# Ensemble Methods

Nan Ye

School of Mathematics and Physics
The University of Queensland

# The Journey Begins with a Cow



Guess the weight of the cow: poll

# Where Are We Heading to?

**How to ~~accurately estimate the cow weight~~ build good ML models**

- Making use of a crowd ⇒ Week 7 Ensemble methods

  *each of us is a biological prediction model trained on different datasets...*

- Using a neural network ⇒ Week 8 and 9 Neural networks

  *brain-inspired models, some are good for images...*

- Making a robust model ⇒ Week 10 Robust machine learning

  *malicious users, outliers,...*

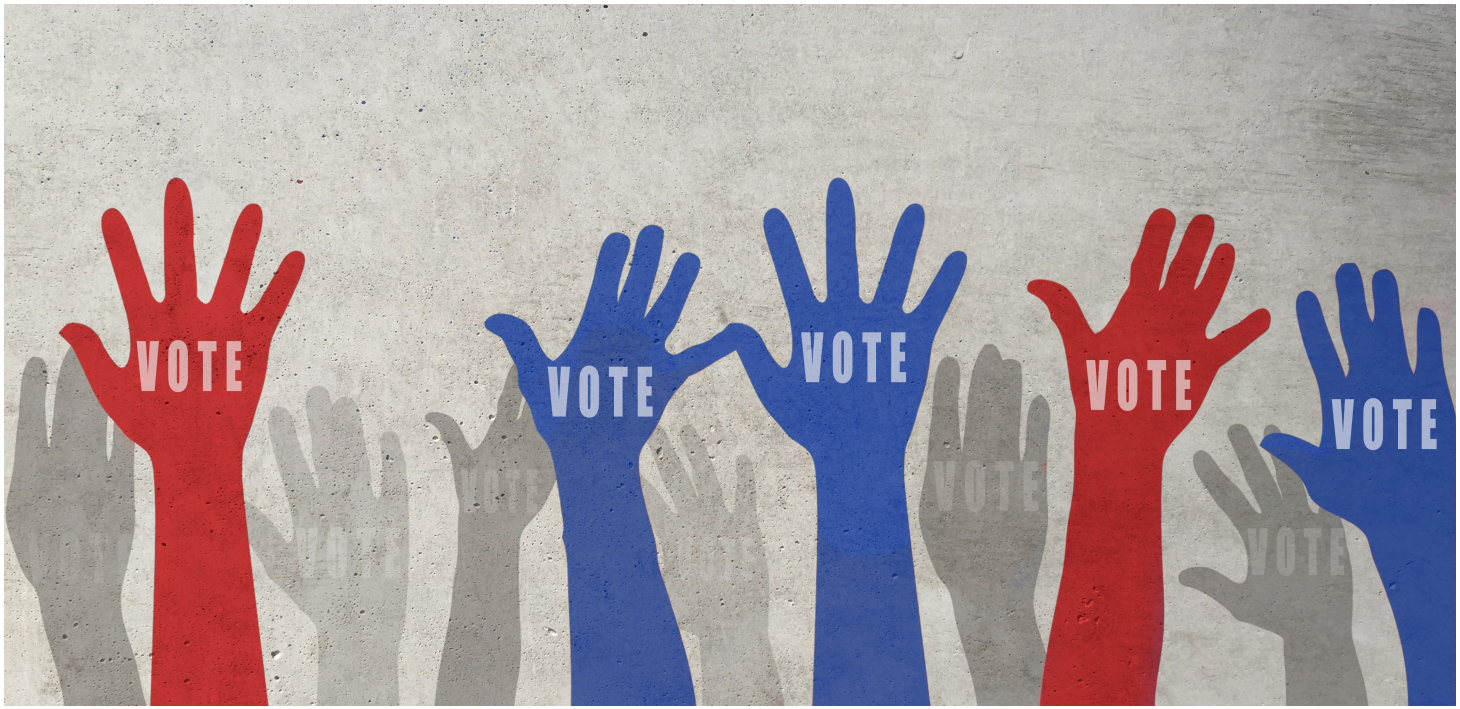
- Asking for explanations ⇒ Week 11 Interpretable machine learning

  *Human behavior cannot be explained for the simple reason that it makes no sense. – Martin Rubin*
  *...still, let's ask the machines for explanations...*

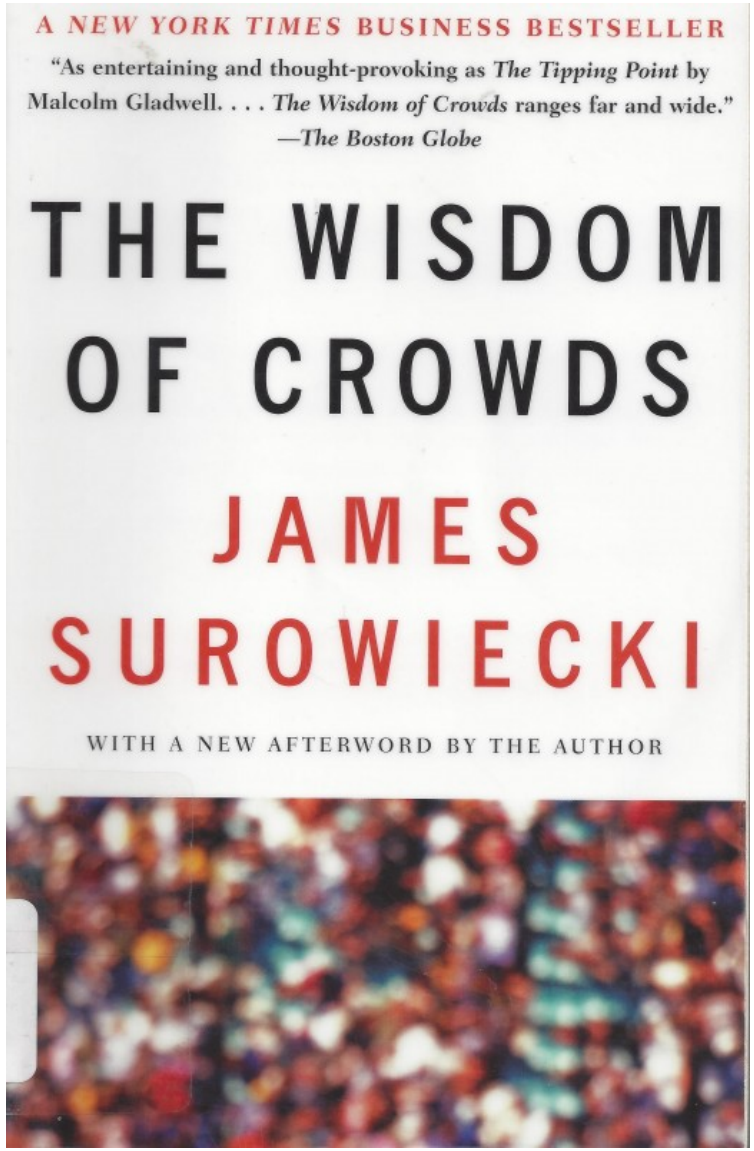
- Exploiting prior beliefs ⇒ Week 12 Bayesian methods

