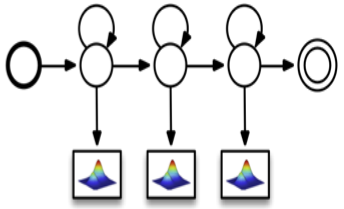
- Different Algorithms



Multi layer perceptron



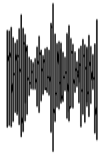
Support vector machine



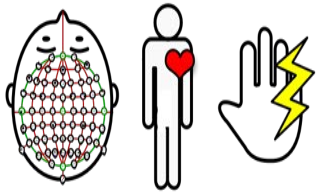
Hidden Markov model

- Different Input Representations

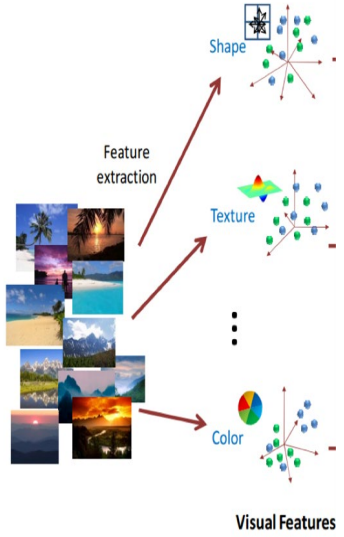
Video input



Audio waveform



Physiological data

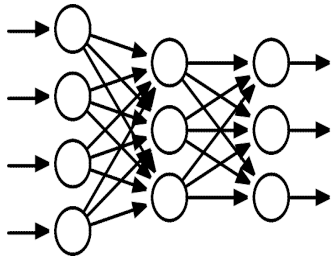



Visual Features

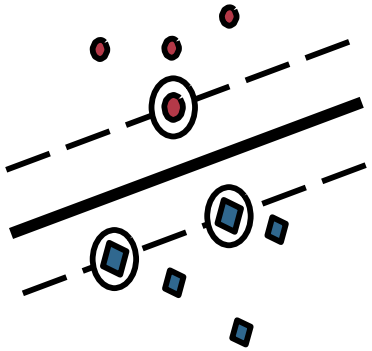
- Different Input Representations (Multi-view)
- Different Training Sets (boosting and bagging)

# Different Algorithms

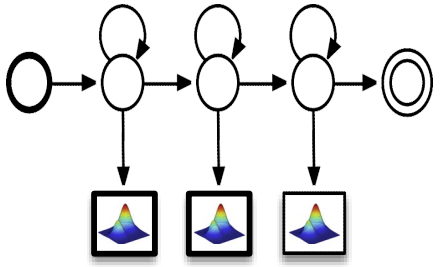
- Through the use of different base learners such as SVM, MLP, HMM we can enforce diversity to some degree
- E.g. an HMM would take temporal dynamics into account for the classification whereas the base models for SVM or MLP would not.



Multi layer perceptron

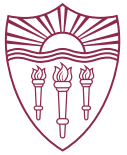


Support vector machine



Hidden Markov model

# Different Hyper-Parameters

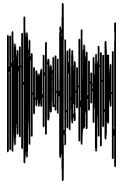


- We can use the same algorithms (e.g. multiple MLP) but use different hyper parameters for training.
- For example, different number of layers in the MLP or different number of neurons; different selection of kernel for SVM etc.
- For classifiers that utilize gradient descent methods even the initialization of weights that define the starting location of the learning algorithms can somewhat influence the outcome of the classifier. This diversity is likely not too strong though.

