COMP47750

Ensembles

Pádraig Cunningham

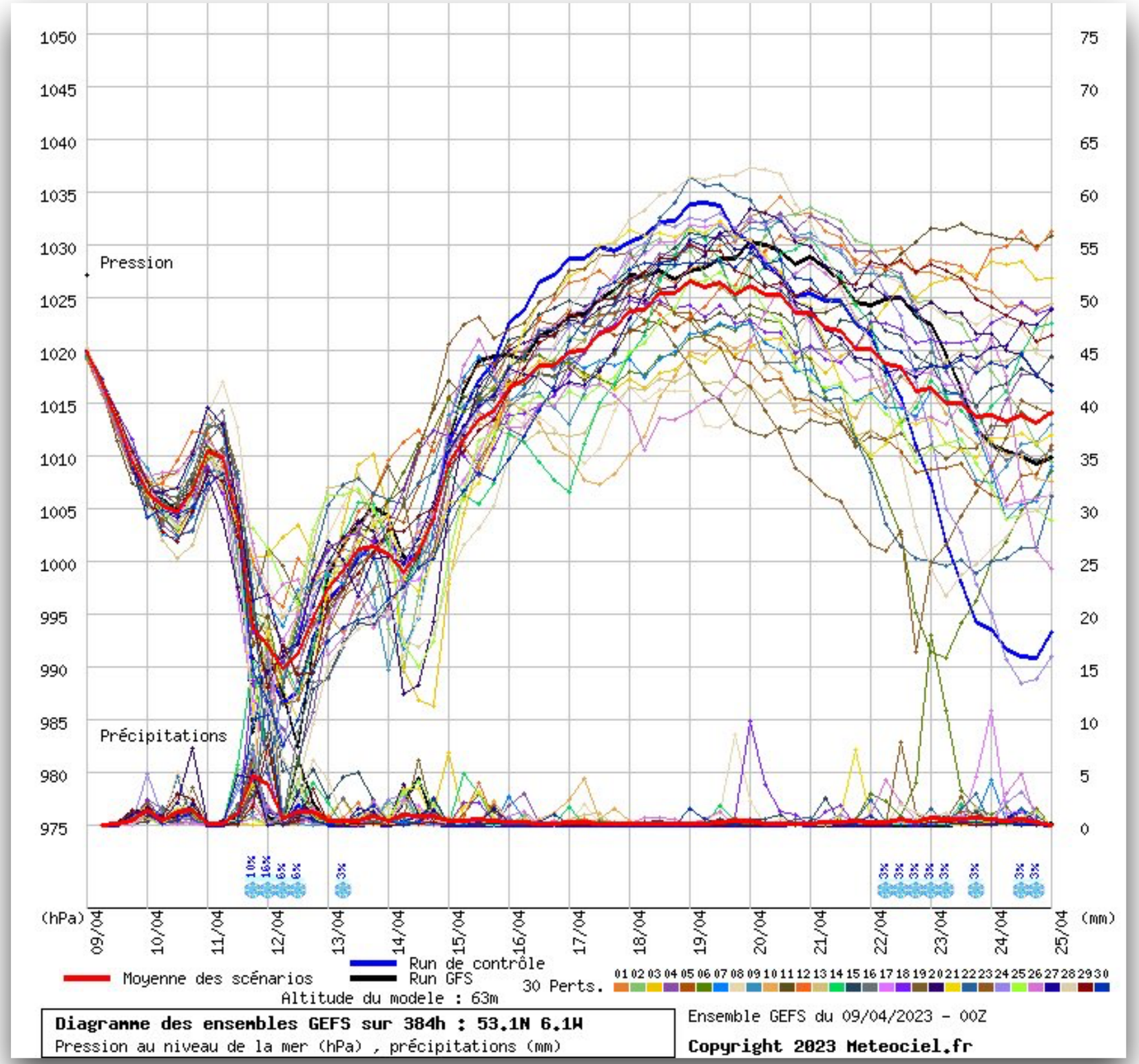
School of Computer Science

© UCD Computer Science

Weather Ensembles

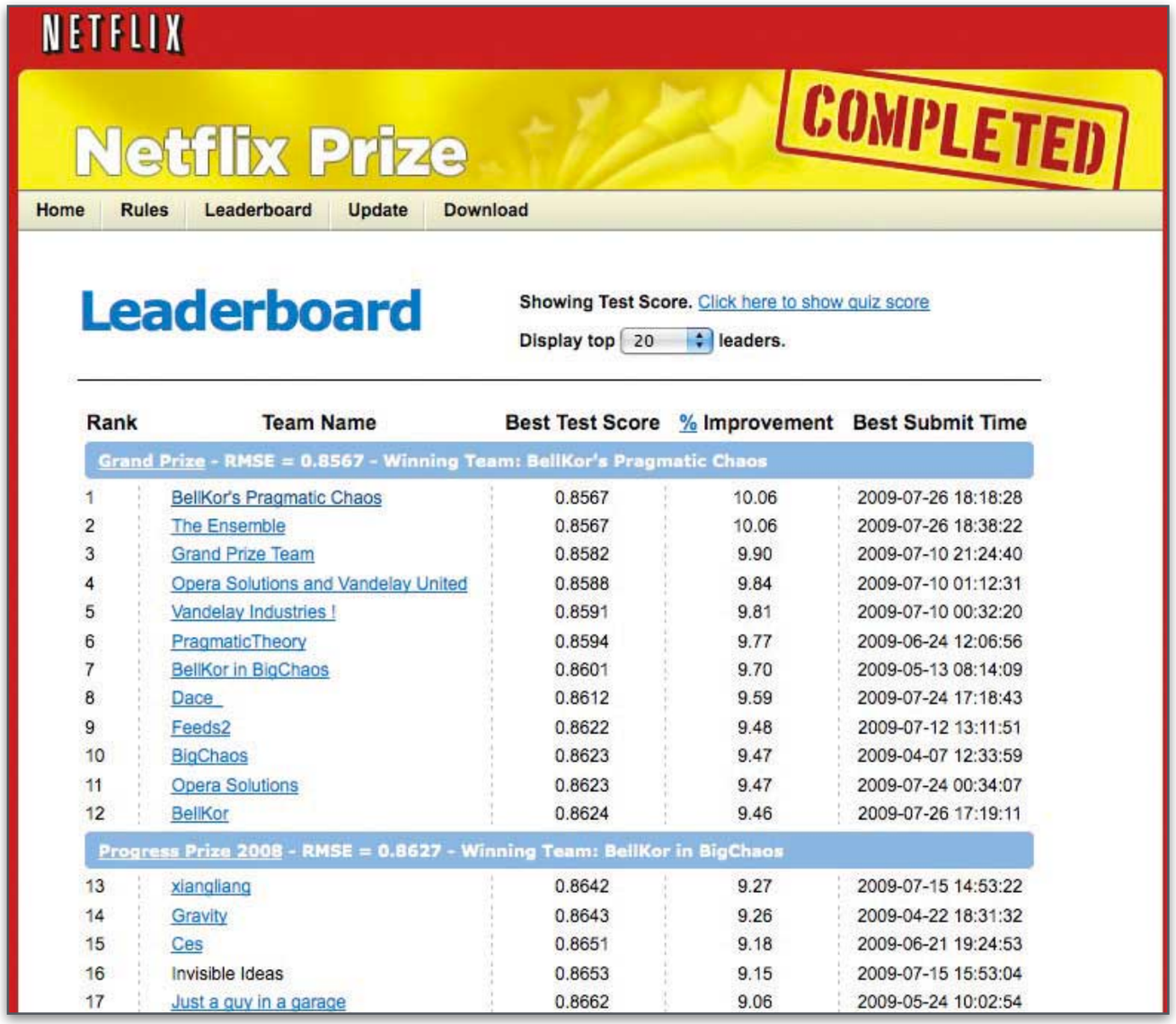
- Build an ensemble of models:
  - ‘Mean’ prediction more accurate
  - Level of ‘agreement’ an indicator of confidence
  - Shows ‘spread’ of possible outcomes

- https://www.meteociel.fr/
- Twitter: @CarlowWeather



Introduction

In 2006, Netflix announced a machine learning competition for movie rating prediction. Prize of $1 million to whoever improved the accuracy of existing system by 10%.



NETFLIX

Netflix Prize

COMPLETED

Home | Rules | Leaderboard | Update | Download

Leaderboard

Showing Test Score. Click here to show quiz score

Display top 20 leaders.

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	2009-07-10 00:32:20
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace	0.8612	9.59	2009-07-24 17:18:43
9	Feeds2	0.8622	9.48	2009-07-12 13:11:51
10	BigChaos	0.8623	9.47	2009-04-07 12:33:59
11	Opera Solutions	0.8623	9.47	2009-07-24 00:34:07
12	BellKor	0.8624	9.46	2009-07-26 17:19:11
Progress Prize 2008 - RMSE = 0.8627 - Winning Team: BellKor in BigChaos				
13	xiangliang	0.8642	9.27	2009-07-15 14:53:22
14	Gravity	0.8643	9.26	2009-04-22 18:31:32
15	Ces	0.8651	9.18	2009-06-21 19:24:53
16	Invisible Ideas	0.8653	9.15	2009-07-15 15:53:04
17	Just a guy in a garage	0.8662	9.06	2009-05-24 10:02:54

Top submissions all combine several teams and algorithms as an ensemble.

BellKor Team:
"Our final solution consists of blending 107 individual results"

http://techblog.netflix.com/2012/04/netflix-recommendations-beyond-5-stars.html

© UCD Computer Science

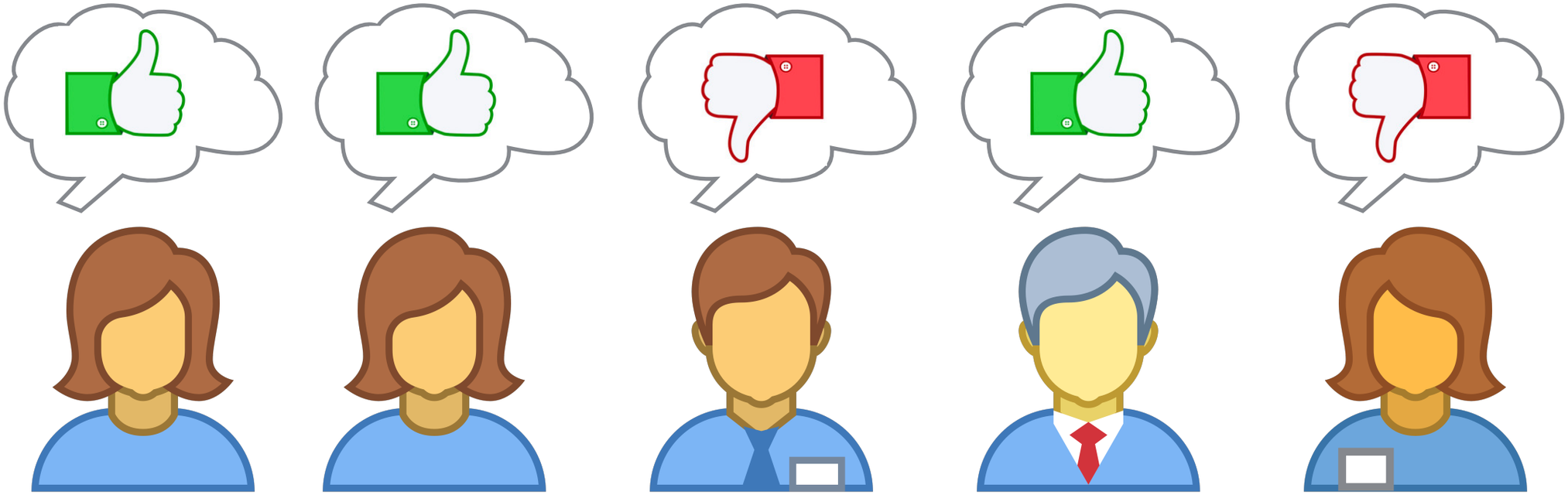
Ensembles

- Introduction to Ensembles
- Why Do Ensembles Work?
- Ensemble Generation
  - Bagging
  - Boosting
- Ensemble Combination
  - Voting v Weighted Voting
- Random Forest
  - Feature Importance
- Bias/Variance Decomposition of Error
- Ensemble Classification in *scikit learn*