# Ensemble: A Popular Idea...

**Francis Galton's Ox (1906)**

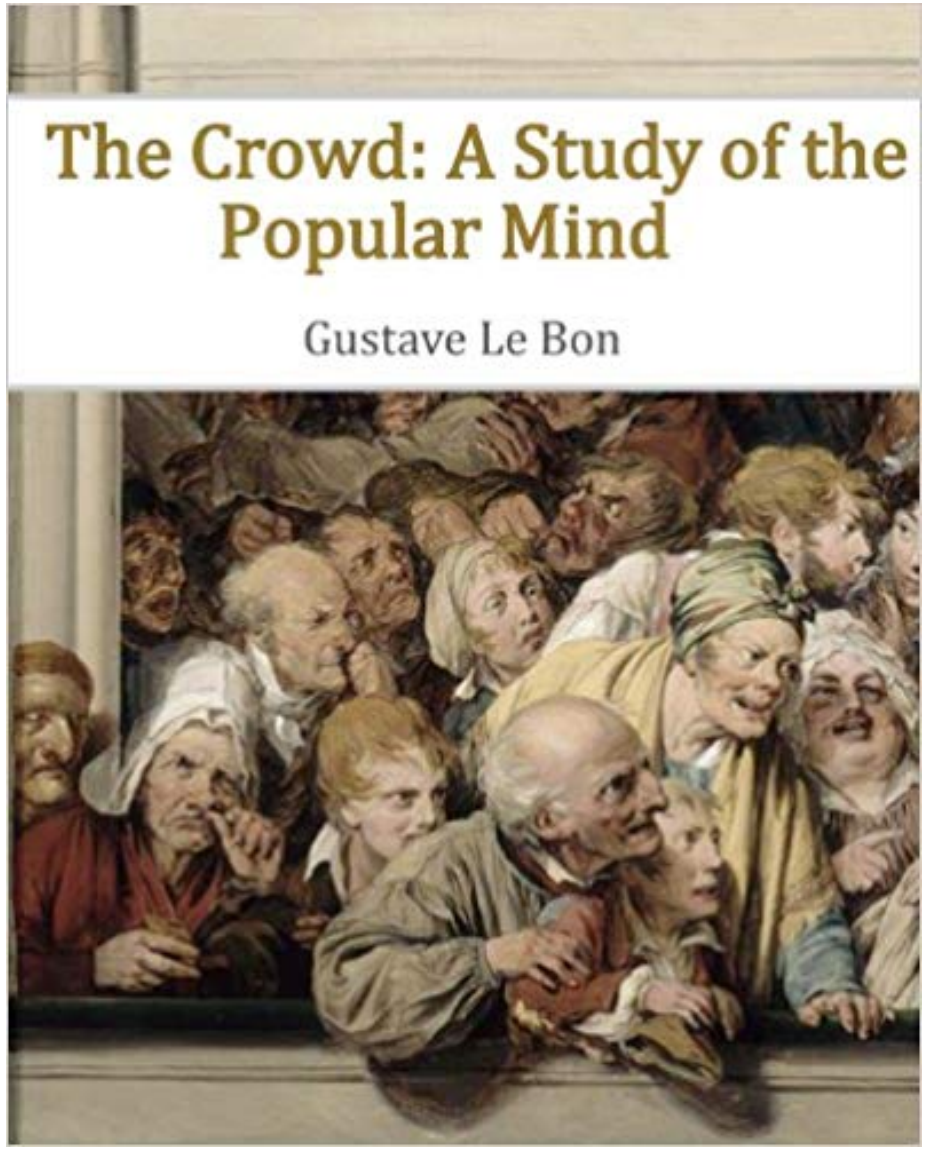



## Condorcet's Jury Theorem (1785)




**the group as a whole knew them all (2004)**

*anecdote: the author guessed Penelope's weight and was off by 46%*




**Ensemble is not the panacea...**

# Ensembles in Machine Learning

- Various methods have been proposed to build powerful ensembles of machine learning models.
- These can often be used as generic methods to improve the performance of basic machine learning models like decision trees (DTs), SVMs.
- Ensemble methods have found successes in various applications.


# The Viola-Jones algorithms for face detection

*(AdaBoost using one-level DTs)*

Viola and Jones, Robust real-time object detection, 2001

*4.3. Fraud detection*

We have often met many kind of customer fraud events in our society [24–26]. For examples, the cellular fraud in the telecommunication industry, the credit card fraud in the banking industry, and compensation cheating in the insurance company. Fraud not only costs very much for a company, but also causes inconvenience and much cost for the customer. One method for fraud detection, called user profiling method [24], is checking suspicious changes in user behaviors and determining general patterns of fraud by a massive amount of user action data analysis. There are many different approaches in modelling fraud patterns, such as Bayesian network [26] and HMM [27]. Here, we tackled a mobile telecommunication payment fraud detection using the proposed SVM ensemble.

## SVM ensemble for fraud detection

Kim et al., Constructing support vector machine ensemble, 2003

# Ensemble > Basis Functions

- An ensemble of basis models can often be used to represent a more complex functional relationship than each individual model can do.
- This is true even the basis models are "simple" (e.g. threshold classifiers or decision stumps).

  *decision stump = one-level DT*

**Example. Optimal operating temperature**

- A manager of a circuit assembly production facility noted that the temperature affects the defect rates of the products.
- Two machine learning experts were hired to build models to predict whether a temperature is suitable based on historical default rates.
- Both built a model under the assumption that only a threshold need to be learned
  - Expert $c_1$'s model: $+1$ (✓) if temperature $\geq 10$, -1 (✗) otherwise.
  - Expert $c_2$'s model: $+1$ (✓) if temperature $< 20$, -1 (✗) otherwise.

- Consider a combined model

$$c(x) = \text{sgn}(0.5c_1(x) + 0.5c_2(x) - 0.5).$$

*Equivalently,* $c = \text{sgn}(0.5c_1 + 0.5c_2 + 0.5c_3)$*, where* $c_3$ *always predicts -1.*

- The combined model is a "two-thresholds" model

| classifier | temperature | | |
|---|---|---|---|
| | $(-\infty, 10)$ | $[10, 20)$ | $[20, \infty)$ |
| $c_1$ | -1 | $+1$ | $+1$ |
| $c_2$ | $+1$ | $+1$ | -1 |
| $c$ | -1 | $+1$ | -1 |

- In general, a linear combination of threshold classifiers can be more expressive than a threshold classifier.

## Example. 2-level decision trees in 2D

- Basis classifers
  - each classifies four quadrants in a 2D feature space into -1 or $+1$

$$c_1 = \begin{array}{|c|c|}\hline +1 & -1 \\ \hline -1 & -1 \\ \hline \end{array} \qquad c_2 = \begin{array}{|c|c|}\hline -1 & -1 \\ \hline -1 & +1 \\ \hline \end{array} \qquad c_3 = \begin{array}{|c|c|}\hline +1 & +1 \\ \hline +1 & +1 \\ \hline \end{array}$$

- Combined classifier

$$2c_1 + 2c_2 + c_3 = \begin{array}{|c|c|}\hline +1 & -3 \\ \hline -3 & +1 \\ \hline \end{array}$$

$$c = \text{sgn}(2c_1 + 2c_2 + c_3) = \begin{array}{|c|c|}\hline +1 & -1 \\ \hline -1 & +1 \\ \hline \end{array}$$

# Ensemble Learning

- Independent methods
  - Each model is trained independently of others.
  - e.g. bagging, random forest.
- Dependent methods
  - A model in the ensemble makes use of previously trained models.
  - e.g. AdaBoost.
- We will cover bagging, random forest, and AdaBoost.

# Checking Your Understanding

Which of the following statement is correct? (Multiple choice)

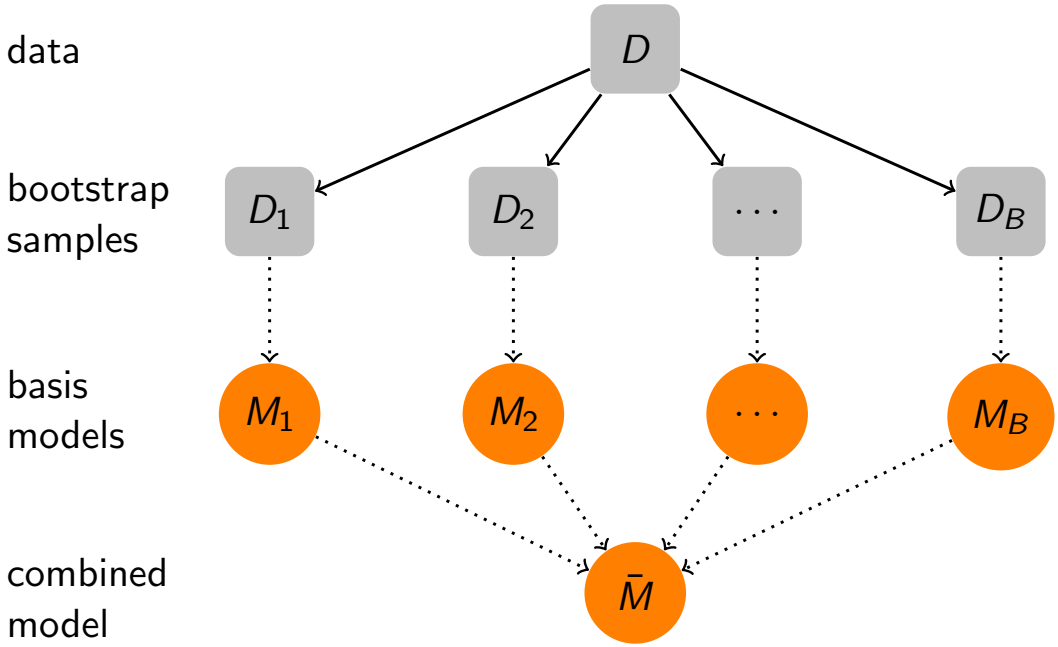
**(a)** An ensemble of models can represent functional relationships not representable by basis models.