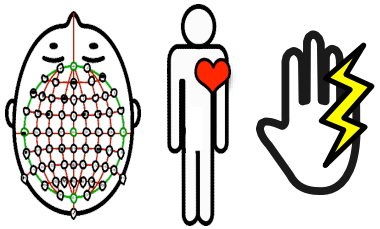
# Different Input Representations

- Separate base learners may make use of different representations of the same input “object” or “event”.
- Information of different representations from different sources/sensors/modalities often lead to diversity in classifiers allowing for better identification.
  - In human emotion recognition or user state recognition many modalities contain information. We typically fuse audio, video, and sometimes physiology. This paradigm is used as **“sensor fusion” or “multimodal fusion”**

Video input



Audio waveform



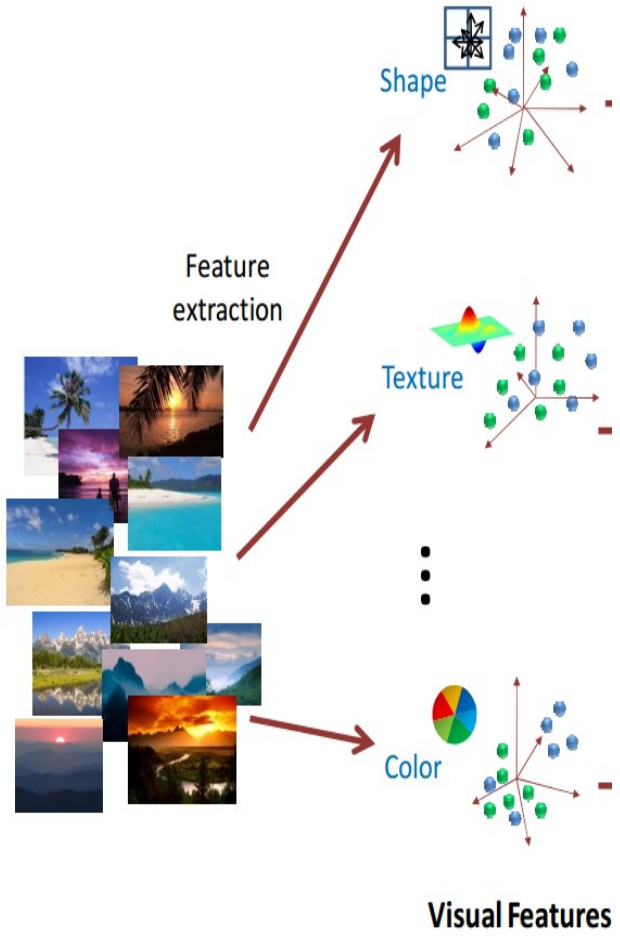
Physiological data

# Different Input Representations

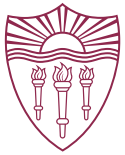
- Sometimes it is possible to extract multiple views from the same data (e.g. shape, texture, color, and maybe keywords)

  → multi-view learning

- In the random subspace method we can also randomly choose certain features to be input to one classifier and not the other etc.

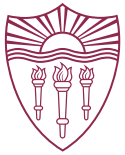


# Different Training Sets



- If sufficient data are available it is possible to train classifiers on different subsets of the data, which leads to different "views" on the same type of data
- This can be done through randomly choosing different subsets (named ***bagging***)
- Or the learners can be trained serially so that instances on which the previous classifiers are not accurate are given more *emphasis* in the training for later base learners (named ***boosting*** or ***cascading***)

# Diversity vs. Accuracy



- Base learners are chosen for their simplicity or expertise on a certain subspace on the data
- Each learner does not need to be optimized for the entire input space. However, we need them to be **diverse**
- For example, if we have one classifier that is 80% accurate on the entire input space, a second classifier only needs to be accurate on the remaining 20% that the first classifier misclassifies
- In some cases (e.g., voting) we will require all learners to be accurate on the entire dataset

# Model Combination Schemes

- Multiexpert combination
  - In the global approach **learner/classifier fusion** the learners work in parallel on the entire set and their outputs are fused (e.g., ***voting*** or ***stacking***)
  - In the local approach **learners or classifiers are selected;** a gating or selection methods is to be employed. It is checked which learner is responsible for a certain input
- Multistage combination
  - Here a serial approach is followed; in particular, one classifier is only trained on instances for which the previous classifier is not accurate enough for; an example is **cascading or boosting**