© UCD Computer Science

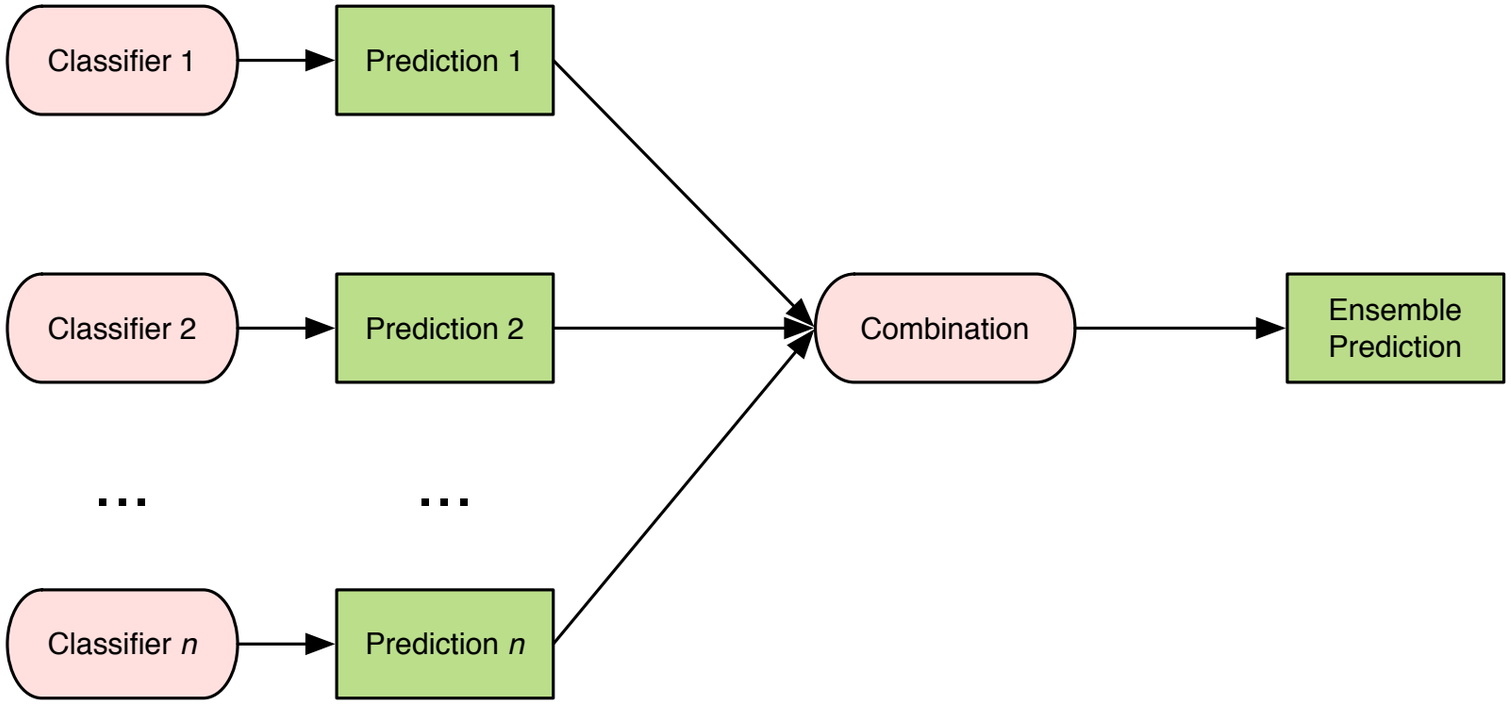
Introduction

- In many tasks, we often ask for a second opinion when making a decision (e.g. medical diagnosis).
- Some times we might seek many opinions before arriving at a final decision (e.g. a committee).
- Why not apply the same idea when making predictions in machine learning?



Introduction

- Ensemble Classification: Combine individual predictions from multiple classifiers in order to make more accurate predictions.



- An ensemble of “weak learners” can provide a strong committee.
  - Weak learner: simple model - slightly better than random chance
- Applied using many different types of classifiers - e.g decision trees, *k*-NNs, neural networks, support vector machines…

© UCD Computer Science

Why Do Ensembles Work?

The Condorcet Jury Theorem

- Proposed by the Marquis of Condorcet in 1784, and relates to the relative probability of an ensemble of individuals arriving at a correct decision.
- If each voter has a probability *p* of being correct and the probability of a majority of voters being correct is *M*…
  - Then $p > 0.5$ implies $M > p$
  - Also if *p* always > 0.5, then *M* approaches 1.0 as the number of voters approaches infinity.

➡ "When the average probability of an individual being correct is > 50%, the chance of the ensemble of them reaching the correct decision increases as more members are added".

© UCD Computer Science

Why Do Ensembles Work?

- Condorcet Jury Theorem revisited…
  - We now know that *M* will be greater than *p* only if there is diversity in the pool of voters - i.e. there is some disagreement between their decisions.
  - The probability of a majority of voters being correct will increase as the ensemble grows only if the diversity in the ensemble continues to grow as well.
- Eventually, new ensemble members will have voting patterns collinear with existing members - i.e. they will vote the same way.
- Typically the diversity of the ensemble will plateau as will the accuracy of the ensemble at some size between 10-50 members.

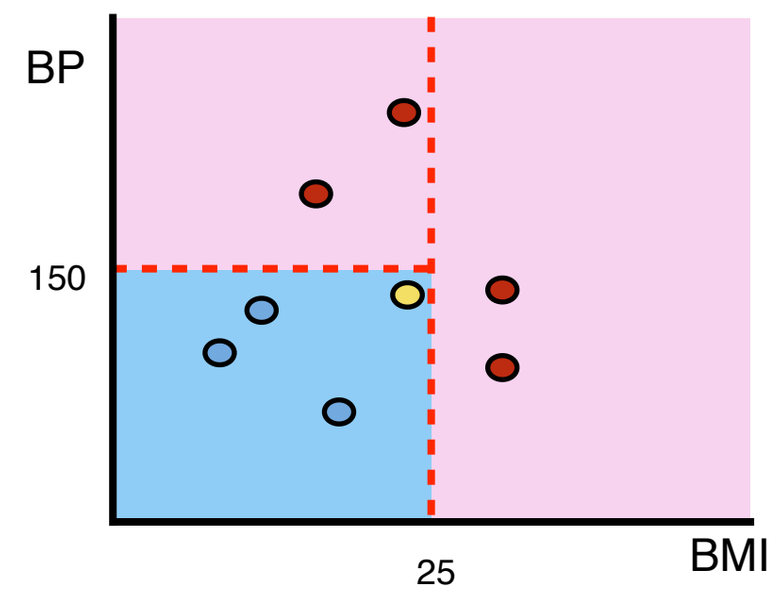
© UCD Computer Science

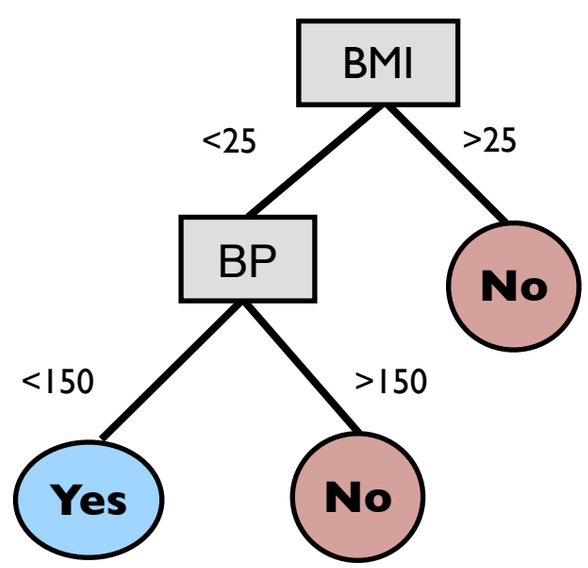
Example: Classification

- **Data:** Heart attack patient admitted. 19 variables measured during first 24 hours. Blood pressure, age, BMI + 16 other variables, considered important indicators of patient's condition.
- **Task:** Identify high risk patients (i.e. will not survive 30 days), based on evidence of initial 24-hour data.

No	Age	BMI	BP	Ok?
1	60	20	140	Yes
2	60	21	145	Yes
3	85	23	130	Yes
4	81	22	160	No
5	70	24	170	No
6	72	26	135	No
7	81	26	145	No
8	66	23	155	No

Q	66	24	148	?



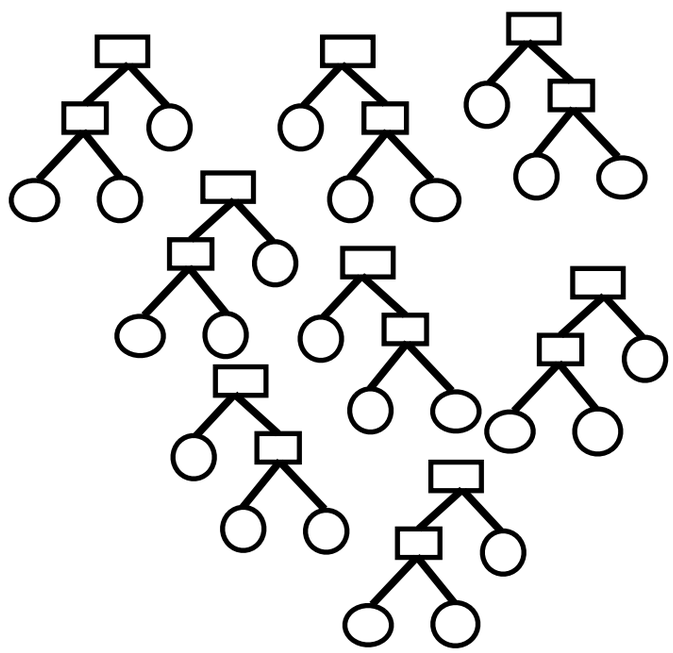


© UCD Computer Science

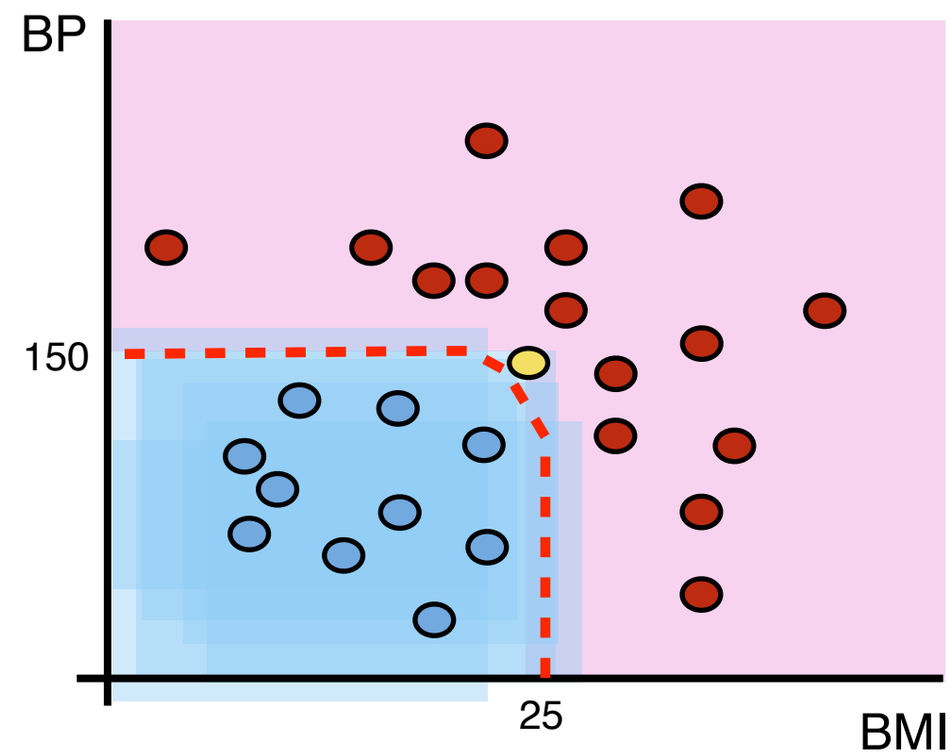
Ensemble Idea

- Build many “base” decision trees, using different subsets of the data. These trees can vote on the class of a new input example.