**(b)** Each model in an ensemble is constructed independently.

**(c)** The Viola-Jones algorithm builds an ensemble of SVMs.

# Bagging (Bootstrap Aggregating)

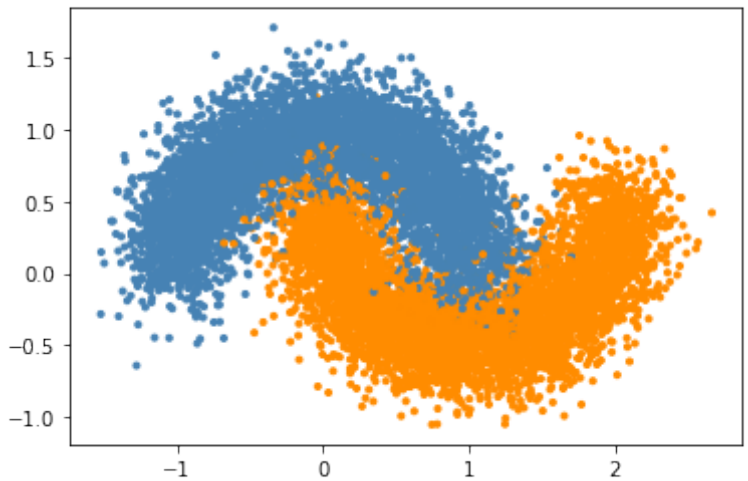
- Training
  - Create multiple datasets known as bootstrap samples.
    - bootstrap sample of a dataset of size $n$ = a sample of size $n$ obtained by sampling with replacement from the given dataset.
    - for large $n$, each bootstrap sample looks like a random training set drawn from the true data distribution
    - a bootstrap sample contains roughly $0.632n$ different examples from the given dataset.
  - Train basis models independently, one on each bootstrap sample.
- Testing
  - Classification: predict the majority of the basis models' predictions.
  - Regression: predict the average of the basis models' predictions.



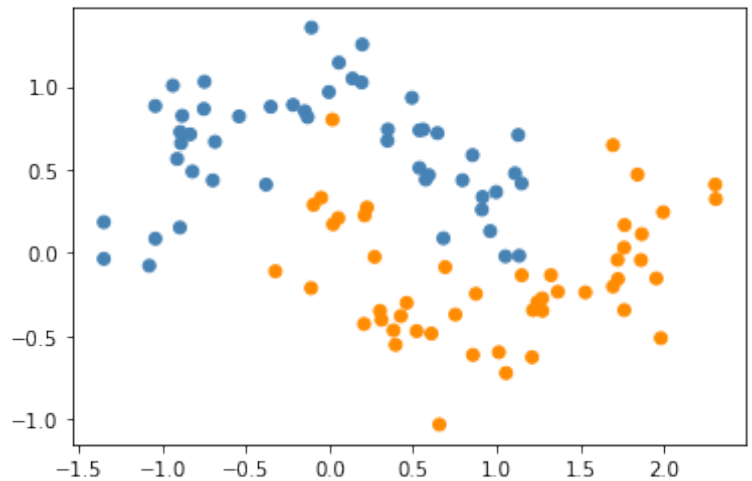

(*Classifcation*) $\bar{M}(x) = \text{majority}\{M_1(x), \ldots, M_B(x)\},$

(*Regression*) $\bar{M}(x) = \frac{1}{B}\sum_{i=1}^{B} M_i(x)$

# Case Study: Two-moons



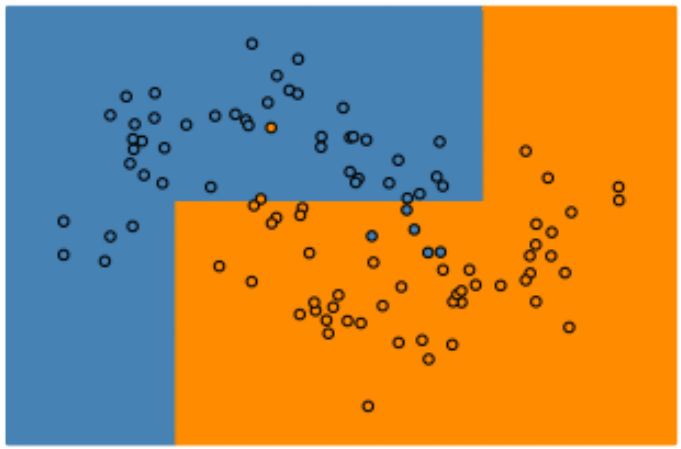

how the "entire" dataset looks like



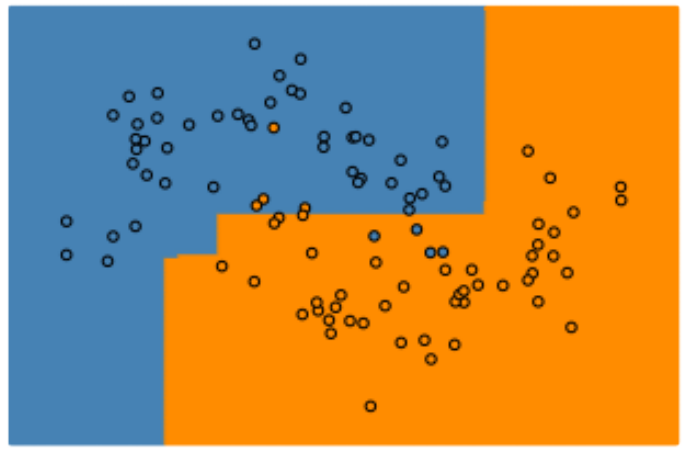

our training set

## Bagging with 2-level DTs

2-level DT



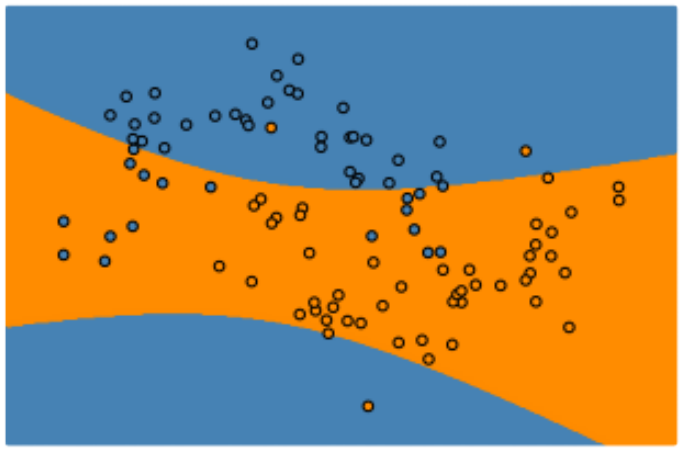
bagging ($B = 100$)



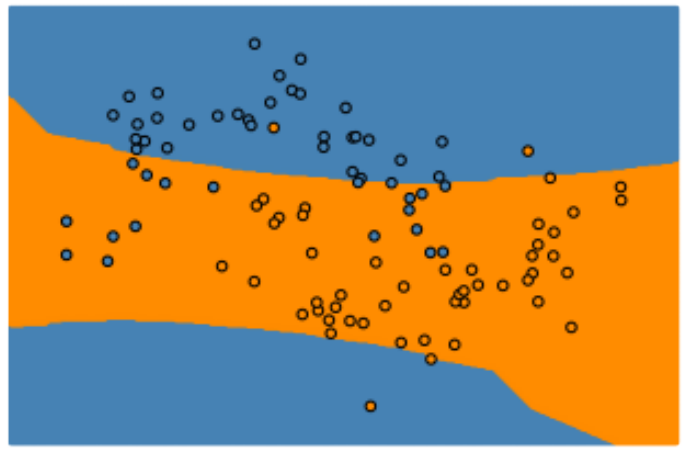
- 2-level DTs creates axis-aligned decision boundaries.
- Bagging creates a slightly more complex decision boundary.