# Fusion Schema: Voting

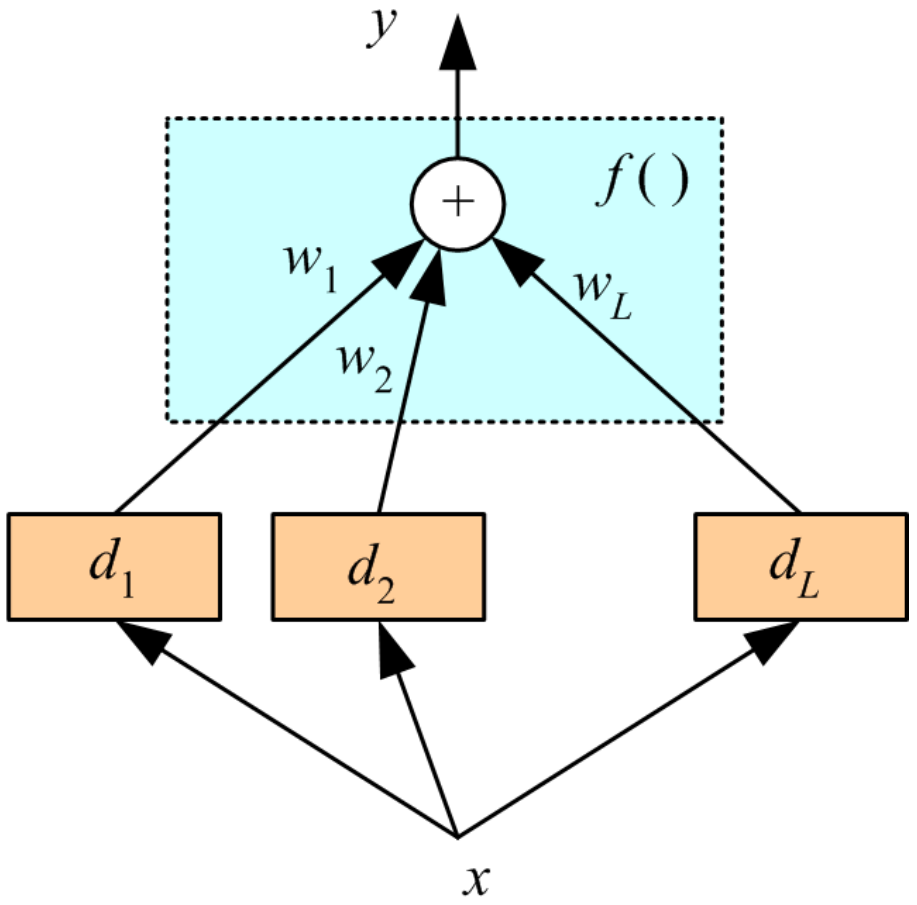
- Linear combination

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

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

- Classification

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



- Bayesian perspective:

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

- 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

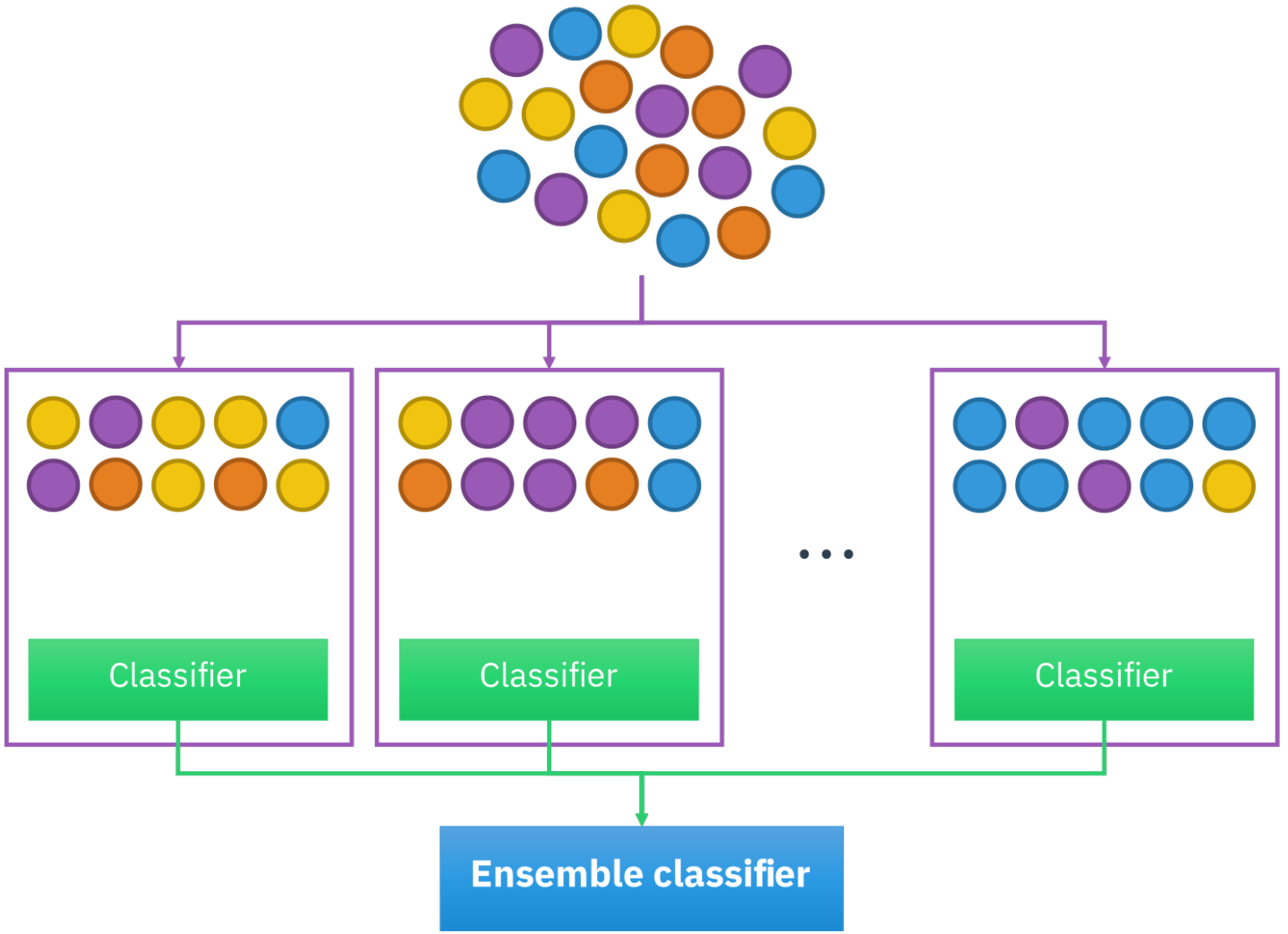
$$\text{Var}(y) = \frac{1}{L^2} \text{Var}\left(\sum_j d_j\right) = \frac{1}{L^2}\left[\sum_j \text{Var}(d_j) + 2\sum_j \sum_{i<j} Cov(d_i, d_j)\right]$$

# 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 | **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 |

# Bagging (=bootstrap aggregation)




Original Data

**Bootstrapping**

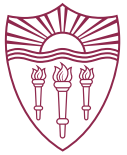
(sample data with replacement)

**Aggregating**

Use voting (Average or median with regression)

**Bagging**

# Boosting



- Select the subset of the training data for the classifiers manually
- Choose a subset for a learner that another learner is performing poorly on
- There are many variants of boosting and the originally proposed variants require a lot of training data to succeed
- **AdaBoost** (adaptive boosting) reduces the issue of requiring huge amounts of data; in principle in AdaBoost the distribution with which a training sample is selected in each classifier is changed based on the error other classifiers make; samples that are misclassified by a classifier are sampled more often to improve performance on them.

