Classifier Combination

Semester 1, 2021

Ling Luo

Recap

- We have discussed
  - individual classification algorithms
  - Performance evaluation and error analysis
- If we were to carry out error analysis of multiple classifiers over a given dataset, would the instances misclassified by better-performing classifiers be a subset of the errors made by worse-performing classifiers? No

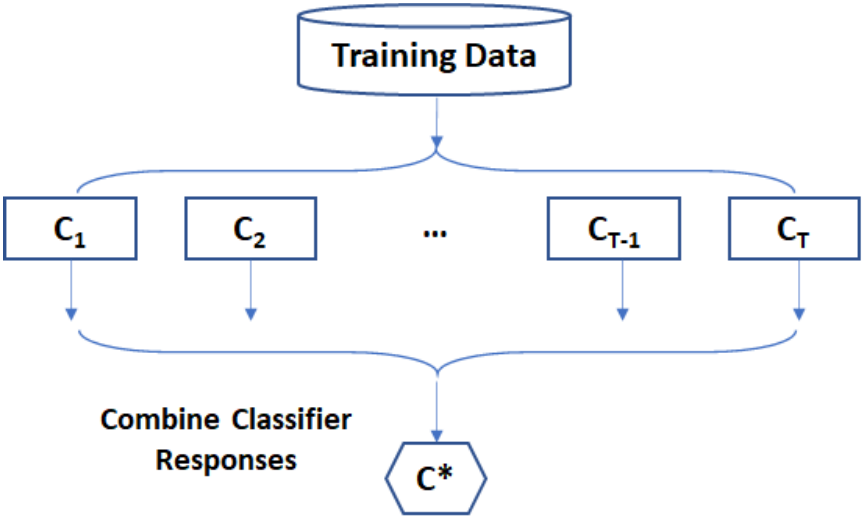
Different classifiers can have strengths & weakness which make different misclassification

Outline

- Classifier Combination
- Bagging
- Random Forests
- Boosting
- Stacking

Classifier Combination

- **Classifier combination/ensemble learning** constructs a set of **base classifiers** from training data and performs classification by **aggregating** the outputs made by each base classifier.



Does Combination Work?

- Intuitions:
  - take into account the opinions of several experts rather than relying only on one
  - the combination of lots of weak classifiers can be at least as good as one strong classifier
  - the combination of a selection of strong classifiers is (usually) at least as good as the best of the base classifiers

- Does combination always have better performance?

Does Combination Work?

- The following tables show the performance of different classifiers (three base classifiers C_1, C_2, C_3 and their combination C^* using majority voting) on three instances t_1, t_2, t_3, where √ is correct and x is incorrect

	t_1	t_2	t_3
C_1	√	√	x
C_2	x	√	√
C_3	√	x	√
C*	√	√	√

C* is better

	t_1	t_2	t_3
C_1	√	√	x
C_2	√	√	x
C_3	√	√	x
C*	√	√	x

C* is the same

	t_1	t_2	t_3
C_1	√	x	x
C_2	x	√	x
C_3	x	x	√
C*	x	x	x

C* is worse

Does Combination Work?

- When does the combination work?
  - the base classifiers do not make the same mistakes
  - each base classifier is reasonably accurate

	t_1	t_2	t_3
C_1	√	√	x
C_2	x	√	√
C_3	√	x	√
C^*	√	√	√

C* is better

	t_1	t_2	t_3
C_1	√	√	x
C_2	√	√	x
C_3	√	√	x
C^*	√	√	x

C* is the same

	t_1	t_2	t_3
C_1	√	x	x
C_2	x	√	x
C_3	x	x	√
C^*	x	x	x

C* is worse

Construct Base Classifiers

bagging
boosting
random forest

- **Instance manipulation**: generate multiple training datasets through sampling, and train a base classifier over each (e.g. bagging)
- **Feature manipulation**: generate multiple training datasets through different feature subsets, and train a base classifier over each (e.g. random forest)
- **Algorithm manipulation**: semi-randomly tweak internal parameters within a given algorithm to generate multiple base classifiers over a given dataset

eg. DT
instead of selecting the best attribute
select one of top k attributes
introduce randomness