## Bagging with SVMs (quadratic kernel)

SVM with quadratic kernel



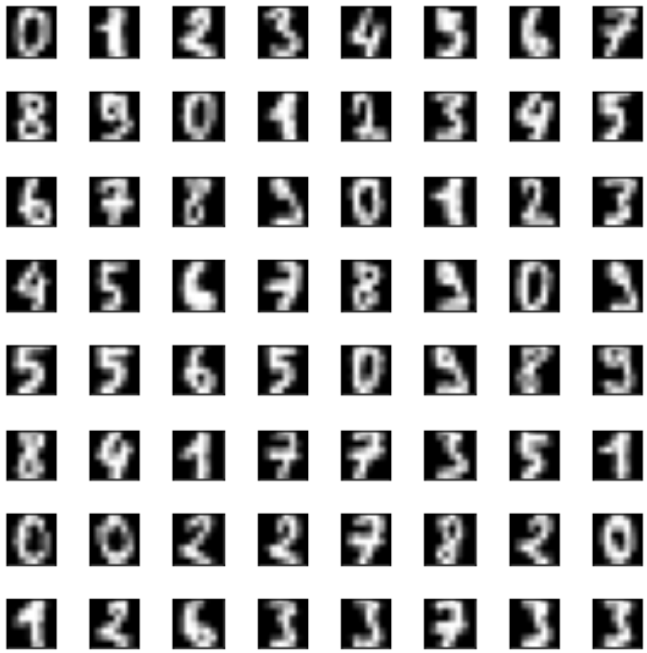
bagging ($B = 100$)



- SVM with quadratic kernel creates a hyperbolic decision boundary.
- Bagging creates a slightly more complex decision boundary.

# Case Study: Digit Dataset

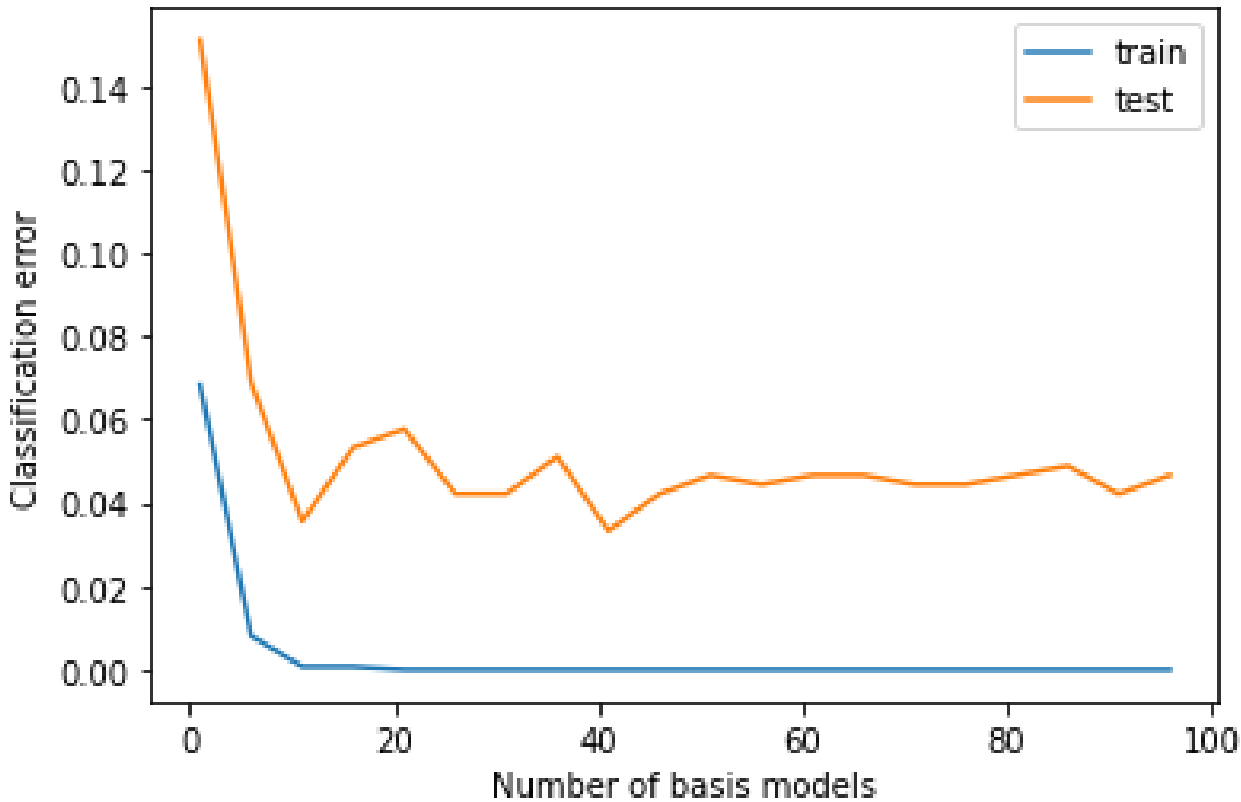
- 1797 handwritten digit images of size 8x8



  available from `sklearn.datasets.load_diabetes`
- Random 75%/25% train-test split

## Bagging with DTs

- Bagging significantly reduces both the training and test errors as we use more basis models.



# Model Selection for Bagging

**Primer**

- Model selection is about choosing a model of the right complexity.
  - Right complexity = good generalization performance.
  - In other words, we want a model that does not underfit or overfit.
- It can be about choosing between different classes of models, or choosing suitable values for hyperparameters controlling the complexity of a model.
- Standard techniques: validation/development set, cross-validation.

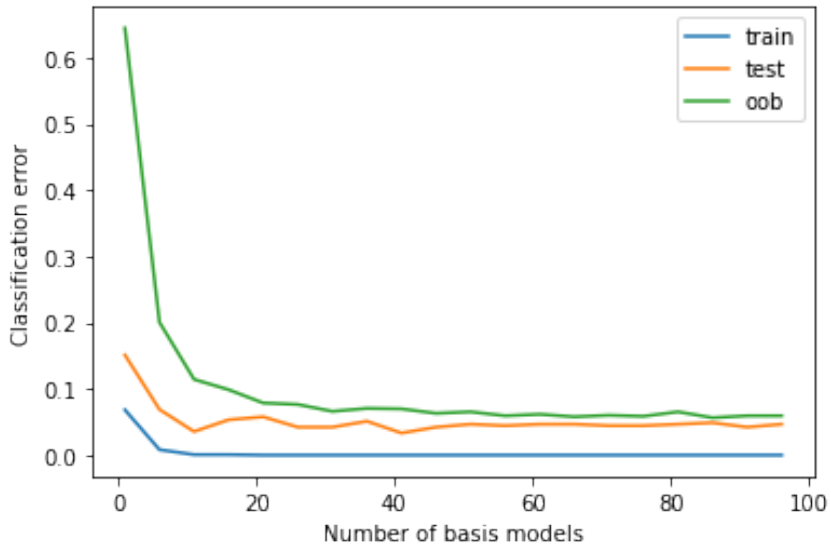
## Effect of number of models for bagging

- For bagging, the number of models used is a key complexity measure.
- In theory, the larger the number models, the better the ensemble is.
- In practice, there is a diminishing return to add additional models to a large ensemble.
- $\Rightarrow$ Stop adding more models when doing so leads to little improvement in generalization performance.

## Out-of-bag (OOB) error

- Validation set method requires additional data, and cross-validation is computationally expensive.
- OOB error provides an estimate of the generalization performance of bagging that
  - does not require additional data, and
  - is efficiently computable.
- Calculation of the OOB error
  - For each training example, find all basis models trained without it, and compute a prediction (OOB prediction) on it using these models.
  - OOB error = the error for all OOB predictions.
- Example: suppose we have 100 basis models. If a training example $(x_i, y_i)$ is not used for training 81 of them, and 61 of them predict $+1$ on $x_i$ and 20 predicts -1 on $x_i$, then the OOB prediction on $x_i$ is $+1$.

## Revisiting bagging with DTs on digit dataset




- The test error has not stabilised yet when training error stabilises.
- The test error has stabilised when the OOB error stabilises.
- Overall, OOB error is a better estimator for test performance.

# Why Bagging Works

**Measuring an algorithm's performance**

- In practice, we usually work with a single training and test set randomly sampled from the data distribution.
- However, what we are interested in is whether an algorithm works in general, that is, whether it works well on different randomly sampled training and test sets.
- In theory, we measure the average performance using the expected prediction error.