# 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:

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

Testing:

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

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

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

# Mixture of Experts

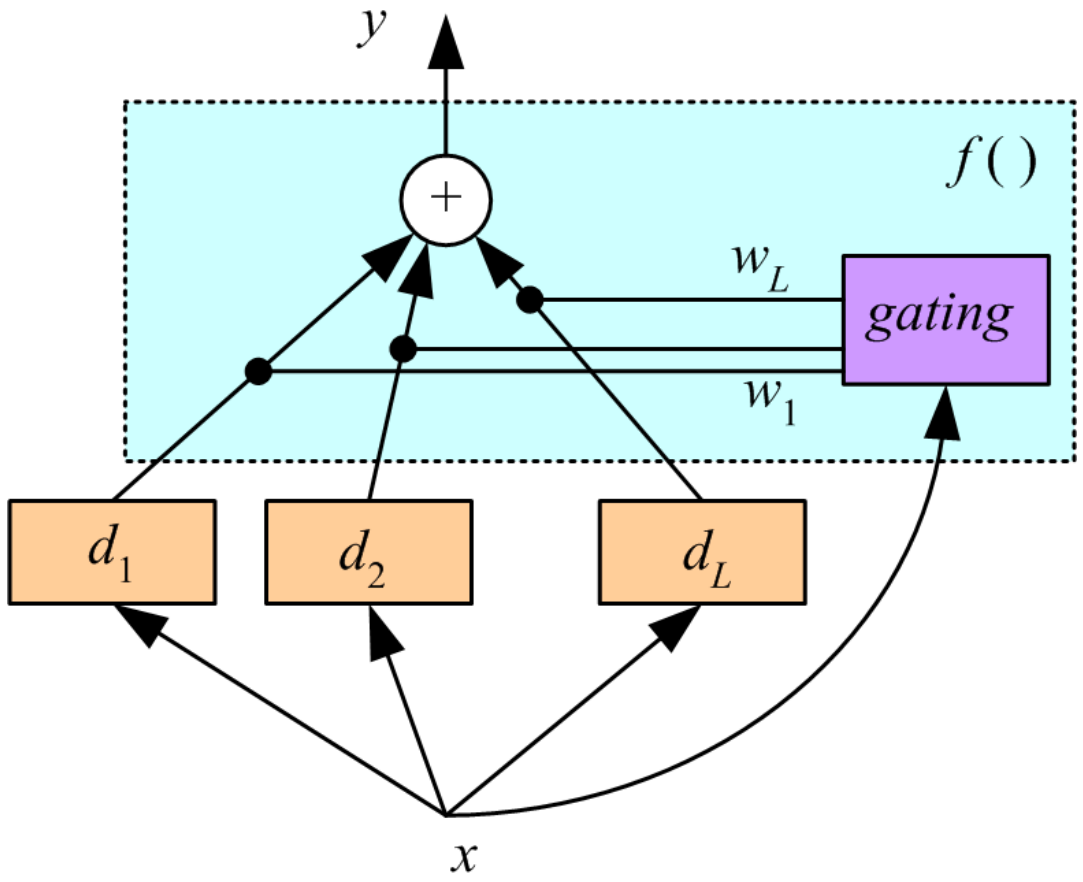
Voting where weights are input-dependent (gating)

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

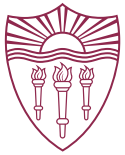
(Jacobs et al., 1991)

Experts or gating

can be nonlinear

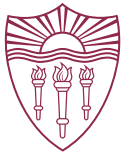


# Fine-Tuning an Ensemble

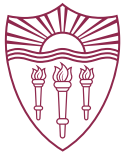


- Given an ensemble of dependent classifiers, do not use it as is, try to get independence
  1. Subset selection: Forward (growing)/Backward (pruning) approaches to improve accuracy/diversity/independence
  2. Train metaclassifiers: From the output of correlated classifiers, extract new combinations that are uncorrelated. Using PCA, we get “eigenlearners.”
- Similar to feature selection vs feature extraction

