TEXAS A&M UNIVERSITY

# CSCE 633: Machine Learning

## Lecture 9

## Overview

- Motivation for combining multiple learners
- Multi-expert combination
    - Learner fusion: voting, stacked generalization
    - Learner selection: mixture of experts, dynamic classifier selection
- Multi-stage combination
    - Sequential combination of learners: bagging, boosting
- Bias and variance in learning models

## Overview

- Motivation for combining multiple learners
- Multi-expert combination
    - Learner fusion: voting, stacked generalization
    - Learner selection: mixture of experts, dynamic classifier selection
- Multi-stage combination
    - Sequential combination of learners: bagging, boosting
- Bias and variance in learning models

## Combining multiple learners

### Motivation

- Each learning model comes with certain assumptions $\rightarrow$ works well on some parts of the data, but not well in other parts
- Learning is an ill-posed problem: with finite data, each algorithm converges to a different solution and fails under different circumstances
- There might be different learners that perform better in different circumstances
- Question 1: How do we generate multiple learners that complement each other?
- Question 2: How do we combine these learners to get a final result that performs better on the overall dataset?

## Combining multiple learners

### Diverse learners

- Not any learner combination works
- We need to find a set of diverse learners that:
    - differ in their decisions
    - complement each other

### How to increase diversity between learners?

- Different model representations (e.g. parametric & and non-parametric)
- Different hyper-parameters of the same model
- Different input representations (e.g. facial expression and speech characteristics for person identification)
- Different parts of data

## Combining multiple learners

### Combining multiple learners to final output

- Given $L$ learners and test input $\mathbf{x}$
- $d_j(\mathbf{x})$: the prediction of $j^{th}$ learner that uses data $\mathcal{X}_j$
- Final prediction
$$y = f(d_1, \ldots, d_L | \boldsymbol{\Phi})$$
where $\boldsymbol{\Phi}$ are the parameters of the learners

## Combining multiple learners

Taxonomy

Multi-expert combination: parallel combination of learners

- *learner fusion*: all learners generate an output for all inputs, then outputs are combined (e.g. voting)
  - Voting
  - Stacked generalization
- *learner selection*: chooses one (or few) learners for generating the output of each input
  - Mixture of experts with input-dependent weights
  - Dynamic classifier selection

Multi-stage combination: sequential combination of learners

- each learner is trained based on the data that have not been learned (enough) from the previous learner

## Overview

- Motivation for combining multiple learners
- Multi-expert combination
    - Learner fusion: voting, stacked generalization
    - Learner selection: mixture of experts, dynamic classifier selection
- Multi-stage combination
    - Sequential combination of learners: bagging, boosting
- Bias and variance in learning models

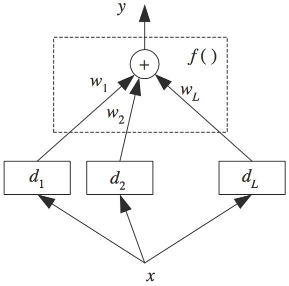
## Multiexpert Combination: Learner selection

### Voting

- Generating output as a linear combination of learners

$$y_i = \sum_j w_j d_{ji}, \quad w_j > 0, \quad \sum_j w_j = 1$$

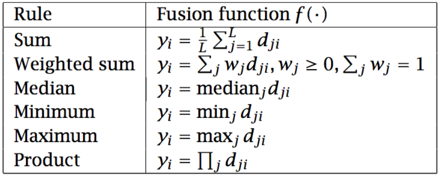
where $d_{ij}$ is the output of learner $j$ for class $i$



- Outputs from each learner should be normalized

## Multiexpert Combination: Learner selection

### Voting



| 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 | $y_i = \prod_j d_{ji}$ |

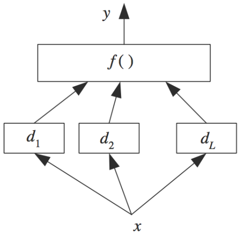
- median rule: robust to outliers
- product rule: each learner has veto power (it only takes one learner to have 0 output)
- weighted sum rule: $w_j$ can be related to accuracy of the learner or how much the learner voted for each class (e.g. distance from nearest training samples)

## Multiexpert Combination: Learner selection

### Stacked generalization

- The outputs of the base learners define a new $L$-dimensional space
- The output of the system is not a linear combination of the base learners, but it is learned through a combiner system

$$y = f(d_1, \ldots, d_L | \boldsymbol{\Phi})$$



## Multiexpert Combination: Learner selection

Stacked generalization

- The combiner learns
  - The correct output given a certain output combination of the learners
  - How the learners make errors
- We would like the learners to be as diverse as possible, so that they complement each other
- More flexible and with less bias, but adds extra parameters (and possible higher variance), and is computationally more expensive