➡ Accuracy of ensemble should be better than the individual trees.



Ensemble of Decision Trees



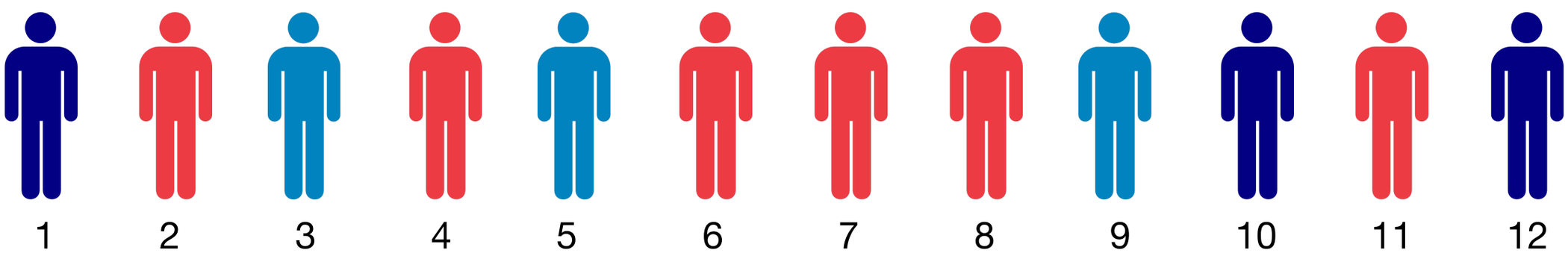
Q. How do we <u>generate</u> base classifiers that complement each other?

Q. How do we <u>combine</u> the outputs of base classifiers to maximise accuracy?

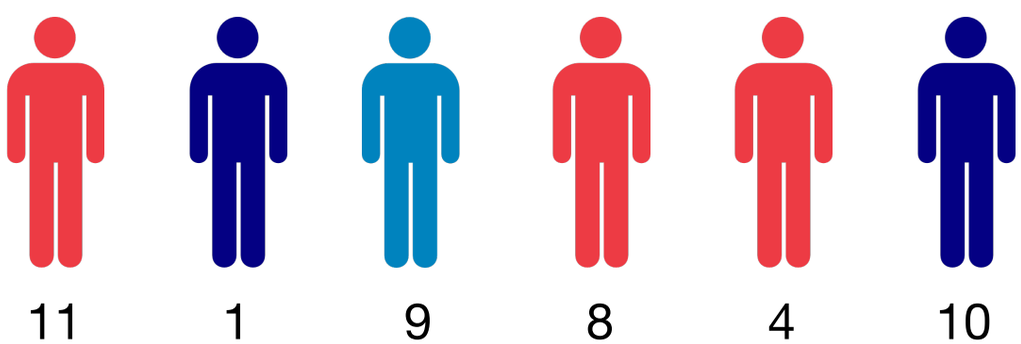
© UCD Computer Science

Random Sampling

- Simple random sampling: A subset of items is chosen from a larger population. Each item is chosen by chance, and each member of the population has an equal probability of being chosen.
- The idea is that the randomly selected sample has similar characteristics to the overall population from which it was chosen.
- If we sample without replacement, we deliberately avoid choosing any item from the population more than once.

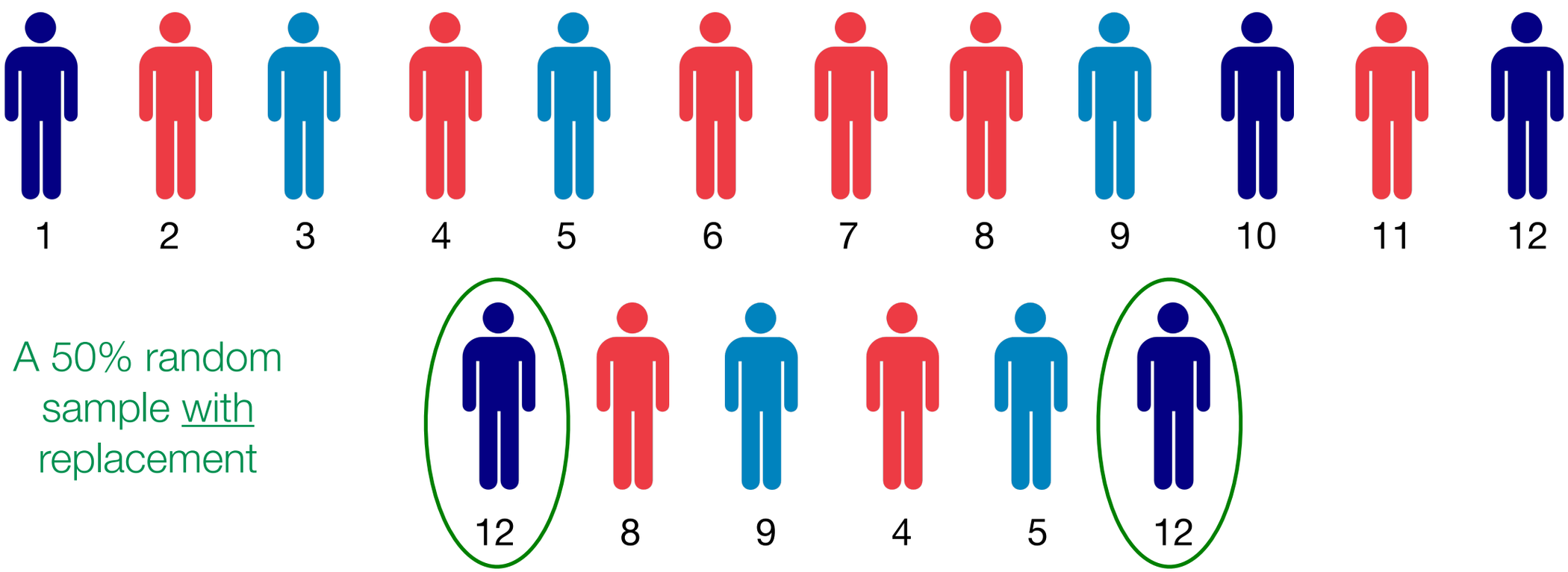


A 50% random sample without replacement



Random Sampling

- Simple random sampling: A subset of items is chosen from a larger population. Each item is chosen by chance, and each member of the population has an equal probability of being chosen.
- The idea is that the randomly selected sample has similar characteristics to the overall population from which it was chosen.
- We can also sample with replacement, where the same item can potentially appear more than once in the same sample.



Ensemble Generation: Bagging

- **Key Idea:** Train *n* classifiers on different subsets of the training data.
- Bootstrap aggregation / Bagging (Breiman, 1996):
  - Randomly sample from training data <u>with replacement</u>.
  - 100% bootstrap sample will contain ~63% of training examples. Remaining data is "out-of-bag" (OOB).

Complete dataset has 8 examples

Original	A	B	C	D	E	F	G	H
Set 1	B	G	H	C	G	F	C	A
Set 2	G	H	E	F	D	B	G	A
Set 3	C	F	B	G	E	F	B	B
Set 4	D	E	A	D	E	D	C	H
Set 5	E	F	A	C	E	F	A	H
Set 6	C	H	B	F	D	B	H	F

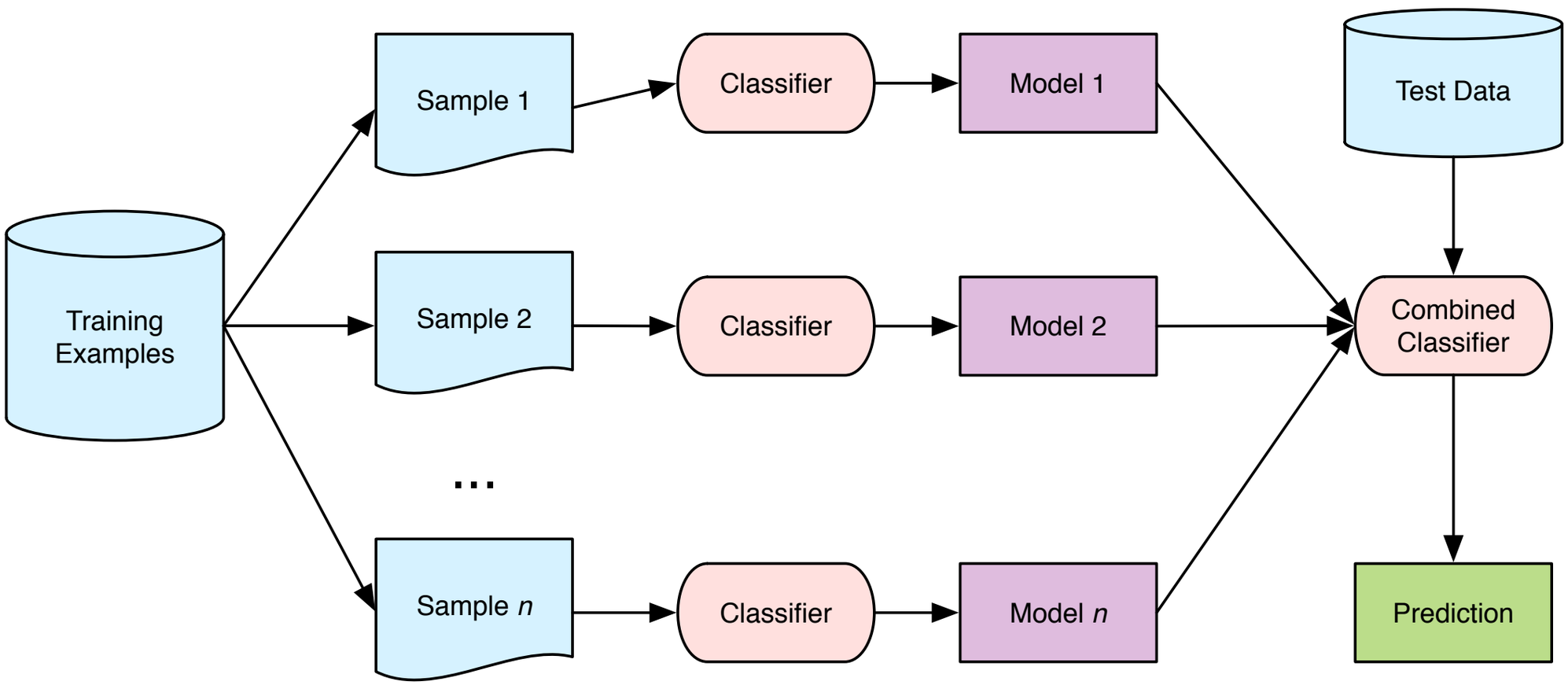
Each bootstrap subset has 8 examples.

Some examples may be duplicated, others left out.

© UCD Computer Science

Bagging

- Bootstrap aggregation: Randomly sample from training data <u>with replacement</u>, apply a classifier to each sample.



➡ Encourages diversity in the ensemble, works better for “unstable” classifiers - e.g. decision trees, neural networks.

Feature Subspaces

- Previously we have seen...
  - A feature space is a *D*-dimensional coordinate space used to represent the input examples for a given problem, with one coordinate for each feature in the dataset.
  - Feature selection involves selecting a subset of the original features (i.e. a feature subspace), on which to apply a classifier.

	region-centroid-col	region-centroid-row	region-pixel-count	vedge-mean	vedge-sd	hedge-mean	hedge-sd	intensity-mean
x1	218.0	178.0	9.0	0.8	0.5	1.1	0.5	59.6
x2	113.0	130.0	9.0	0.3	0.3	0.3	0.4	0.9
x3	202.0	41.0	9.0	0.9	0.8	1.1	1.0	123.0
x4	32.0	173.0	9.0	1.7	1.8	9.0	6.7	43.6
x5	61.0	197.0	9.0	1.4	1.5	2.6	1.9	49.6
x6	149.0	185.0	9.0	1.6	1.1	3.1	1.9	49.3
...	...	...	...	...	...	...	...	...