# Fine Tuning Ensembles

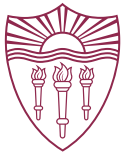


- There is no guarantee that using multiple base learners will improve overall classification results.
- Base learners can be removed from an ensemble if they do not add to the accuracy or are highly correlated to another learner (i.e. low diversity).
- Ensemble selection can follow several different strategies (e.g. brute force … test all combinations; or greedy … forward or backward selection of classifiers)
- *Forward selection:* start with a single best classifier and add the one that adds the most until no improvement is reached.
- *Backward selection:* start with all classifiers and remove the one that improves the performance the most; continue until no improvement is reached



# RANDOM FOREST = ENSEMBLE OF DECISION TREES

# Random Forests

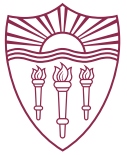


As in bagging, we build a number of decision trees on bootstrapped training samples each time a split in a tree is considered, a random sample of *m* predictors is chosen as split candidates from the full set of p predictors.

- Take a random sample (with replacement) of the training data

- To avoid creating highly correlated trees, choose a random sample (without replacement) of predictors (features).
  - De-correlation increases accuracy

# Random Forests Algorithm



For b = 1 to B:

(a) Draw a bootstrap sample Z∗ of size *N* from the training data.

(b) Grow a random-forest tree to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size $\boldsymbol{n_{min}}$ is reached.

i. Select ***m*** variables at random from the *p* variables.

ii. Pick the best variable/split-point among the *m*.

iii. Split the node into two daughter nodes.

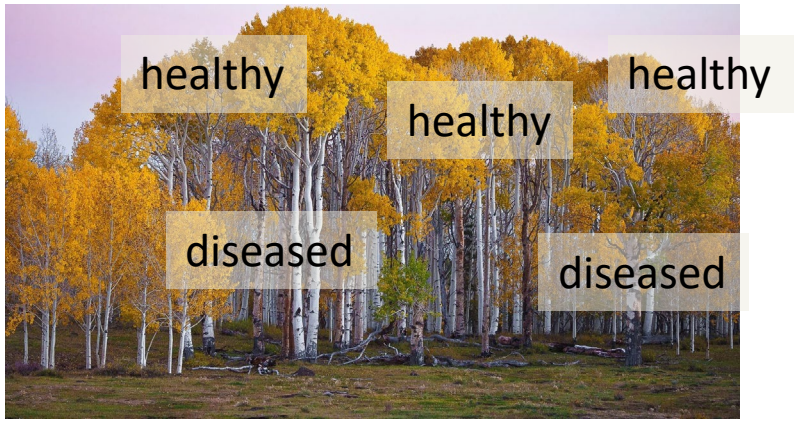
Output the ensemble of trees.

To make a prediction at a new point *x* we do:

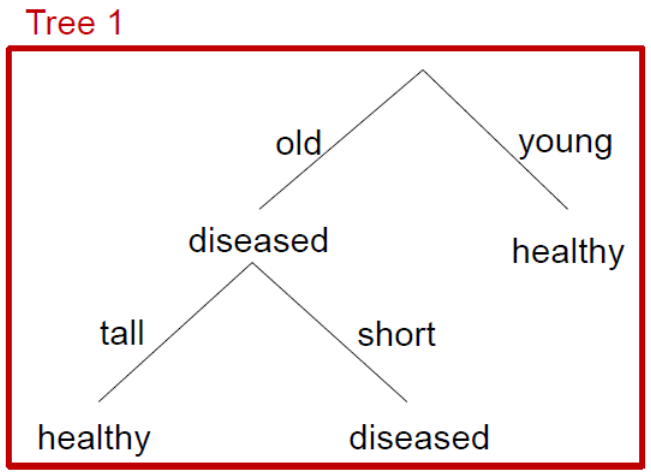
**For regression: average the results**

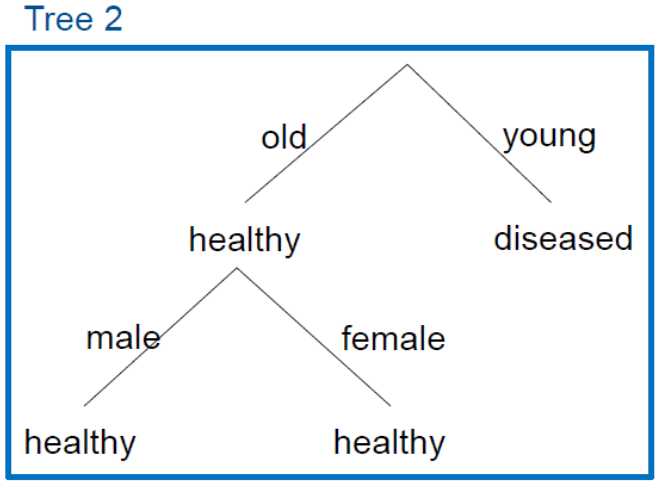
**For classification: majority vote**

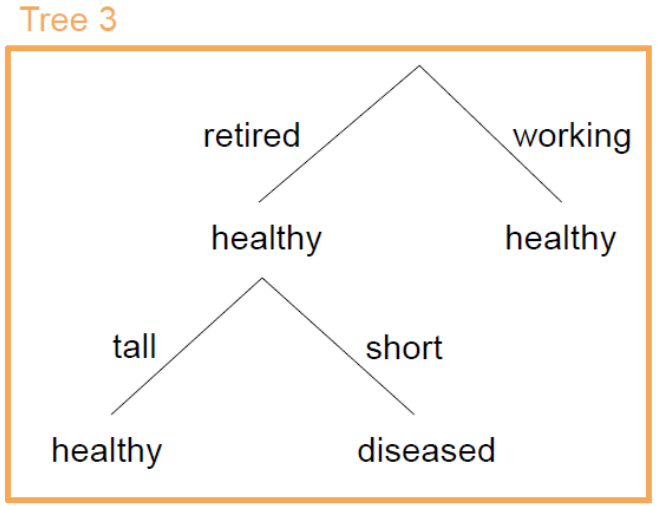
# Illustration





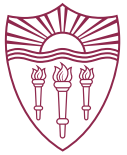

**New sample:** old, retired, male, short
**Tree predictions:** diseased, healthy, diseased

**Majority rule:**
**diseased**

# Feature importance

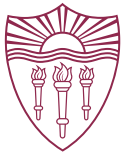
- measure feature importance by looking at how much the tree nodes that use that feature reduce impurity on average (across all trees in the forest).
- A weighted average, where each node's weight is equal to the number of training samples that are associated with the feature

# Difference from decision trees



- Train each tree on bootstrap resample of data
  - Bootstrap resample of data set with N samples:
  - Make new data set by drawing with replacement N samples; i.e., some samples will probably occur multiple times in new data set
- For each split, consider only m randomly selected variables
- Don't prune
- Fit B trees in such a way and use average or majority voting to aggregate results

# Trees vs Random Forests

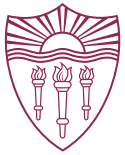


+ Trees yield insight into decision rules
+ Rather fast
+ Easy to tune parameters

- Prediction of trees tend to have a high variance

+ RF as smaller prediction variance and therefore usually a better general performance
+ Easy to tune parameters

- Rather slow
- “Black Box”: Rather hard to get insights into decision rules



- Questions?

- Virtual office hour
- https://usc.zoom.us/j/95136500603?pwd=VEJhblhWK25lT2N3RC9FNWk3eTJKQT09