## Overview

- Motivation for combining multiple learners
- Multi-expert combination
  - Learner fusion: voting, stacked generalization
  - Learner selection: mixture of experts, dynamic classifier selection
- Multi-stage combination
  - Sequential combination of learners: bagging, boosting
- Bias and variance in learning models

## Multiexpert Combination: Learner selection

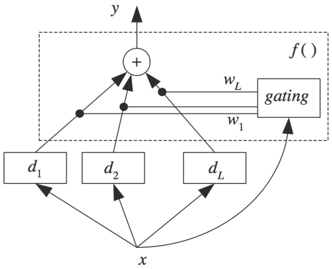
### Mixture of experts with input-dependent weights

- Voting method where the weights of each learner depend on the input

$$y = \sum_j w_j(\mathbf{x}) d_j, \quad w_j(\mathbf{x}) > 0, \quad \sum_j w_j(\mathbf{x}) = 1$$

where $d_{ij}$ is the output of learner $i$ for class $i$

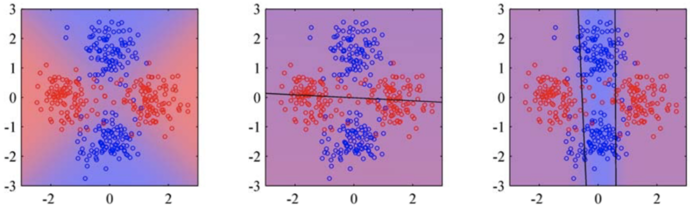
- Gating network whose outputs are the weights of the experts



## Multiexpert Combination: Learner selection

### Example: Mixture of experts in logistic regression

- $p(y|\mathbf{x}) = \sum_{k=1}^{K} \pi_k(\mathbf{x}) p_k(y|\mathbf{x})$
- Left figure: true probabilities of red/blue classes in the input space
- Center figure: Class probabilities from single logistic regression
  $\rightarrow$ ~0.5 probability per class for each sample
- Right figure: Class probabilities from mixture of two logistic regression models
  $\rightarrow$ higher probabilities assigned to the correct labels



## Conditional Mixture Models

### Mixtures of linear regression models

- K linear regression models
- Each with weight parameter $\mathbf{w_k}$
- All governed by the same precision $\beta$ for the Gaussian noise
- $y_k = \mathbf{w_k}^T\mathbf{x} + \epsilon$, $\epsilon \sim \mathcal{N}(0, 1/\beta)$
- We want to estimate $\boldsymbol{\theta} = [\{\pi_k\}_{k=1}^K, \{\mathbf{w}_k\}_{k=1}^K, \beta]$
- Likelihood of $k^{th}$ component $y_k \sim \mathcal{N}(\mathbf{w_k}^T\mathbf{x}, \beta^{-1})$
- Likelihood of final output

$$p(y|\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(y|\mathbf{w_k}^T\mathbf{x}, \beta^{-1})$$

- Joint estimation of model parameters and hidden variables $\rightarrow$ Expectation Maximization

## Conditional Mixture Models

- Original data-likelihood

$$\log p(\mathbf{y}|\boldsymbol{\theta}) = \sum_{n=1}^{N} \log \left( \sum_{k=1}^{K} \pi_k \mathcal{N}(y_n|\mathbf{w_k}^T \mathbf{x_n}, \beta^{-1}) \right)$$

- Set of binary variables $\mathbf{Z} = \{\mathbf{z_n}\}$, $z_{nk} \in \{0, 1\}$: all $z_{nk}$ are zero, expect the one that indicates the linear regression model responsible to generating prediction for the $n^{th}$ data point
- Data-likelihood after integrating hidden variables

$$\log p(\mathbf{y}, \mathbf{Z}|\boldsymbol{\theta}) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \log\{\pi_k \mathcal{N}(y_n|\mathbf{w_k}^T \mathbf{x_n}, \beta^{-1})\}$$

## Conditional Mixture Models

- **E-step:** Evaluating responsibilities

$$\gamma_{nk} = p(z_{nk} = 1 | \mathbf{x_n}, \boldsymbol{\theta^{old}}) = \frac{\pi_k \mathcal{N}(y_n | \mathbf{w_k}^T \mathbf{x_n}, \beta^{-1})}{\sum_{k=1}^K \pi_k \mathcal{N}(y_n | \mathbf{w_k}^T \mathbf{x_n}, \beta^{-1})}$$