## Expected prediction error (EPE)

- The EPE of a regression algorithm on a fixed example $x$ is

$$\mathbb{E}((\tilde{Y} - Y)^2), \text{ where}$$

  - $Y$ is the output on $x$ with distribution $P(Y \mid x)$,
  - $\tilde{Y}$ is the predicted value on $x$ for a model trained on a random training set (typically, of a fixed size).

- If we have $N$ independently sampled outputs $y^{(1)}, \ldots, y^{(N)}$ for $x$, and $N$ predicted values $\tilde{y}^{(1)}, \ldots, \tilde{y}^{(N)}$ given respectively by models trained on $N$ random training sets, then for large $N$,

$$\text{EPE} \approx \frac{1}{N} \sum_{i=1}^{N} (y^{(i)} - \tilde{y}^{(i)})^2$$

.

## Bias-variance decomposition

- EPE can be decomposed as

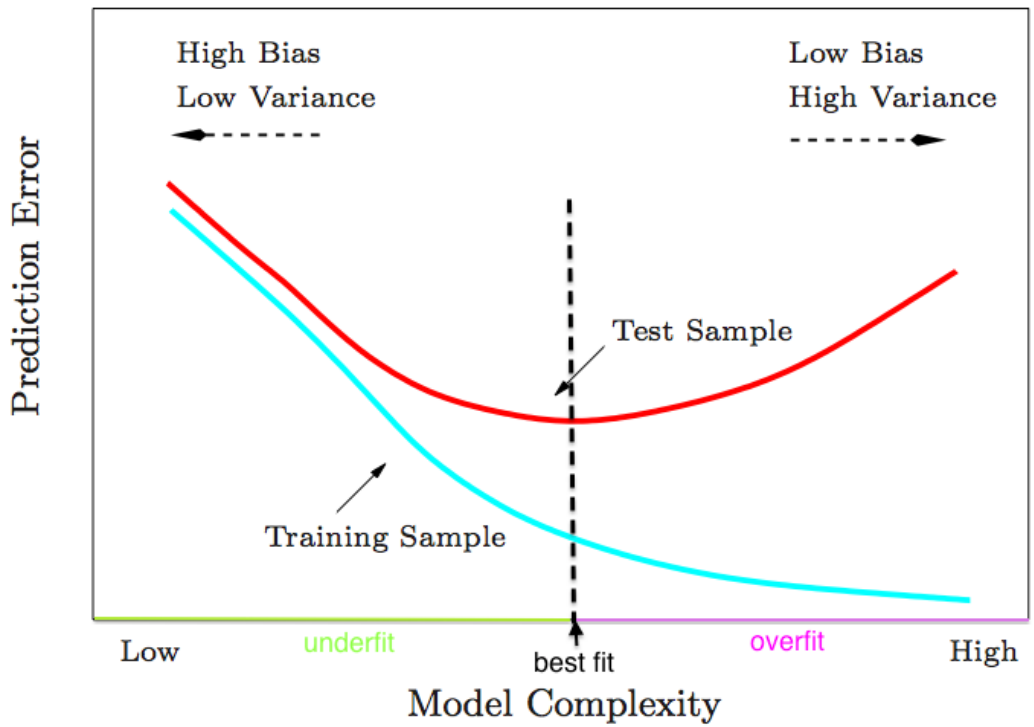
$$\overbrace{\mathbb{E}\big((\tilde{Y}-Y)^2\big)}^{\text{expected prediction error}} = \overbrace{\mathbb{E}\big((\tilde{Y}-\mathbb{E}(\tilde{Y}))^2\big)}^{\text{variance}} + \overbrace{\big(\mathbb{E}(\tilde{Y})-\mathbb{E}(Y)\big)^2}^{\text{bias (squared)}} + \overbrace{\mathbb{E}\big((Y-\mathbb{E}(Y))^2\big)}^{\text{irreducible noise}},$$

  where expectation is wrt the random training set and $Y$.

  - Variance: characterizes how different the prediction on $x$ will be if we get a different random training set.
  - Bias: characterizes whether the model is able to capture the input-output relationship on average.

## Bias-variance tradeoff

- In general, we can't minimize bias and variance at the same time
  - A less complex model has higher bias and lower variance.
  - A more complex model has lower bias and higher variance.



**Finally, why bagging works...**

- When using $B$ basis models, bagging produces $B$ *identically distributed* but *correlated* predictions $\tilde{Y}^{(1)}, \ldots, \tilde{Y}^{(B)}$, and predicts

$$\bar{Y} = \frac{1}{B} \sum_{i=1}^{B} \tilde{Y}^{(i)}$$

- Identically distributed $\Rightarrow$ averaging doesn't change bias, because

$$\mathbb{E}(\bar{Y}) = \mathbb{E}(\tilde{Y}^{(i)})$$

  for all $i$.
  - N.B. The bias of a model trained on a bootstrap sample can be different from the bias of a model trained on the original sample, but we assume that they are the same here.

- Correlation $\Rightarrow$ variance reduction, because

$$\text{Var}(\bar{Y}) = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2 < \sigma^2 = \text{Var}(\tilde{Y}^{(i)})$$

  - Assumptions: pairwise correlation $\text{corr}(\tilde{Y}^{(i)}, \tilde{Y}^{(j)})$ is a constant $\rho < 1$ for $i \neq j$, and $\text{Var}(\tilde{Y}^{(i)})$ equals $\sigma^2$ for any $i$.

- In short, bagging is better than a single model because it keeps the bias the same, while reducing the variance.

# Checking Your Understanding

Which of the following statement is correct? (Multiple choice)

**(a)** Computing the OOB error for bagging is more expensive than computing the cross-validation error.

**(b)** Bagging is only used to construct SVM ensembles.

**(c)** Bagging reduces bias while keeping variance the same.

**(d)** In general, a more complex model has lower bias but larger variance.

# Random Forests (RFs)

- Random forest is
  - a modified bagging method for DTs,
  - designed to further reduce variance by building a collection of *decorrelated* trees.
- RF and bagging with DTs differs *only* in how a decision is trained on a bootstrap sample
  - Bagging: for each node, choose the splitting variable from all $d$ features.
  - RF: for each node, first randomly sample $m < d$ features, then choose the splitting variable among them.
- That's the only difference, but this subtle difference is important.

## Why RF is an improvement of bagging

- Recall that for bagging

$$\mathrm{Var}(\bar{Y}) = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2 < \sigma^2 = \mathrm{Var}(\tilde{Y}^{(i)})$$

- Choosing the splitting variable from a random subset helps to decorrelate the trees, that is, $\rho$ is reduced.
- With a suitable value of $m < d$, the bias of each individual model remains roughly the same.
- Overall, EPE can be smaller (with suitable choice of hyper-parameters).

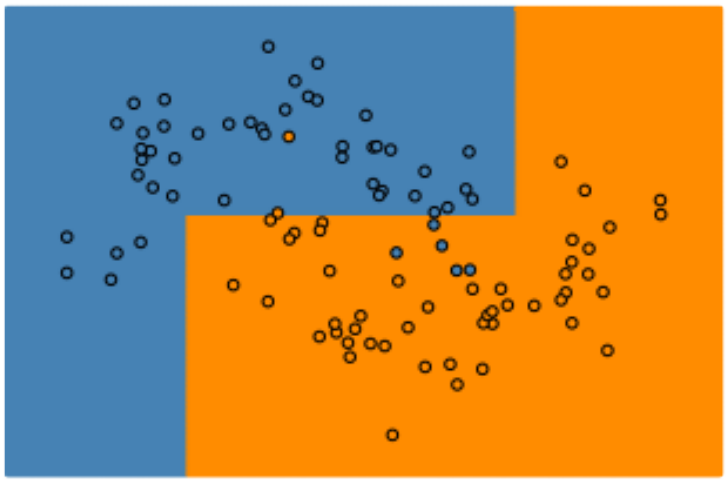
## Choices of $m$ and minimum node size

- Larger $m$: larger variance, smaller bias.
- Larger minimum node size: larger bias, smaller variance.