4-dimensional feature subspace of the original 8-dimensional dataset (i.e. 50% of the original features)

	region-centroid-row	hedge-mean	hedge-sd	intensity-mean
x1	178.0	1.1	0.5	59.6
x2	130.0	0.3	0.4	0.9
x3	41.0	1.1	1.0	123.0
x4	173.0	9.0	6.7	43.6
x5	197.0	2.6	1.9	49.6
x6	185.0	3.1	1.9	49.3
...	...	...	...	...

© UCD Computer Science

Random Subspacing

- **Key Idea:** Train *n* base classifiers, each on a different subset of features.
- Random Subspace Method:
  - A subset of features is randomly selected without replacement.
  - Train a classifier using only selected features to represent the training data.

  ➡ Encourages diversity in the ensemble, works well for *k*-NNs.

Original dataset (5 features)

X	Col	H	W	Wgt	HW
1	210	60	62	186	0.97
2	220	70	53	180	1.32
3	215	55	50	152	1.10
4	180	76	40	152	1.90
...	...	...	...	...	...
10	190	60	58	175	1.03

Feature Subset 1 (3 features)

X	H	W	HW
1	60	62	0.97
2	70	53	1.32
3	55	50	1.10
4	76	40	1.90
...	...	...	...
10	60	58	1.03

Classifier → Model 1

Feature Subset 2 (3 features)

X	Col	H	HW
1	210	60	3.50
2	220	70	3.14
3	215	55	3.91
4	180	76	2.37
...	...	...	...
10	190	60	3.17

Classifier → Model 2

Feature Subset 3 (3 features)

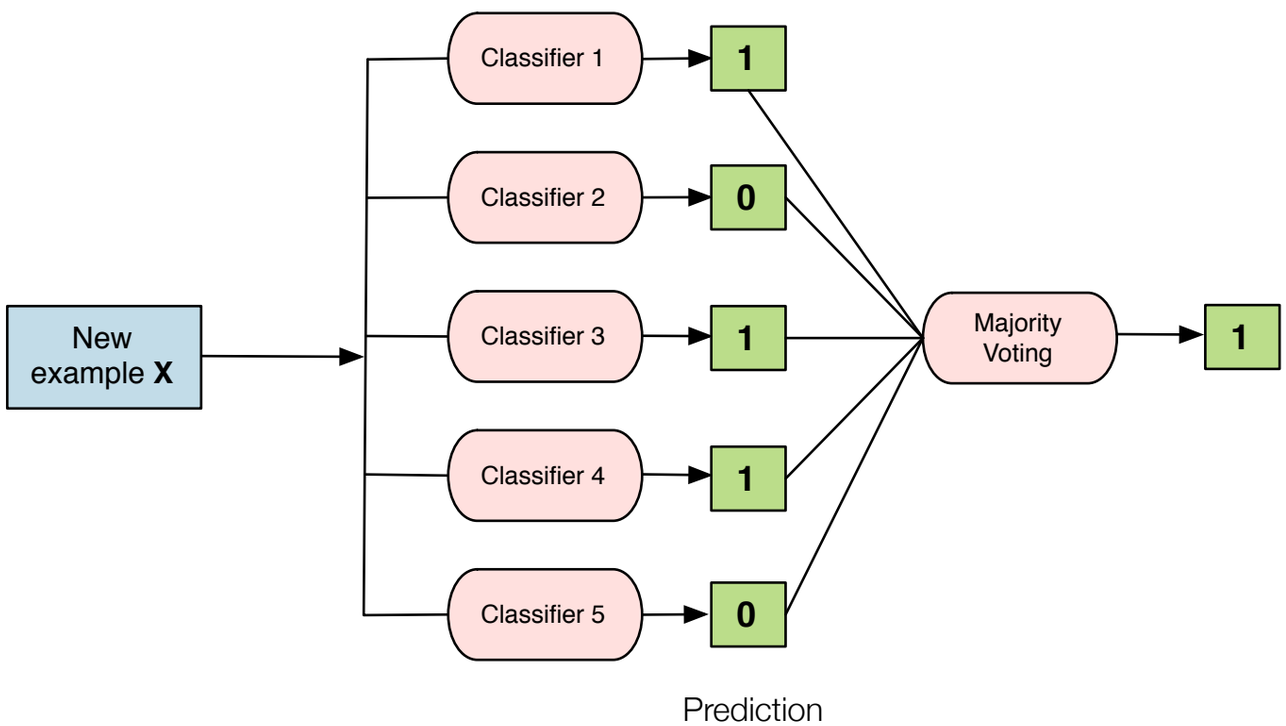
X	Col	H	W
1	210	60	62
2	220	70	53
3	215	55	50
4	180	76	40
...	...	...	...
10	190	60	58

Classifier → Model 3

© UCD Computer Science

Ensemble Combination: Voting

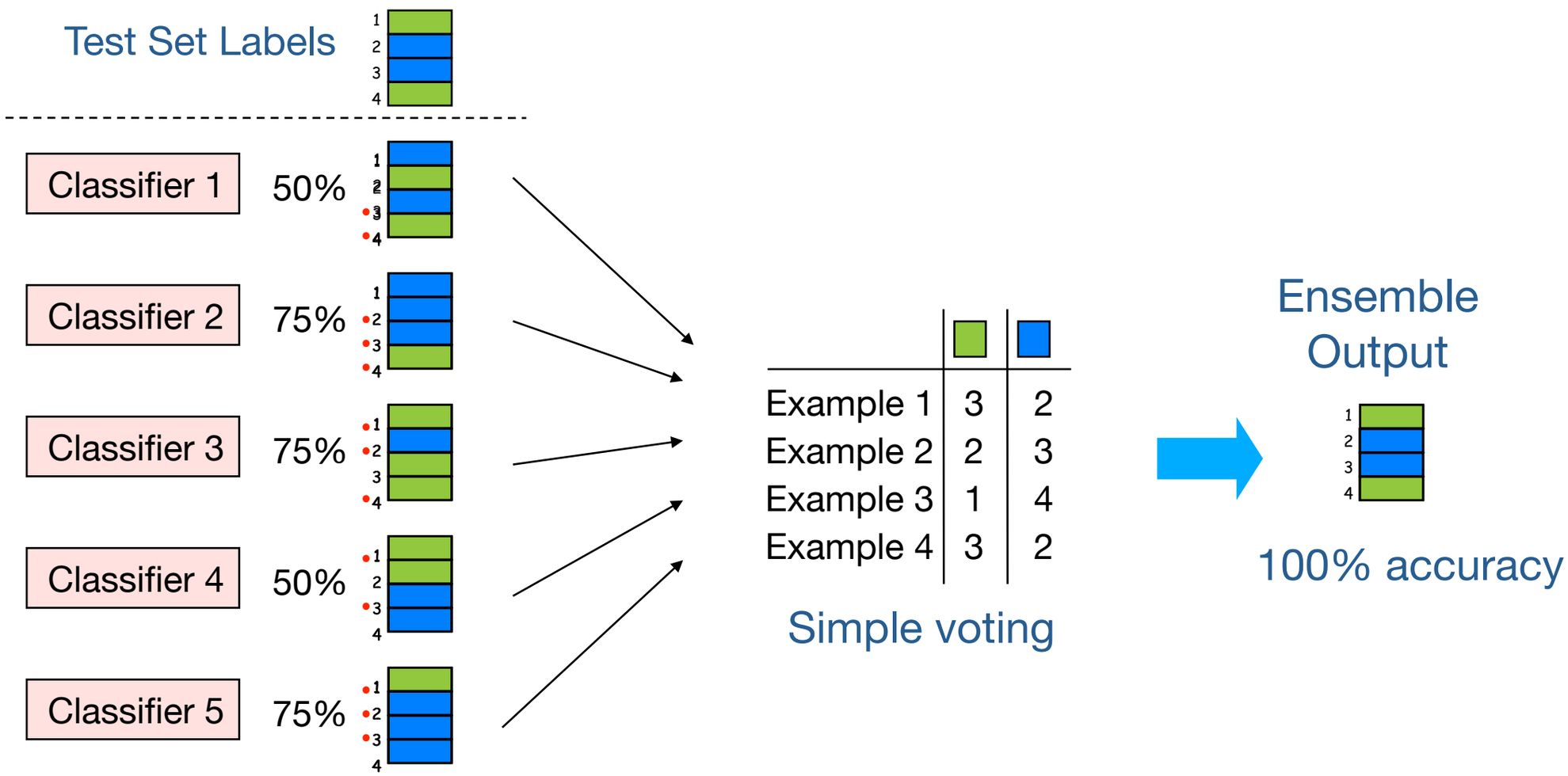
- Simplest way to combine the output of multiple classifiers is to use majority voting.
- All classifiers are run independently in parallel. Results are combined when all runs have completed.
- Each classifier "votes" for a particular class, where all classifiers carry equal weight. The class with the majority vote in the ensemble wins.



Ensemble votes for label "1" for **X** by 3:2

Ensemble Diversity

- **Local Learning in Ensembles**
  - Every single classifier performs well on a subset of the test set.
  - The mistakes that one classifier makes are “corrected” by the other classifiers.



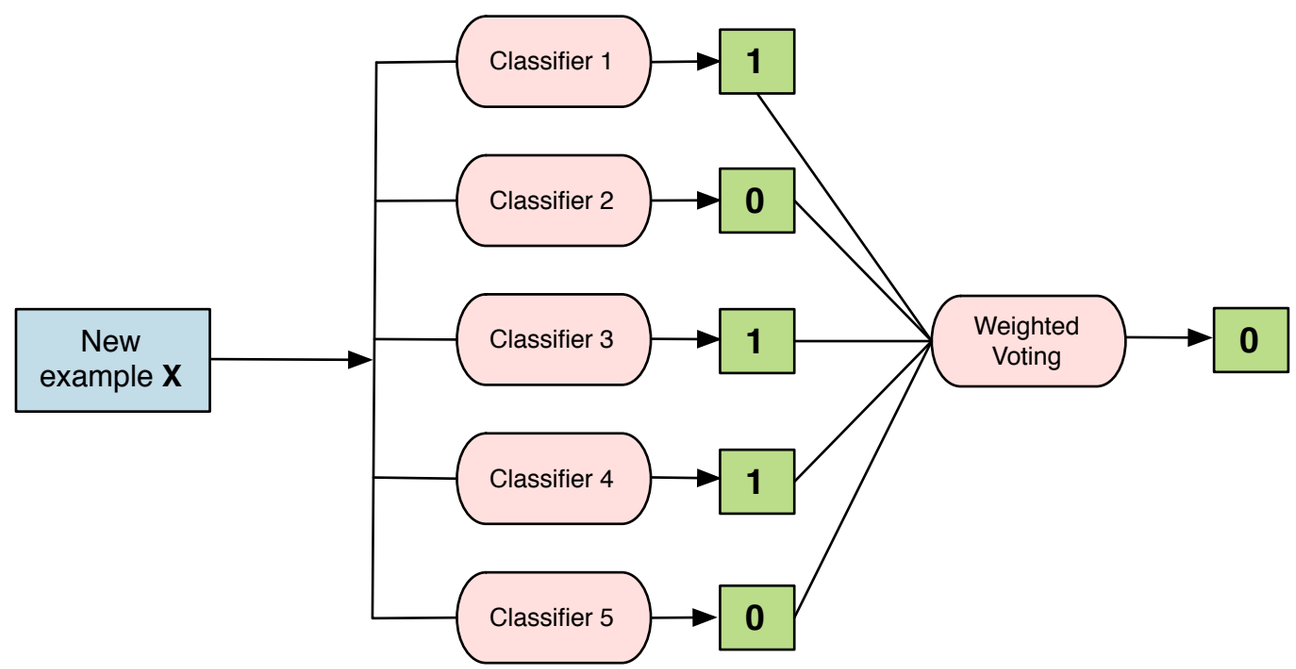
Ensemble Combination: Weighted Voting

- **Intuition:** If individual classifiers do not give equal performance, we should give more influence to better classifiers.
- Weighted Voting Combination:
  - Rather than treating every classifier's vote equally, we weight each classifier's vote based on its accuracy/error.