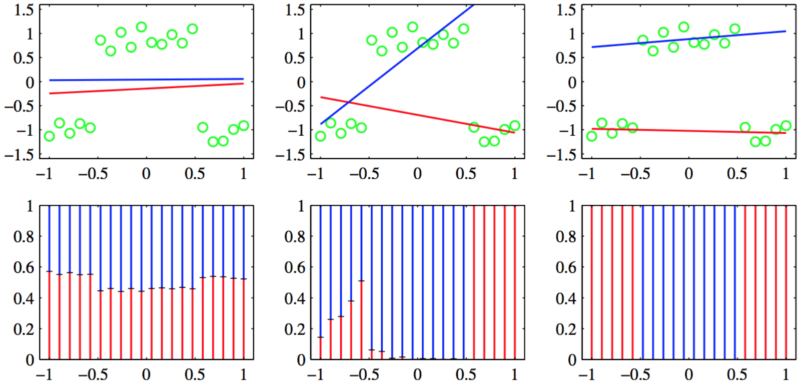
- **M-step:** Estimating model parameters

$$\pi_k = \frac{1}{N} \sum_{n=1}^{N} \gamma_{nk}$$

$$\mathbf{w_k} = (\mathbf{X}^T \mathbf{R_k} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{R_k} \mathbf{y} \ , \ \ \mathbf{R_k} = \mathrm{diag}(\gamma_{nk}) \ \ \text{(weighted least squares)}$$

$$\frac{1}{\beta} = \frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_{nk} (y_n - \mathbf{w_k}^T \mathbf{x_n})^2$$

# Conditional Mixture Models



upper figure: input data and a mixture of two linear regression models (initial, 30 EM iterations, 50 EM iterations)

lower figure: estimated responsibilities (initial, 30 EM iterations, 50 EM iterations)

## Multiexpert Combination: Learner selection

Dynamic classifier selection

Given a test input

- Find the $K$ nearest training samples $\mathcal{X}_k$ to the input
- Look at the accuracies of all learners $d_1, \ldots, d_L$ in $\mathcal{X}_k$
- Chose the learner $d^*$ that performs the best on $\mathcal{X}_k$
- Report the final result of the test input using learner $d^*$

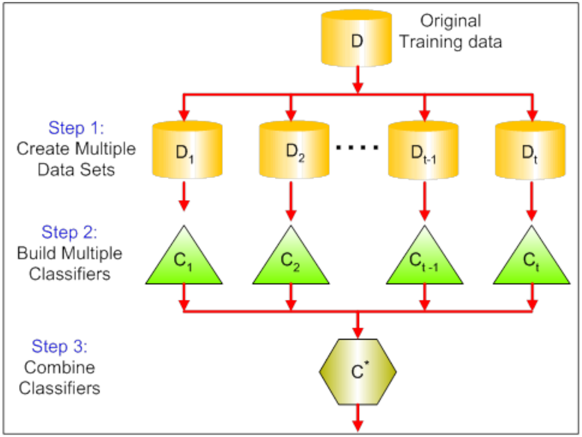
## Overview

- Motivation for combining multiple learners
- Multi-expert combination
  - Learner fusion: voting, stacked generalization
  - Learner selection: mixture of experts, dynamic classifier selection
- Multi-stage combination
  - Sequential combination of learners: bagging, boosting
- Conditional Mixture Models: Linear regression
- Bias and variance in learning models

## Bagging

- Train learners on different parts of the data $\mathcal{D}$
- Generate $L$ different samples $\mathcal{D}_i$ using bootstrap
    - Draw $N$ samples randomly with replacement
- Train a model $C_i$ on each $\mathcal{D}_i$
- Combine the decision from all models to a final one $C^*$

## Bagging


D
Original Training data
Step 1: Create Multiple Data Sets
$D_1$
$D_2$
....
$D_{t-1}$
$D_t$
Step 2: Build Multiple Classifiers
$C_1$
$C_2$
$C_{t-1}$
$C_t$
Step 3: Combine Classifiers
$C^*$

- If the training set is very unstable (i.e., small changes in different samples cause significant changes in the output), then bagging is likely to help

## Boosting

- Multistage learning: each learner is trained on the data that have not been learned from the previous one
- We actively generate complementary base-learners by training the next learner on the mistakes of the previous learners
- We use weak learners to get complex decision boundaries, i.e. learners with accuracy slightly more than $1/2$
- We will example Adaboost (Adaptive Boosting), which is a modified version of the original boosting algorithm