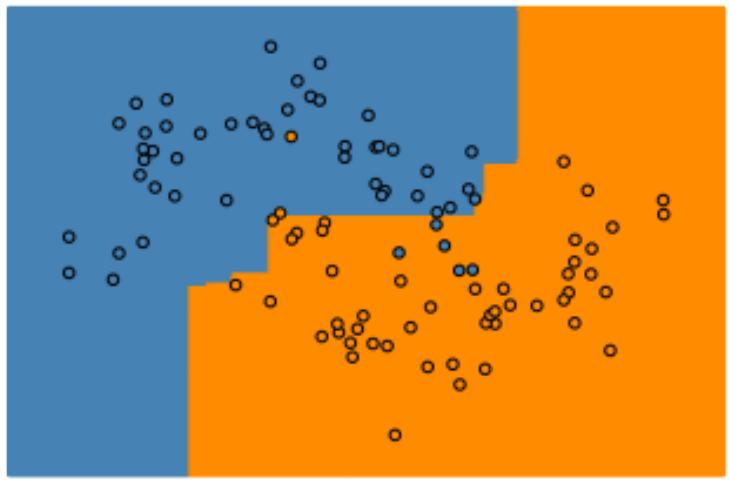
  node size: number of training examples falling into a node
- Recommended heuristics
  - Regression: $m = \lfloor d/3 \rfloor$, minimum node size $= 5$.
  - Classification: $m = \lfloor \sqrt{d} \rfloor$, minimum node size $= 1$.

## RFs for two moons

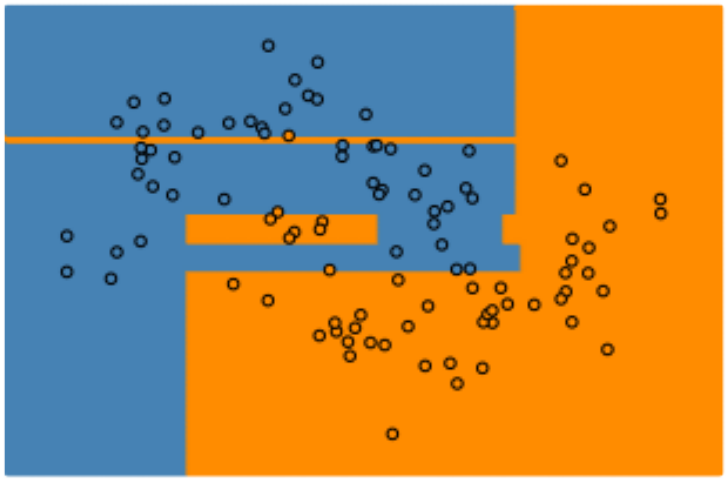
2-level decision tree



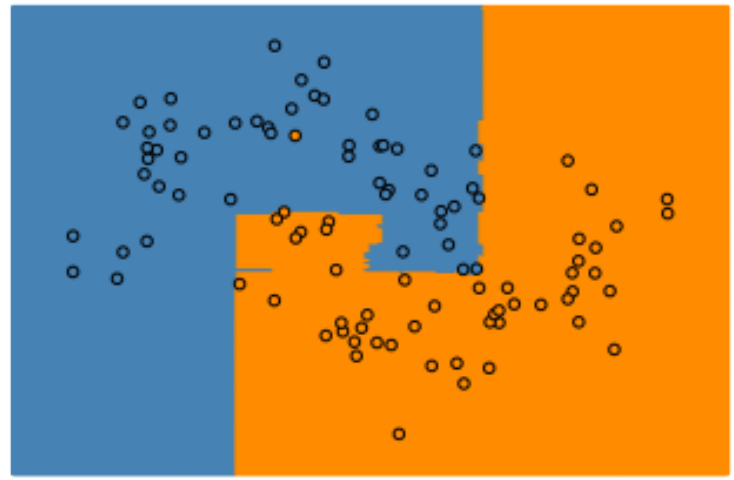
random forest (2-level DT, $B = 100$)



decision tree

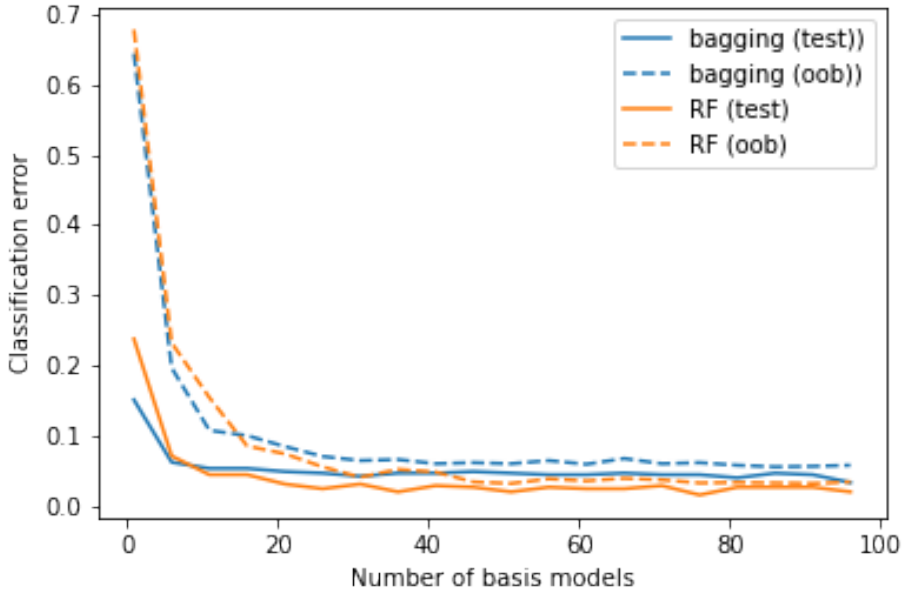


random forest



- A single decision tree's decision boundary is overfitting, while an RF produces a pretty good decision boundary.

## RF for digit recognition




- RF performs better than bagging.

# Boosting

**Weak learner and strong learner**

- Weak learner: a learning algorithm that produces a model slightly better than random guessing.
- Strong learner: a learning algorithm that can produce arbitrary good model.

## Boosting

- Problem: if we have a weak learner, can we use it to build a strong learner?
- Surprisingly, the answer is yes, and an algorithm for converting a weak learner to a strong learner is called a boosting algorithm.
- Many boosting algorithms have been developed.

## How boosting algorithms work

- While bagging and RF are independent ensemble learning methods, boosting is a dependent method (i.e. a basis model is built based on previous models).
- Boosting learns a combination of basis models, by sequentially adding one model at a time, with each new model learned on a training set re-weighted based on previous models.
- We focus on the classical AdaBoost algorithm.

# AdaBoost (Adaptive Boosting)

**Problem**

- Given: a training set $\{(x_1, y_1), \ldots, (x_n, y_n)\} \in \mathcal{X} \times \{-1, +1\}$, and a set $G \subseteq \{-1, +1\}^{\mathcal{X}}$ of simple/weak/basis classifiers.
- AdaBoost produces a score model

$$F(x) = \sum_{t=1}^{T} \alpha_t f_t(x),$$

  where each $\alpha_t$ is non-negative and each $f_t \in G$.
- $F$ classifies an instance $x$ to the class $\text{sgn}(F(x))$.

## AdaBoost

1: $F_0(x) = 0$.
2: Set $w_1(i) = \frac{1}{n}$ for each $i \in [n]$.
3: **for** $t = 1$ to $T$ **do**
4: Train a classifier $f_t \in G$ on the weighted dataset $\{(w_i, x_i, y_i)\}$.
5: $\epsilon_t \leftarrow \sum_{i=1}^{n} w_t(i) I(f_t(x_i) \neq y_i)$.
6: $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$.
7: $F_t \leftarrow F_{t-1} + \alpha_t f_t$.
8: Set $w_{t+1}(i) \propto \exp(-y_i F_t(x_i))$ for each $i \in [n]$.
9: $F \leftarrow F_T / \sum_{t=1}^{T} \alpha_t$.

- Turning a weak learner into a strong one in 9 lines.

## AdaBoost

1: $F_0(x) = 0$.
2: **Set** $w_1(i) = \frac{1}{n}$ **for each** $i \in [n]$.
3: **for** $t = 1$ to $T$ **do**
4: Train a classifier $f_t \in G$ on the weighted dataset $\{(w_i, x_i, y_i)\}$.
5: $\epsilon_t \leftarrow \sum_{i=1}^{n} w_t(i) I(f_t(x_i) \neq y_i)$.
6: $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$.
7: $F_t \leftarrow F_{t-1} + \alpha_t f_t$.
8: Set $w_{t+1}(i) \propto \exp(-y_i F_t(x_i))$ for each $i \in [n]$.
9: $F \leftarrow F_T / \sum_{t=1}^{T} \alpha_t$.

- We start with a null classifier and an equally weighted dataset.