➡ More accurate classifiers contribute more to the ensemble.

Classifier	Accuracy	Weight
1	0.52	0.14
2	1.00	0.28
3	0.57	0.16
4	0.55	0.15
5	0.95	0.26
TOTAL	**3.59**	**1.00**

e.g. C1: 0.52/3.59 = 0.14



Vote "0": 0.28 + 0.26 ≅ 0.54

Vote "1": 0.14 + 0.16 + 0.15 ≅ 0.46

© UCD Computer Science

Committees of Experts

- **Consider:** *" … a medical school that has the objective that all students, given a problem, come up with an identical solution"*.
- No value in a committee of experts from such a group - the committee will not improve on the judgement of an individual.
- There needs to be disagreement for the committee to have the potential to be better than an individual.

- Fundamental work by Krogh & Vedelsby (1995) for regression:
  - Increasing "ambiguity" (disagreement) decreases overall combined error, provided it does not result in an increase of average error.

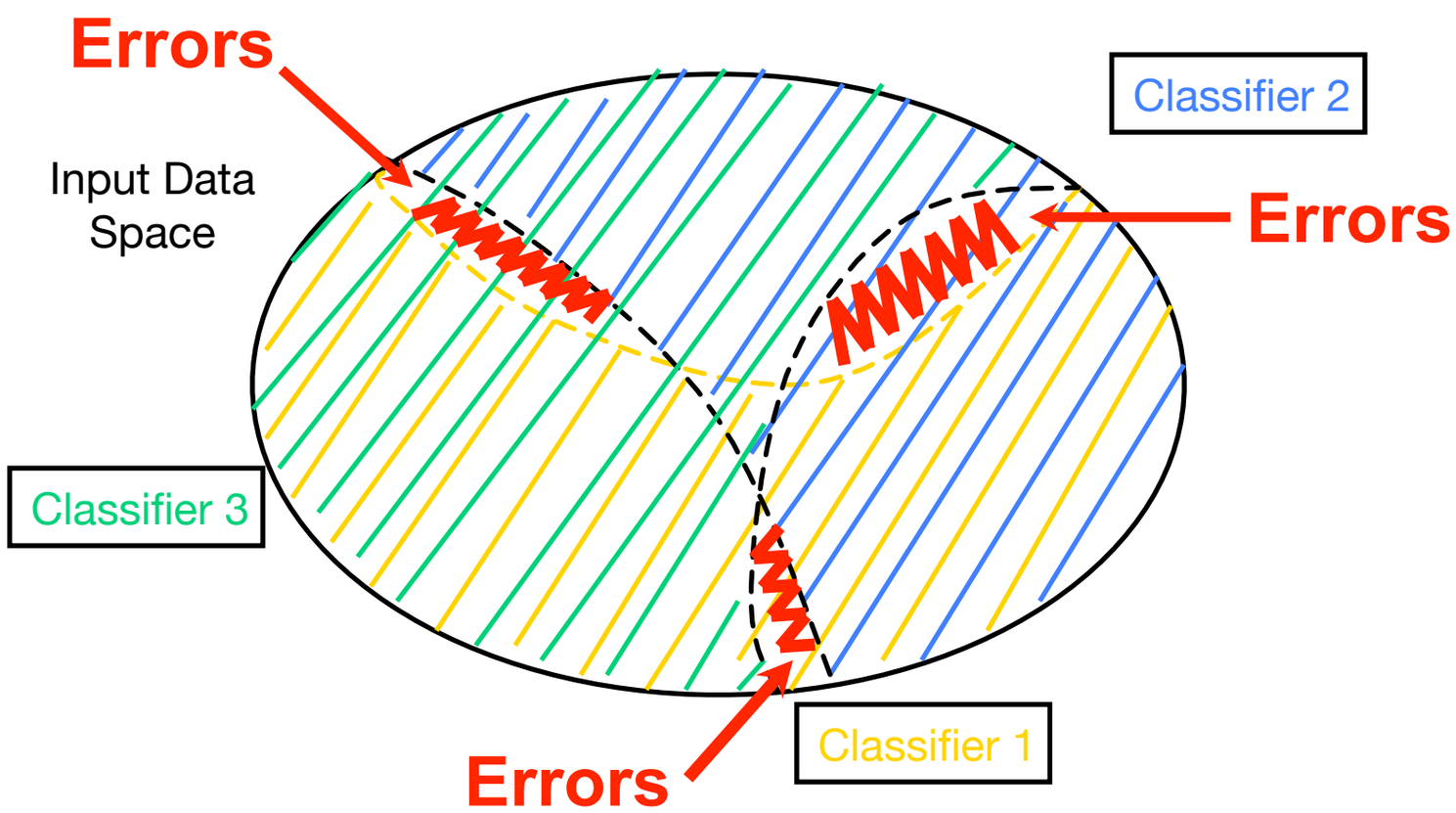
$$\overline{E} - \overline{A} = E$$

Average error of classifiers — Ensemble ambiguity — Ensemble error

We need accuracy + diversity in classifier ensembles!

Ensemble Specialisation

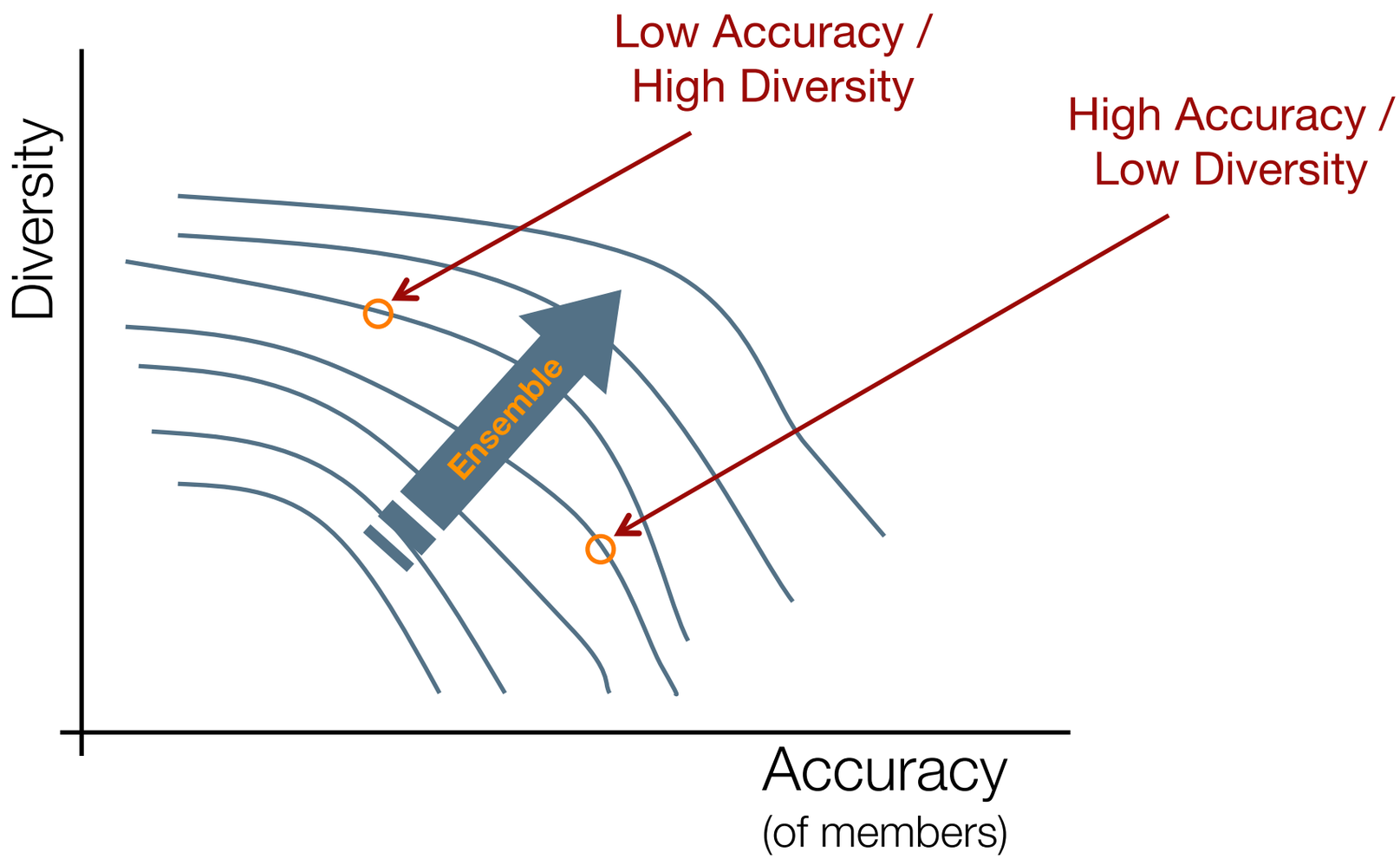
- Classifiers can specialise on accurately classifying only related examples from certain regions of the input data space.
- **Example:** Visualisation of specialisation in classifier ensembles…



➡ Want ensemble members to make mistakes in different areas.

Accuracy v Diversity Trade-Off

- Krogh & Vedelsby said an ideal ensemble is one that consists of highly accurate members which at the same time disagree.
- We often face a trade-off between diversity and accuracy when constructing an ensemble of classifiers.



Python code notebook 15 Ensembles

```
tree_bag = BaggingClassifier(dtree,
                             n_estimators = 10,
                             max_samples = 1.0,
                             bootstrap = True)
```

Bagging
10 ensemble members
Sampling with replacement

```
random_SS_kNN = BaggingClassifier(kNN,
                                  n_estimators = 10,
                                  max_samples=1.0,
                                  max_features=0.5)
```

Random Subspace

50% of features

COMP47750

Ensembles

Part III
Random Forest
Boosting

Pádraig Cunningham

School of Computer Science
© UCD Computer Science

Random Forests

Machine Learning, 45, 5–32, 2001
© 2001 Kluwer Academic Publishers. Manufactured in The Netherlands.

Random Forests

LEO BREIMAN
Statistics Department, University of California, Berkeley, CA 94720

Editor: Robert E. Schapire

Abstract. Random forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. The generalization error for forests converges a.s. to a limit as the number of trees in the forest becomes large. The generalization error of a forest of tree classifiers depends on the strength of the individual trees in the forest and the correlation between them. Using a random selection of features to split each node yields error rates that compare favorably to Adaboost (Y. Freund & R. Schapire, *Machine Learning*: *Proceedings of the Thirteenth International conference*, ∗ ∗ ∗, 148–156), but are more robust with respect to noise. Internal estimates monitor error, strength, and correlation and these are used to show the response to increasing the number of features used in the splitting. Internal estimates are also used to measure variable importance. These ideas are also applicable to regression.

Keywords: classification, regression, ensemble

1. Random forests

1.1. Introduction

Significant improvements in classification accuracy have resulted from growing an ensemble of trees and letting them vote for the most popular class. In order to grow these ensembles, often random vectors are generated that govern the growth of each tree in the ensemble. An early example is bagging (Breiman, 1996), where to grow each tree a random selection (without replacement) is made from the examples in the training set.

Random forests
L Breiman - Machine learning, 2001 - Springer
… The simplest **random forest** with **random** features is formed by selecting at **random**, at each node, a small group of input variables to split on. Grow the tree using CART methodology to …
☆ Save 99 Cite Cited by 141021 Related articles All 84 versions

~140,000 citations Nov. 2024

Random Forests

- Combination of Bagging and Random Subspacing
  - kinda

- Basic strategy - build loads of trees (100-1,000)
  - For each ensemble member:

1. The training set X is sub-sampled with replacement to produce a training set of size $|X|$ (as in bagging)

Not exactly Random Subspacing

2. Where D is the set of features that describes the data, $m << |D|$ is selected as the number of features to be used in the feature selection process. At each stage (i.e. node) in the building of a tree m features are selected at random to be the candidates for splitting at that node.