## Boosting: Adaboost

### Algorithm Outline

- Input: $N$ samples $\{\mathbf{x_n}, y_n\}$, $y_n \in \{-1, 1\}$
- Each sample is assigned a weight, initialized at $w_1(n) = \frac{1}{N}, \forall n$
- For $t = 1, \dots, T$
    - Step 1: Train a weak classifier $h_t(\mathbf{x})$ based on current weights $w_t(n)$ by minimizing the *weighted classification error*

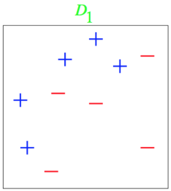
$$\epsilon_t = \sum_n w_t(n) \mathbb{I}[y_n \neq h_t(\mathbf{x_n})]$$

    - Step 2: Calculate contribution of classifier $t$: $\beta_t = \log \frac{1-\epsilon_t}{\epsilon_t}$
    - Update weights $w_{t+1}(n) = w_t(n) \exp\left(\beta_t \mathbf{I}\left(y_n \neq h_t(\mathbf{x_n})\right)\right)$ and normalize them s.t. $\sum_n w_t(n) = 1$
- The final output is $h(\mathbf{x}) = \text{sign}\left(\sum_t \beta_t h_t(\mathbf{x})\right)$

## Boosting: Adaboost

Example: 10 data points

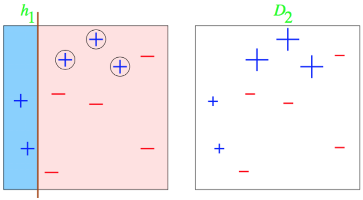
- Base classifier $h(\cdot)$: either horizontal or vertical lines
- The data points are clearly not linear separable
- In the beginning, all data points have equal weights (the size of "$+$" and "$-$" markers is the same)



## Boosting: Adaboost

Example: Round 1 (t=1)

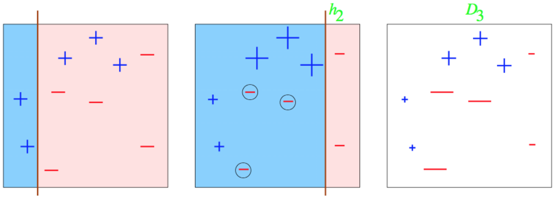
- 3 misclassified (with circles): $\epsilon_1 = 0.3 \rightarrow \beta_1 = 0.42$
- Weights recomputed: the 3 misclassified data points receive larger weights



## Boosting: Adaboost

### Example: Round 2 (t=2)

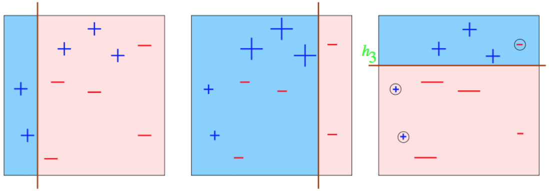
- 3 misclassified (with circles): $\epsilon_2 = 0.21 \rightarrow \beta_2 = 0.65$ Note that $\epsilon_2 \neq 0.3$ as those 3 data points have weights less than $1/10$
- Weights recomputed: the 3 misclassified data points receive larger weights. The data points classified correctly on round t = 1 receive much smaller weights as they have been it consistently classified correctly.



## Boosting: Adaboost

### Example: Round 3 (t=3)

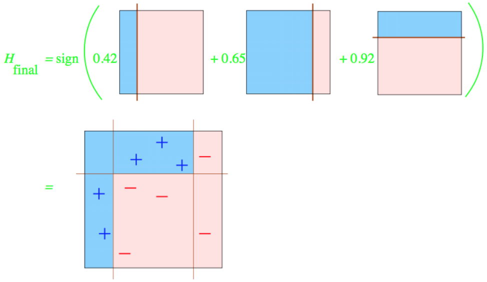
- 3 misclassified (with circles): $\epsilon_3 = 0.14 \rightarrow \beta_3 = 0.92$
- Note that those previously correctly classified data points are now misclassified. However, we might be lucky on this as if they have been consistently classified correctly, then this round?s mistake is probably not a big deal.