## AdaBoost

1: $F_0(x) = 0$.
2: Set $w_1(i) = \frac{1}{n}$ for each $i \in [n]$.
3: **for** $t = 1$ to $T$ **do**
4: **Train a classifier $f_t \in G$ on the weighted dataset $\{(w_i, x_i, y_i)\}$.**
5: $\epsilon_t \leftarrow \sum_{i=1}^{n} w_t(i) I(f_t(x_i) \neq y_i)$.
6: $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$.
7: $F_t \leftarrow F_{t-1} + \alpha_t f_t$.
8: Set $w_{t+1}(i) \propto \exp(-y_i F_t(x_i))$ for each $i \in [n]$.
9: $F \leftarrow F_T / \sum_{t=1}^{T} \alpha_t$.

- The weak learner is used to obtain a weak classifier $f_t$.
- $f_t$ doesn't need to minimize its error rate on the weighted training set.
- $f_t$ just need to be slightly better than random guessing.

## AdaBoost

1: $F_0(x) = 0$.
2: Set $w_1(i) = \frac{1}{n}$ for each $i \in [n]$.
3: **for** $t = 1$ to $T$ **do**
4: Train a classifier $f_t \in G$ on the weighted dataset $\{(w_i, x_i, y_i)\}$.
5: $\epsilon_t \leftarrow \sum_{i=1}^{n} w_t(i) I(f_t(x_i) \neq y_i)$.
6: $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$.
7: $F_t \leftarrow F_{t-1} + \alpha_t f_t$.
8: Set $w_{t+1}(i) \propto \exp(-y_i F_t(x_i))$ for each $i \in [n]$.
9: $F \leftarrow F_T / \sum_{t=1}^{T} \alpha_t$.

- Compute the error rate $\epsilon_t$ of the weak classifier $f_t$.
- A weak learning algorithm ensures that $\epsilon_t < 1/2$.

## AdaBoost

1: $F_0(x) = 0$.
2: Set $w_1(i) = \frac{1}{n}$ for each $i \in [n]$.
3: **for** $t = 1$ to $T$ **do**
4: Train a classifier $f_t \in G$ on the weighted dataset $\{(w_i, x_i, y_i)\}$.
5: $\epsilon_t \leftarrow \sum_{i=1}^{n} w_t(i) I(f_t(x_i) \neq y_i)$.
6: $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$.
7: $F_t \leftarrow F_{t-1} + \alpha_t f_t$.
8: Set $w_{t+1}(i) \propto \exp(-y_i F_t(x_i))$ for each $i \in [n]$.
9: $F \leftarrow F_T / \sum_{t=1}^{T} \alpha_t$.

- Compute the weight $\alpha_t$ of the weak classifier based on its error rate $\epsilon_t$.
- As $\epsilon_t \to 1/2$, $\alpha_t \to 0$; as $\epsilon_t \to 1$, $\alpha_t \to \infty$.

## AdaBoost

1: $F_0(x) = 0$.
2: Set $w_1(i) = \frac{1}{n}$ for each $i \in [n]$.
3: **for** $t = 1$ to $T$ **do**
4: Train a classifier $f_t \in G$ on the weighted dataset $\{(w_i, x_i, y_i)\}$.
5: $\epsilon_t \leftarrow \sum_{i=1}^{n} w_t(i) I(f_t(x_i) \neq y_i)$.
6: $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$.
7: $F_t \leftarrow F_{t-1} + \alpha_t f_t$.
8: Set $w_{t+1}(i) \propto \exp(-y_i F_t(x_i))$ for each $i \in [n]$.
9: $F \leftarrow F_T / \sum_{t=1}^{T} \alpha_t$.

- Add $f_t$ to the ensemble.

## AdaBoost

1: $F_0(x) = 0$.
2: Set $w_1(i) = \frac{1}{n}$ for each $i \in [n]$.
3: **for** $t = 1$ to $T$ **do**
4: Train a classifier $f_t \in G$ on the weighted dataset $\{(w_i, x_i, y_i)\}$.
5: $\epsilon_t \leftarrow \sum_{i=1}^{n} w_t(i) I(f_t(x_i) \neq y_i)$.
6: $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$.
7: $F_t \leftarrow F_{t-1} + \alpha_t f_t$.
8: **Set $w_{t+1}(i) \propto \exp(-y_i F_t(x_i))$ for each $i \in [n]$.**
9: $F \leftarrow F_T / \sum_{t=1}^{T} \alpha_t$.

- $(x, y)$ is correctly classified by $F_t$ if $yF_t(x) > 0$.
  - large positive $yF_t(x)$ means $(x, y)$ is easy for $F_t$
  - large negative $yF_t(x)$ means $(x, y)$ is hard for $F_t$
- Essentially, the weight update assigns hard examples larger weights.
- The weight can be recursively updated as $w_{t+1}(i) \propto w_t(i) \exp(-y_i f_t(x_i))$, and they are often normalized to sum to 1.

## AdaBoost

1: $F_0(x) = 0$.
2: Set $w_1(i) = \frac{1}{n}$ for each $i \in [n]$.
3: **for** $t = 1$ to $T$ **do**
4: Train a classifier $f_t \in G$ on the weighted dataset $\{(w_i, x_i, y_i)\}$.
5: $\epsilon_t \leftarrow \sum_{i=1}^{n} w_t(i) I(f_t(x_i) \neq y_i)$.
6: $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$.
7: $F_t \leftarrow F_{t-1} + \alpha_t f_t$.
8: Set $w_{t+1}(i) \propto \exp(-y_i F_t(x_i))$ for each $i \in [n]$.
9: $F \leftarrow F_T / \sum_{t=1}^{T} \alpha_t$.

- In the final score function $F$, the weights of the weak classifiers are normalized to sum to 1.

## Bound on training error

The error rate of $F_T$ is upper bounded as follows

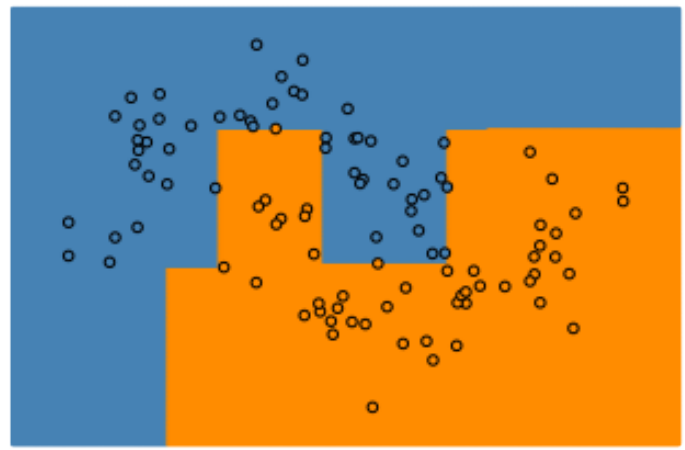
$$\frac{1}{n}\sum_{i=1}^{n} I(\text{sgn}(F_T(x_i)) \neq y_i) \leq \prod_{t=1}^{T} \sqrt{1-4\gamma_t^2},$$

where $\gamma_t = \frac{1}{2} - \epsilon_t$ (how much $f_t$ is better than random guessing).

## AdaBoost with 2-level DTs

RF

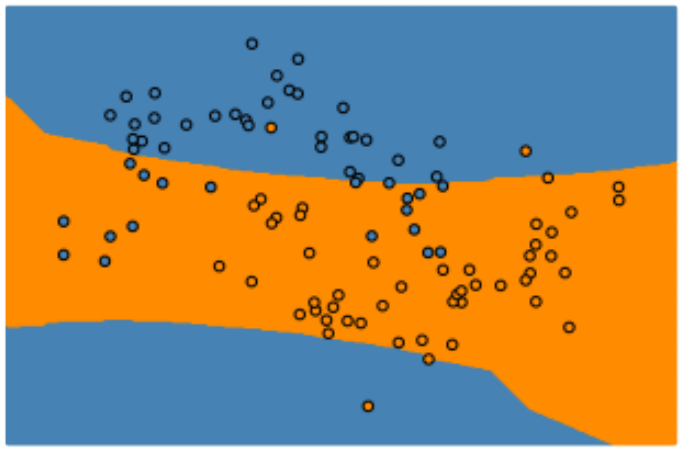
AdaBoost



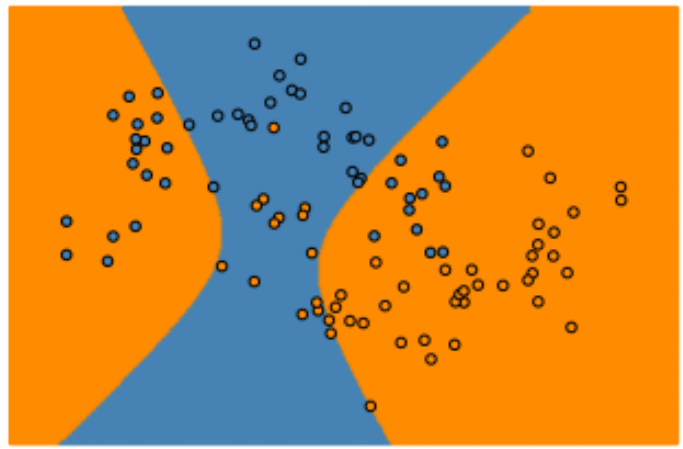
- Both RF and AdaBoost use 2-level DTs as basis models.
- AdaBoost finds a much better decision boundary than RF in this case.