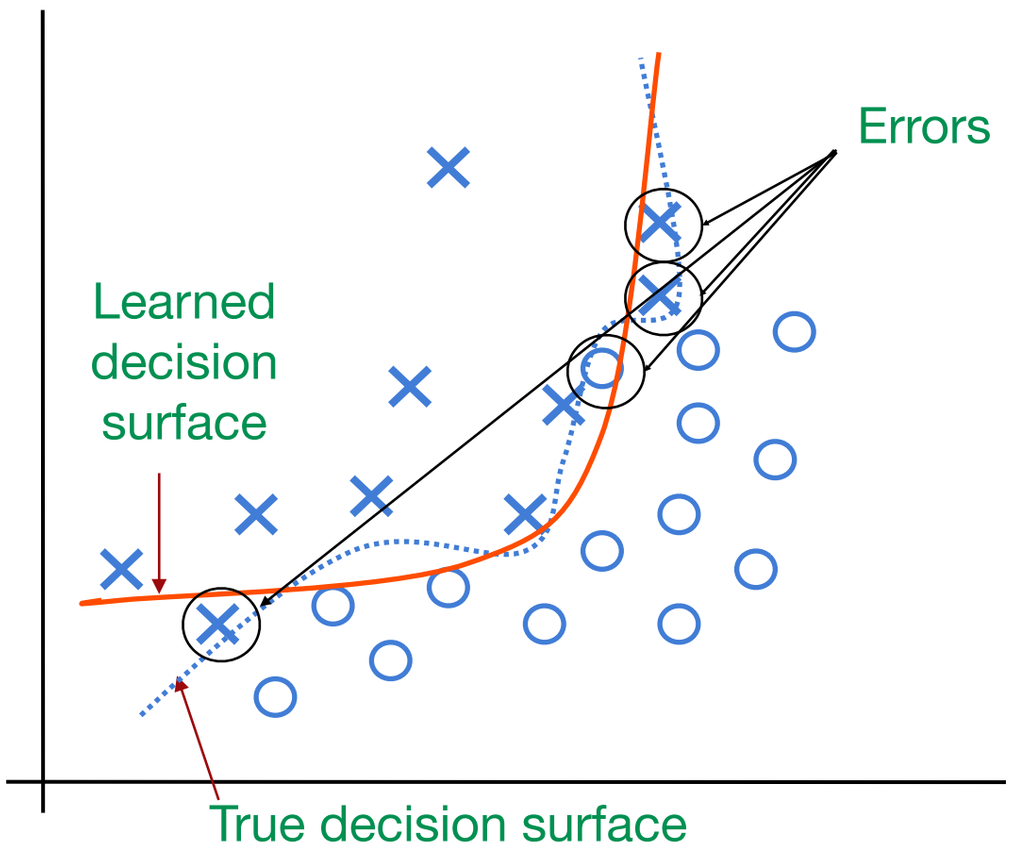
In sklearn $m \leftarrow \sqrt{|D|}$ i.e. very few features considered

© UCD Computer Science

Boosting

- **Key idea:** Train a sequence of classifiers, one after another, so that later classifiers are trained to better predict class labels for examples that earlier ones performed poorly on.
- Train and test a classifier to identify the examples where errors were made. For the next classifier, focus on correctly classifying those "difficult" examples.
- Different versions of boosting have been proposed:

  e.g. AdaBoost, Gradient Boosting



Boosting

- **How do we emphasise certain training examples?**
- Assign numeric weights to all examples in the training set, such that more difficult examples have higher weights.
- When generating each classifier, randomly sample the data, where the probability of selecting an example is proportional to its weight.

Example	A	B	C	D	E
Weight	0.2	0.2	0.2	0.2	0.2

Equal weights, so all examples equally likely to be selected.

Example	A	B	C	D	E
Weight	0.3	0.2	0.2	0.2	0.1

More likely to be selected

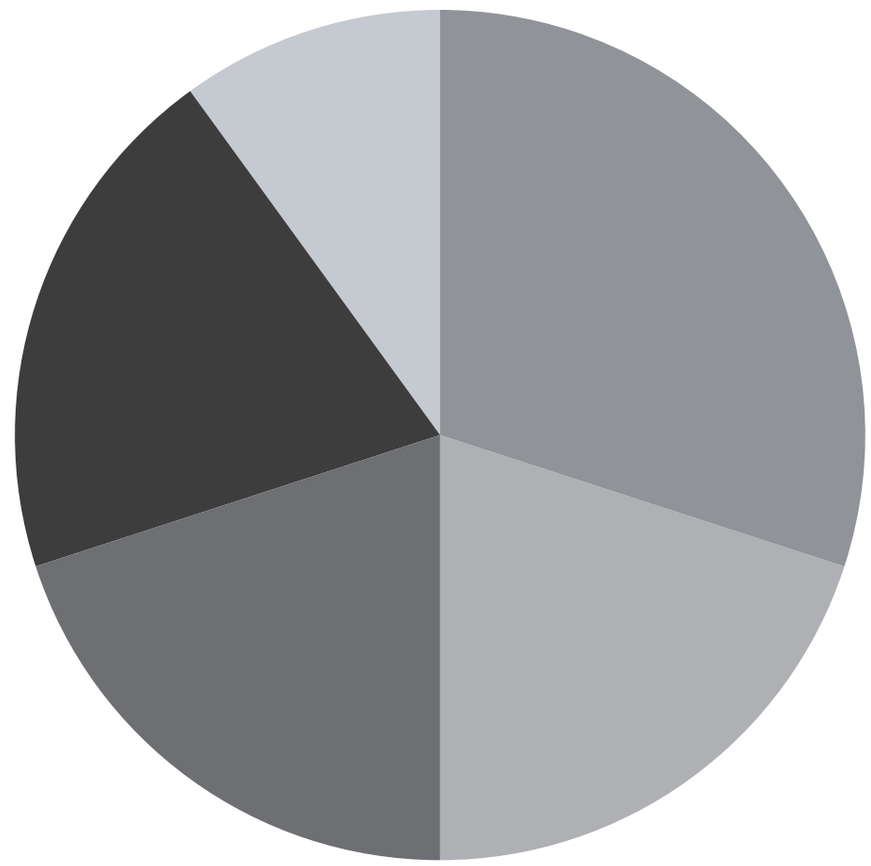
Less likely to be selected

Emphasising difficult training samples

- "Wheel of fortune" metaphor…

Example	A	B	C	D	E
Weight	0.2	0.2	0.2	0.2	0.2

Example	A	B	C	D	E
Weight	0.3	0.2	0.2	0.2	0.1



Boosting

- **Inputs:** Full training set of N examples; maximum ensemble size T; sample size $L \leq N$.
- **Basic approach to boosting:**

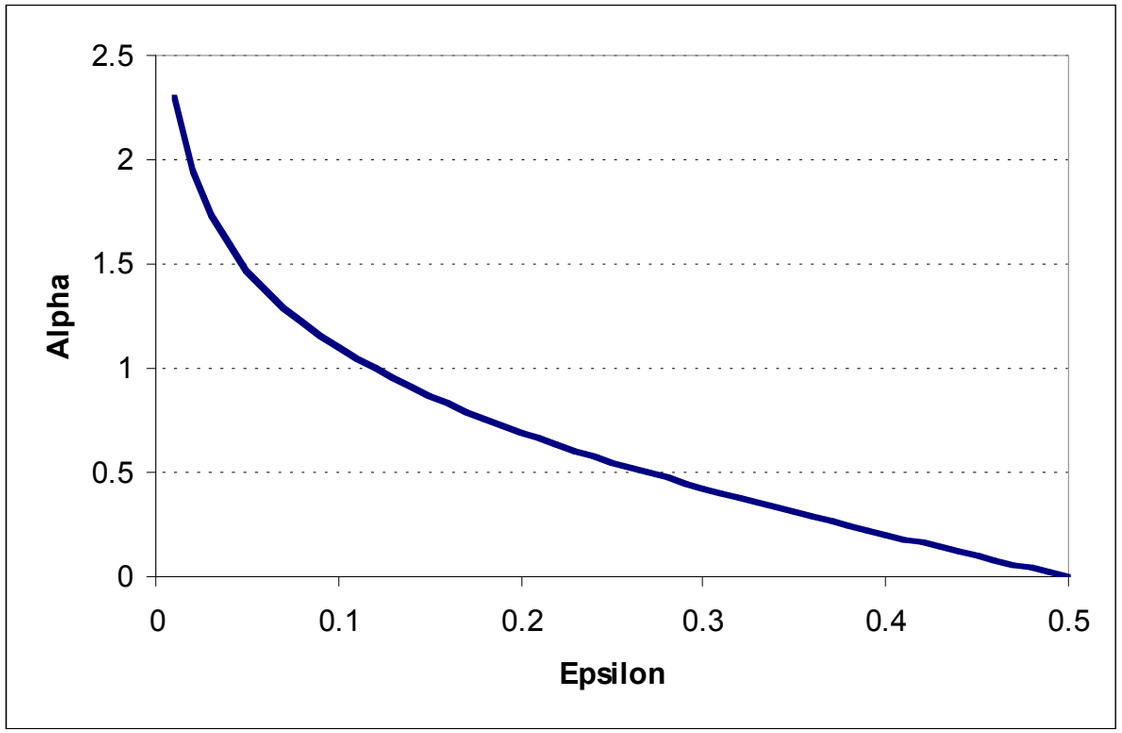
1. Assign an equal weight of $1/N$ to all training examples.
2. FOR $t = 1$ *to* T
   (a) Randomly sample L examples from the full training set with replacement, based on the current weights.
   (b) Train a classifier on this sample.
   (c) Identify examples misclassified by this classifier.
   (d) Increase weights for misclassified examples, decrease weights for correctly classified examples.
3. Output final model based on all classifiers (classifiers weighted according to their error rate).

© UCD Computer Science

Boosting: weight update

- ϵ_t the error for Classifier t
- α_t the weight for Classifier t

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$



- Up-weighting & down-weighting

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if} \quad h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if} \quad h_t(x_i) \neq y_i \end{cases}$$

- Example ϵ_t = 0.2
  - $e^{-\alpha_t}$ = 0.5 (lower)
  - e^{α_t} = 2 (raise)

Example: Bagging v Boosting

- **Problem:** Apply a classifier to a training set with 8 examples *{A, B, C, D, E, F, G, H}*, where example *A* is an outlier and difficult to classify.

- Selected training sets for 4 runs of <u>bagging</u> - i.e. simple random sampling with replacement.

➡ All examples equally weighted.

Original	A	B	C	D	E	F	G	H
Set 1	B	G	H	C	G	F	C	A
Set 2	G	H	E	F	D	B	G	A
Set 3	C	F	B	G	E	F	B	B
Set 4	D	E	A	D	E	D	C	H

- Selected training sets for 4 runs of <u>boosting</u> - i.e. increase weights for misclassified examples.

➡ The “hard” example *A* appears more frequently in later sets.