## Boosting: Adaboost

### Example: Combining all classifiers

- All data points are now classified correctly



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

## Boosting: Adaboost

### Nonlinear basis learned by boosting

- Compute $\text{sign}[f_1(\mathbf{x})], \text{sign}[f_2(\mathbf{x})], \ldots$
- Combine into a linear classification model

$$y = \text{sign}\left\{ \sum_t \beta_t \text{sign}[f_t(\mathbf{x})] \right\}$$

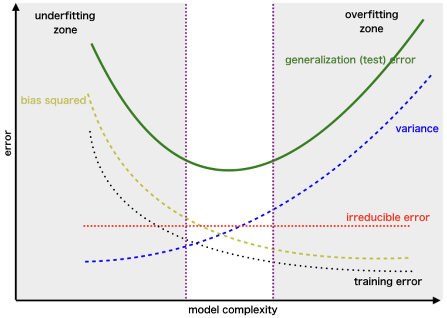
- Equivalently, each stage learns a nonlinear basis $\phi_t(\mathbf{x}) = \text{sign}[f_t(\mathbf{x})]$

## Overview

- Motivation for combining multiple learners
- Multi-expert combination
  - Learner fusion: voting, stacked generalization
  - Learner selection: mixture of experts, dynamic classifier selection
- Multi-stage combination
  - Sequential combination of learners: bagging, boosting
- Bias and variance in learning models

## Generalization error, bias, and variance

- Generalization error: The error on unseen data, test error
- Bias: The error from wrong or oversimplified model assumptions
    - *Underfitting* relates to high bias
- Variance: The error from sensitivity to small fluctuations in training data
    - *Overfitting* relates to high variance
- Noise: The irreducible error inherent to the problem itself



## Generalization error, bias, and variance

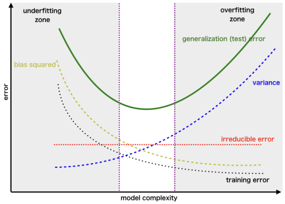
- In supervised machine learning problems we want to find the association between the input $\mathbf{x}$ and output $y$
- Assuming random noise, we have: $y = f(\mathbf{x}) + \epsilon$
- Noise is a random variable with zero mean and variance $\sigma_\epsilon^2$
  - $\mathbb{E}[\epsilon] = 0$ and $var(\epsilon) = \sigma_\epsilon^2$
- In practice, we do not know $f$ or $\epsilon$, but we can estimate them.

## Generalization error, bias, and variance

- A supervised learning algorithm finds function $\hat{f}$ which approximates $f$, therefore $y \approx \hat{f}(\mathbf{x})$
- Generalization error: The error on unseen data, test error, $\mathbb{E}[(y - \hat{f}(\mathbf{x}))^2]$
- Bias: The error from wrong or oversimplified model assumptions, the average prediction (over different realizations of training data) for a given test point, $bias[\hat{f}(\mathbf{x})] = \mathbb{E}[\hat{f}(\mathbf{x})] - f(\mathbf{x})$
- Variance: The error from sensitivity to small fluctuations in training data, the mean squared deviation of $\hat{f}(\mathbf{x})$ from its expected value $\mathbb{E}[\hat{f}(\mathbf{x})]$ over different realizations of training data, $var(\hat{f}(\mathbf{x})) = \mathbb{E}[(\hat{f}(\mathbf{x}) - \mathbb{E}[\hat{f}(\mathbf{x})])^2]$
- Noise: The irreducible error inherent to the problem itself, $\sigma_\epsilon^2$

$$\boxed{\mathbb{E}[(y - \hat{f}(\mathbf{x})^2] = bias[\hat{f}(\mathbf{x})]^2 + var(\hat{f}(\mathbf{x})) + \sigma_\epsilon^2}$$

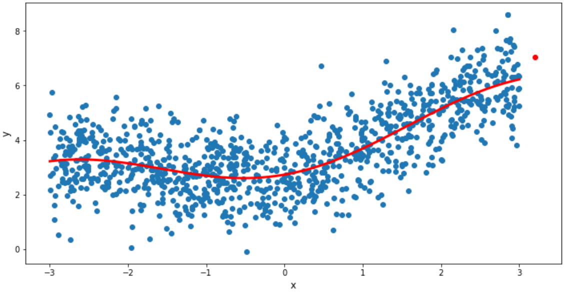
https://towardsdatascience.com/the-bias-variance-tradeoff-8818f41e39e9