## AdaBoost with SVMs with quadratic kernels

bagging

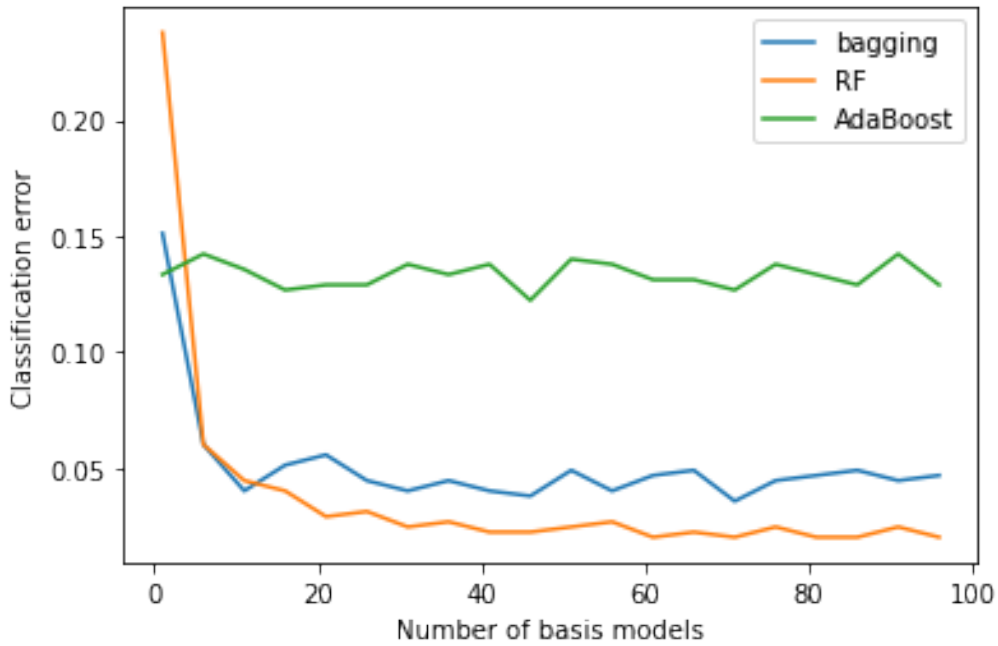


AdaBoost



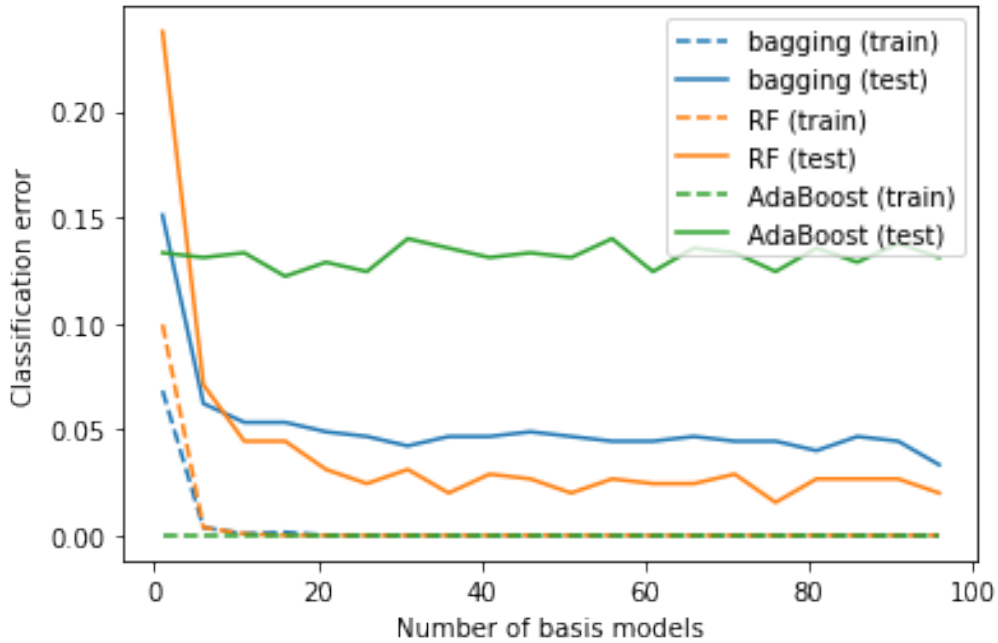
- Bagging's decision boundary is roughly hyperbolic.
- AdaBoost's decision boundary is also roughly hyperbolic, and has a different orientation.

## Comparing AdaBoost, bagging and RF on the digit dataset



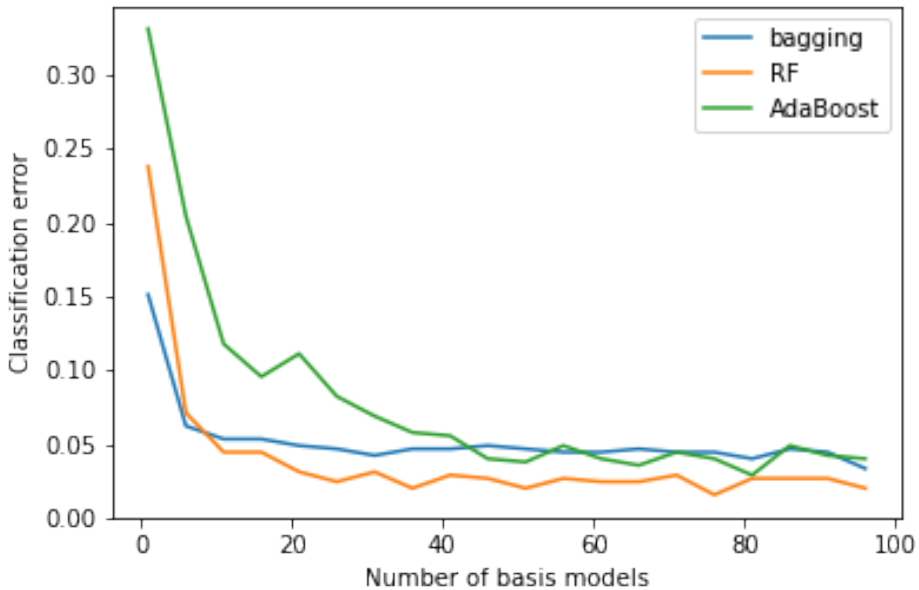

- DTs are used as the basis models.
- Both RF and bagging are much better than AdaBoost.

## Not the end of story yet...




- AdaBoost has zero training error with just one model!
- AdaBoost has overfitted.

## Fixing AdaBoost's performance




- If we use 5-level DTs, AdaBoost has a similar test set performance as RF.
- AdaBoost is more prone to overfitting if the basis model is too rich.

# Beyond Binary Classification

- Boosting algorithms have been designed to handle problems other than binary classification
  - Multi-class classification
  - Regression
- sklearn and XGBoost contains well-documented implementations of boosting algorithms.

# Checking Your Understanding

Which of the following statement is correct? (Multiple choice)

**(a)** AdaBoost is an independent method for ensemble learning.

**(b)** Random forest is an improved bagging algorithm.

**(c)** Random forest constructs decorrelated trees.

# What You Need to Know

- Ensembles can be much more expressive than basis models.
- Two approaches: independent methods, dependent methods
- Bagging: algorithm, model selection, why it works
- Random forest: algorithm, why it works, hyperparameters
- Boosting: weak learning and strong learning, AdaBoost
- Tuning ensemble methods: class of basis models, size of ensemble