Original	A	B	C	D	E	F	G	H
Set 1	B	G	H	C	G	F	C	A
Set 2	A	D	E	D	A	E	F	D
Set 3	G	A	E	H	A	H	A	D
Set 4	A	A	F	A	A	C	A	E

D. Opitz & R. Maclin. "Popular Ensemble Methods: An Empirical Study" (1999)

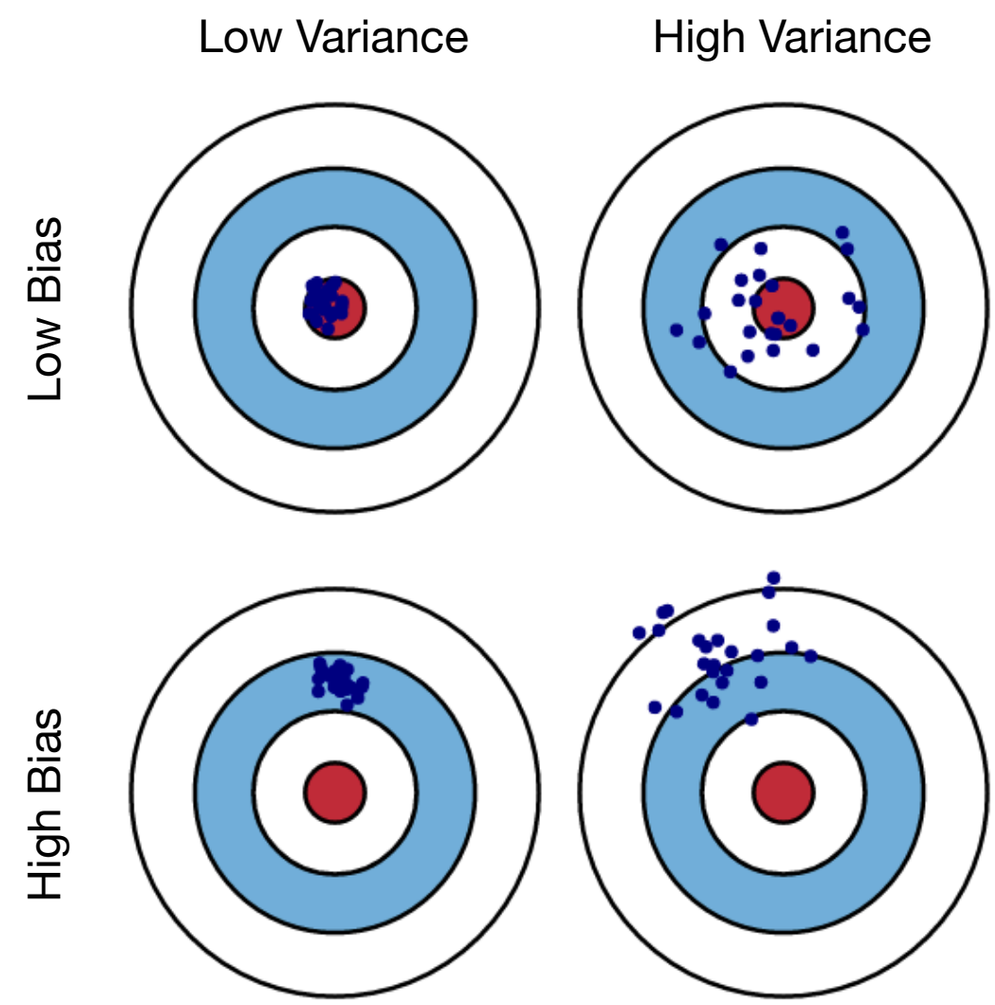
Boosting

```
from sklearn.ensemble import AdaBoostClassifier

adaBoost = AdaBoostClassifier(n_estimators=100, algorithm = 'SAMME')
scores_adaBoost = cross_val_score(adaBoost, X, y, cv=folds, verbose = v, n_jobs = -1)

scores_adaBoost.mean()
Out[45]:
0.7284920742446823
In [46]:
# Train Adaboost on all data and check weights for first 10 models
ab = adaBoost.fit(X,y)
ab.estimator_weights_[:10]
Out[46]:
array([0.85514873, 0.43004882, 0.48229211, 0.4657541 , 0.12063232,
       0.35198136, 0.23122383, 0.13955368, 0.13043829, 0.12244209])
```

Weights for successive ensemble members

Bias and Variance

- We can view the error of a classifier on a given dataset as consisting of three parts:
  1. Bias: systematic error in a particular direction.
  2. Variance: variability in model prediction, sensitivity to small changes in the training set.
  3. Irreducible error (Bayes error) cannot do better than this

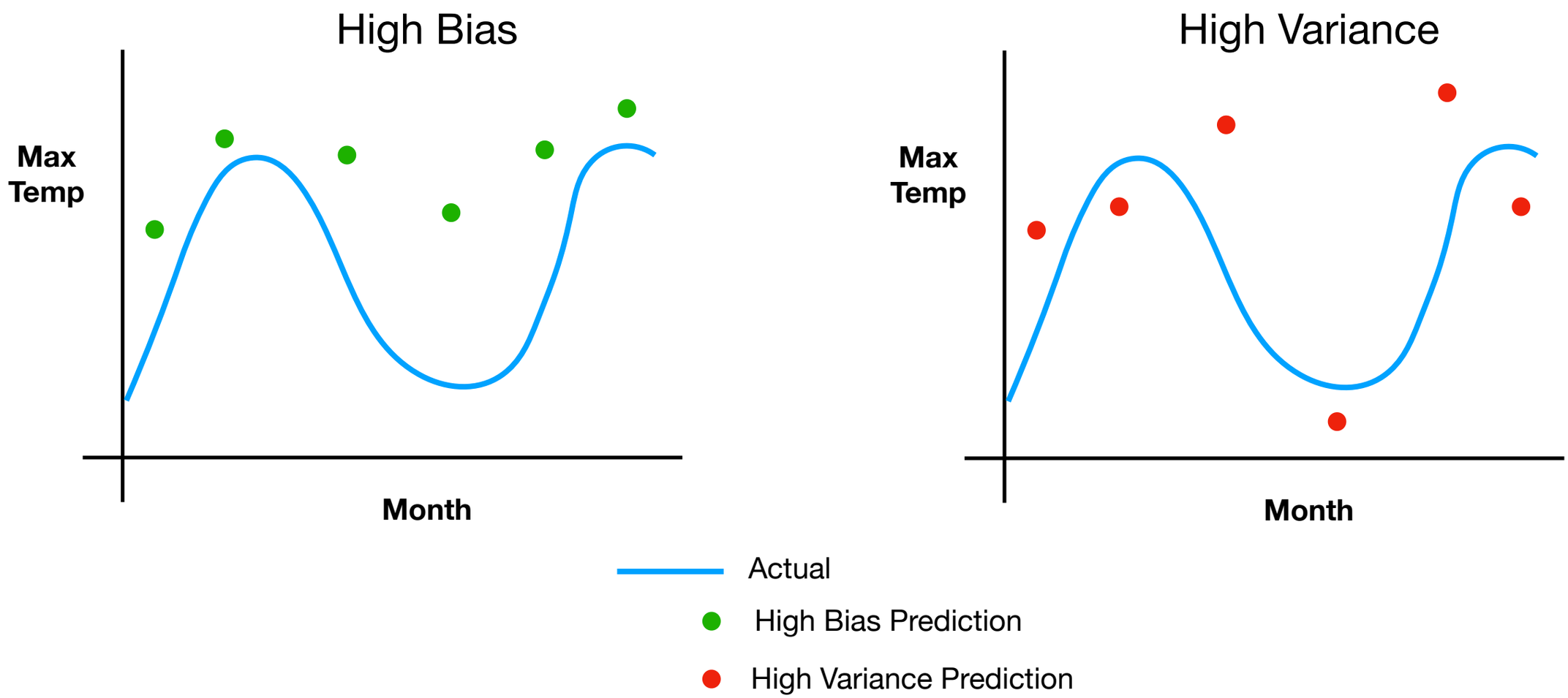


- In machine learning there is often a tradeoff between minimising the bias and the variance sources of error.
  - Low bias but high variance classifiers tend to <u>overfit</u>.
  - High bias but low variance classifiers tend to <u>underfit</u>.

© UCD Computer Science

Bias & Variance: Regression

- Two components of error
- Two ways to be wrong



Bias & Variance: Classification

- Two components of error
- Two ways to be wrong

Bias

Actual	Predicted
✓	✓
✓	✓
✓	✓
✓	✓
✓	✓
✓	✓
✗	✗
✗	✓
✗	✓
✗	✗
✗	✗

Biased toward majority class

Variance

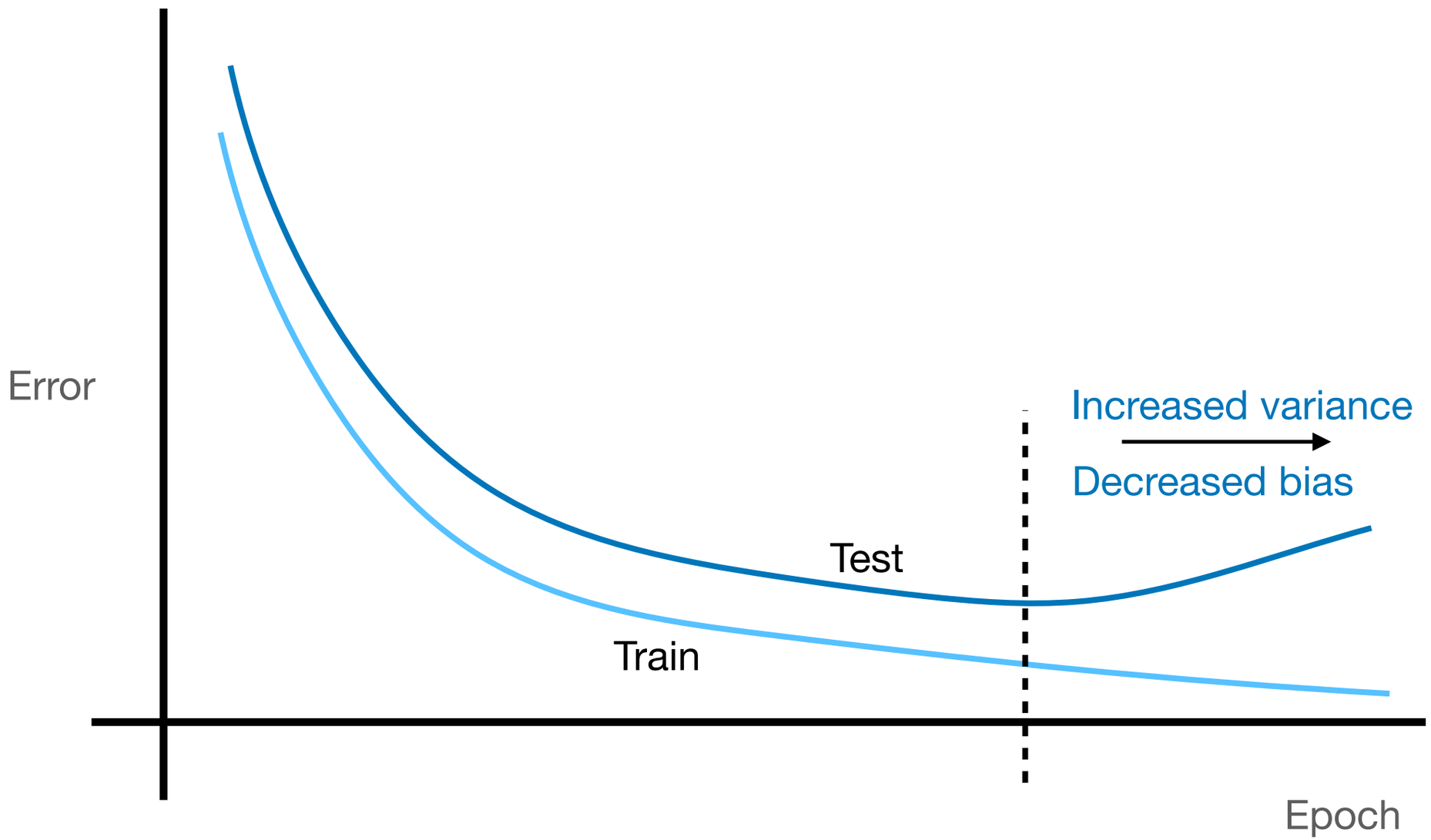
Actual	Predicted
✓	✓
✓	✓
✓	✓
✓	✗
✓	✓
✓	✓
✗	✗
✗	✓
✗	✗
✗	✗
✗	✗

Errors in both directions

© UCD Computer Science

Bias/Variance and Model Fitting

- Early Stopping



Bias and Variance

- Underfitting occurs when a statistical model or machine learning algorithm is unable to capture the underlying trend of the data - i.e. the model or the algorithm does not fit the data well enough.

In machine learning there is often a tradeoff between minimising the bias and the variance sources of error.

How do this apply to different ensemble generation methods?

➡ Bagging can often reduce the variance part of error.