## Tradeoff between bias and variance

### Regression example

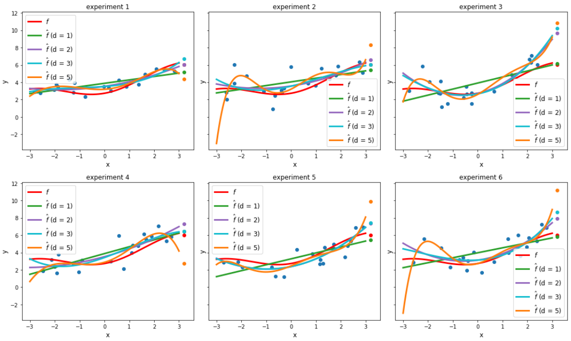
- Data generated from $f(x) = \frac{1}{2}x + \sqrt{max(0,x)} - cosx + 2$
- We randomly generate 1000 data points



## Tradeoff between bias and variance

### Regression example

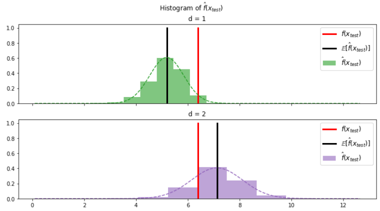
- We will fit polynomial regressions of varying degree $d$, $\hat{f}(x) = w_0 + w_1 x + w_2 x^2 + \ldots + w_d x^d$
- We randomly choose 20 points (out of the 1,000) to train our polynomial regression models
- We repeat the experiment 6 times
- Blue circles: test samples



## Tradeoff between bias and variance

### Regression example

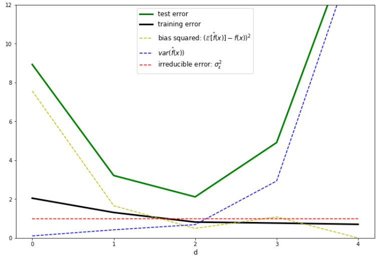
- The deviation of $\hat{f}(x)$ from $f(x)$ *on average* (i.e., bias), is larger for more simplistic models, since our assumptions are not as representative of the underlying true relationship $f$
- A more complex model is much more sensitive to small fluctuations in the training data



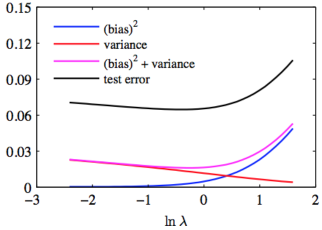
## Tradeoff between bias and variance

### Regression example

#### Increasing polynomial degree



#### Increasing regularization



## Tradeoff between bias and variance

How to compute bias and variance in a given dataset?

- Input dataset $\mathbf{D}$
- Create $B$ variants $\{\mathbf{D_1}, \ldots, \mathbf{D_B}\}$ of the original data $\mathbf{D}$
  - Pick data samples $(\mathbf{x}, y)$ randomly from $\mathbf{D}$ and add it to $\mathbf{D_b}$
- For each variant $\mathbf{D_b}$
  - Split the data in $\mathbf{D_b}$ into training $\mathbf{T_b}$ and testing $\mathbf{U_b}$
  - Train model on $\mathbf{T_b}$, test on $\mathbf{U_b}$
- For each sample $(\mathbf{x}, y)$ of the original data $\mathbf{D}$ we have many predictions $h_1(\mathbf{x}), \ldots, h_n(\mathbf{x})$ coming from the variants $\mathbf{D_b}$, to which sample $(\mathbf{x}, y)$ was in the test set
  - Bias for sample $(\mathbf{x}, y)$: $\frac{h_1(\mathbf{x})+\ldots+h_n(\mathbf{x})}{n} - y$
  - Variance for sample $(\mathbf{x}, y)$:
    $\frac{1}{N}\sqrt{(h_1(\mathbf{x}) - \hat{h}(\mathbf{x}))^2 + \ldots + (h_n(\mathbf{x}) - \hat{h}(\mathbf{x}))^2}$
    $\hat{h}(\mathbf{x}) = \frac{h_1(\mathbf{x})+\ldots+h_n(\mathbf{x})}{n}$

## Tradeoff between bias and variance

How to compute bias and variance in a given dataset?

- By measuring the bias and variance on a problem, we can determine how to improve our model
    - If bias is high, we might want to have a more complex model
    - If variance is high, we might want to reduce the complexity
- Steps taken to fight one can end up worsening the other!

## What have a learned so far

- Inherent trade-off between bias and variance in learning
- Combining multiple learners
  - Learner fusion: voting, stacked generalization
  - Learner selection: mixture of experts, dynamic classifier selection
- Combination of multiple learners can decrease final variance
- Learners must be diverse
- **Readings:** Alpaydin 17.6-17.7