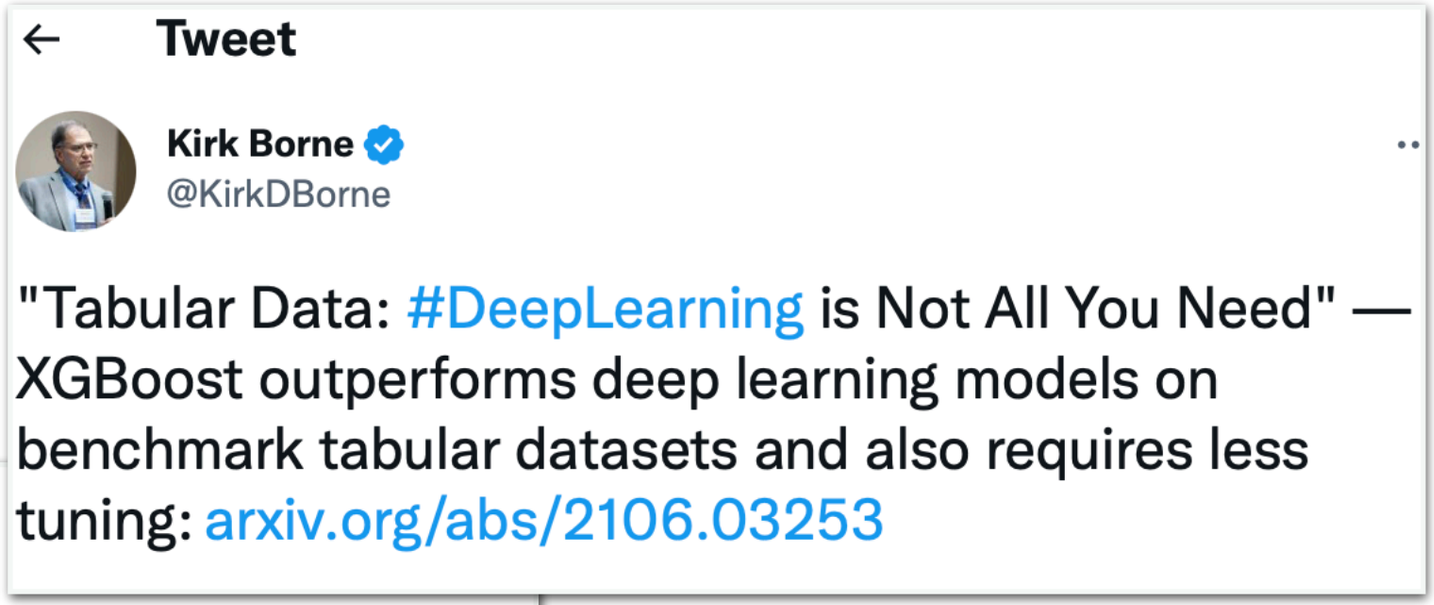
➡ Boosting can often reduce variance AND bias, since it focuses on misclassified examples.

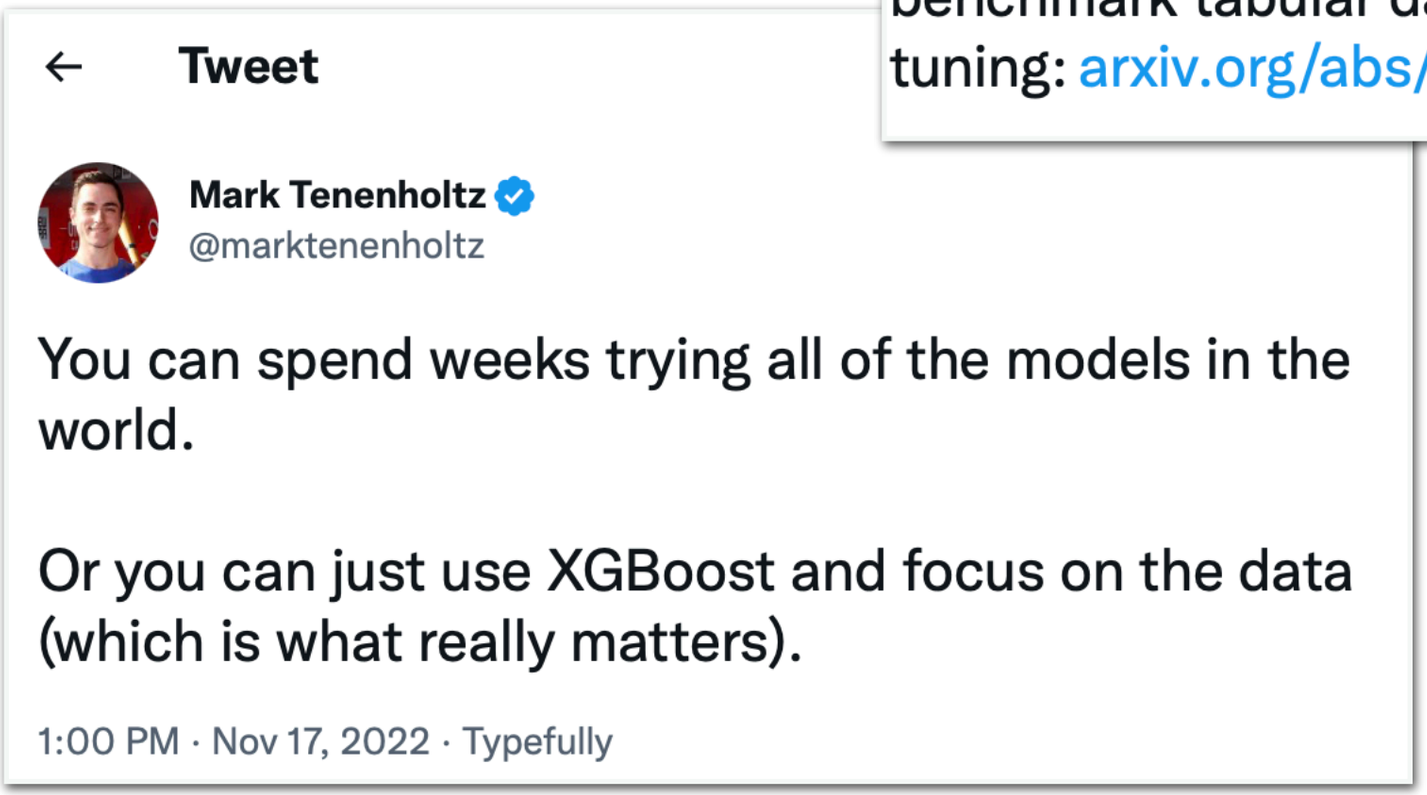
➡ Boosting may sometimes increase error, as it is susceptible to noise, which can lead to overfitting.

© UCD Computer Science

XGBoost: state of the art

- One algorithm to rule them all…

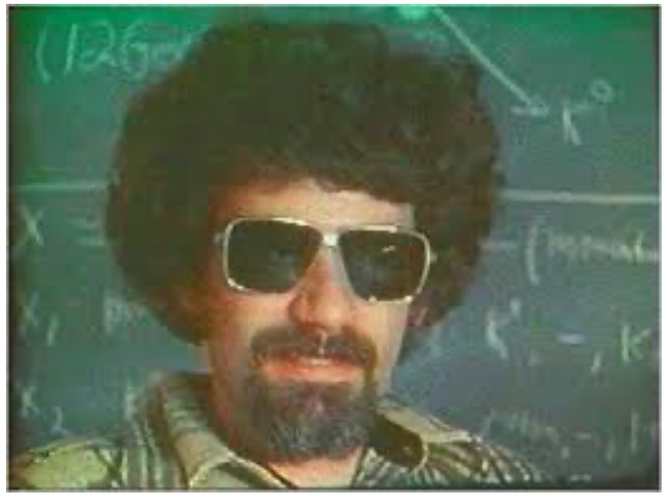




Gradient Boosting — the state of the art

- Gradient Boosting: variation on Boosting that does very well in ML competitions
  - (e.g. Kaggle)
- Sequential building of trees
  - Boosting: focus in on problematic data points
  - Gradient Boosting: use Gradient Descent to minimise error on next tree
- Python
  - sklearn: GradientBoostingClassifier (Regressor)
  - XGBoost
  - LightGBM Microsoft
  - CatBoost Яндекс Yandex

Category Features



Jerome Friedman

Gradient Boosting - choosing the next model

- Boosting: focus in on problematic data points
- Gradient Boosting: use Gradient Descent to minimise error on next tree

$$\hat{y}_i^m = F_m(x_i)$$

estimate from model (ensemble) *m* for sample *i*

$$\epsilon_i^m = y_i - \hat{y}_i^m = y_i - F_m(x_i)$$

error (residual) from model *m* for sample *i*

Next model *learns error*

$$F_{m+1}(x_i) = F_m(x_i) + h_m(x_i) = y_i$$

m+1 model is *m* model + estimate of error of *m* model

estimate of error of *m* model

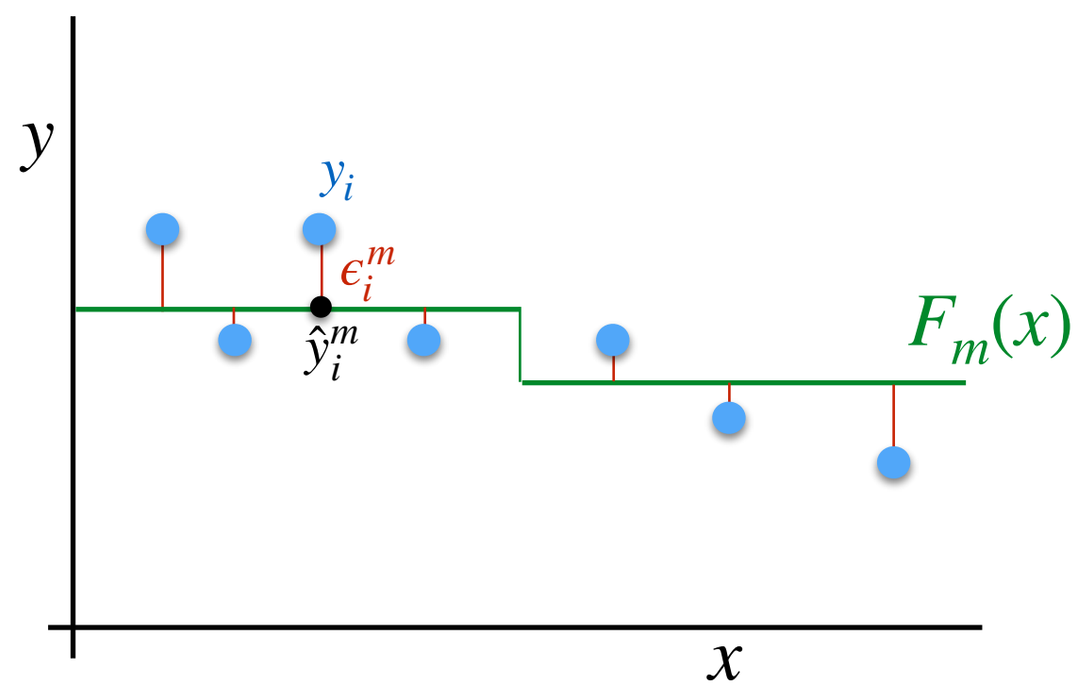
$$h_m(x_i) = y_i - F_m(x_i)$$

loss function (regression)

$$C_{RMS} = \frac{1}{n}\sum_{i=1}^{n}(y_i - F(x_i))^2$$

$$-\frac{\delta C_{RMS}}{\delta F(x_i)} = \frac{2}{n}(y_i - F(x_i)) = \frac{2}{n}h_m(x_i)$$

residuals proportionate to the negative gradient of the loss function



References

- D. Opitz and R. Maclin. "Popular Ensemble Methods: An Empirical Study" (1999). *Journal of Artificial Intelligence Research.*
- Breiman, L., (1996) “Bagging predictors”. *Machine Learning*, 24:123-140.
- Breiman, Leo. "Random forests." *Machine learning* 45.1 (2001): 5-32.
- Krogh, A., Vedelsby, J., (1995) “Neural Network Ensembles, Cross Validation and Active Learning”, in *Advances in Neural Information Processing Systems 7*
- Baur, E., and Kohavi, R. (1999) “An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants”, *Machine Learning*.
- Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In Proceedings of the 22ndACM SIGKDD international conference on knowledge discovery and data mining (pp. 785-794).

© UCD Computer Science