Classify with Combined Classifiers

- The simplest means of classification over multiple base classifiers is voting:
  - for nominal classes, run multiple base classifiers over the test data and select the class predicted by the most base classifiers (e.g. KNN)
  - for continuous output, average over the numeric predictions of our base classifiers

Bias and Variance

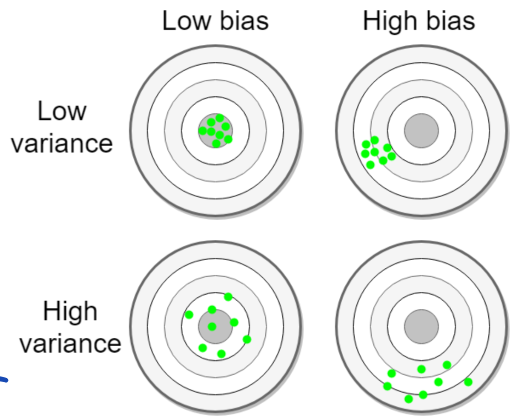
- Analysing the generalisation error of a predictive model
- From model perspective:

Bias: the tendency of our classifier to make systematically wrong predictions.

regression : always make higher prediction
classification: predict A as B

same learner
different
training

Variance: the tendency of producing different models or predictions for different training sets using same learner.



- Lower bias and lower variance → better generalisation

Image source: www.machinelearningtutorial.net/2017/01/26/the-bias-variance-tradeoff/

Classifier Combination

- Bagging
- Random Forests
- Boosting
- Stacking

Bagging

- Bagging = bootstrap aggregating
- Intuition: the more data, the better performance (lower the variance), so how can we get more data out of a fixed training dataset?

  more general for seeing the dataset
- Method: construct new datasets through a combination of **random sampling and replacement**

Bagging: Sampling Examples

- Randomly sample the original dataset N times, with replacement if no replacement, will always get the same data set
- We get a new dataset of the same size, where any individual instance is absent with probability $(1 - \frac{1}{N})^N$ prob of not being selected in one sample
- Construct k random datasets for k base classifiers, and arrive at prediction via voting

1	2	3	4	5	6	7	8	9	10

original training data

7	2	6	7	5	4	8	8	1	10
1	3	8	10	3	5	10	8	2	9
2	9	4	2	7	9	3	10	5	4

bootstrap samples

Bagging: Classification

- The same base classification algorithm is used throughout
- Reduces the variance of predictions
- Effective for unstable classifiers
  - unstable: small changes in the training set result in large changes in predictions, e.g. DTs
  - may slightly decay the performance of stable classifiers, e.g. kNN

when $N \to \infty$
absent rate will converge to 37%
new set cover 63% data

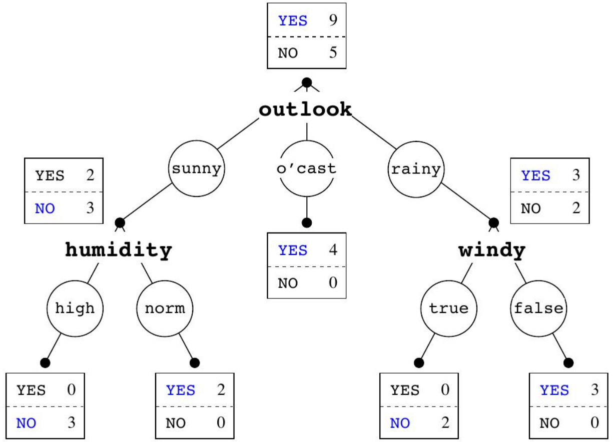
less training instance
?
with preset k for KNN, it is stable for small changes.
use bagging may not help. ?

Bagging: Classification

- Simple method based on sampling (instance manipulation) and voting
- Possibility to parallelise computation of individual base classifiers data sampled independently → base classifiers also indepently
- Effective over noisy datasets, as the outliers may vanish base model without outliers beat base model with outlier
- Performance is generally significantly better than the base classifiers and only occasionally substantially worse

Random Tree feature manipulation

- Random Tree: a Decision Tree, but only some of the possible attributes are considered at each node
  - e.g., a fixed proportion τ of all the attributes
  - pro faster to build than a deterministic Decision Tree, but increases model variance con



Decision Tree

Random Forests

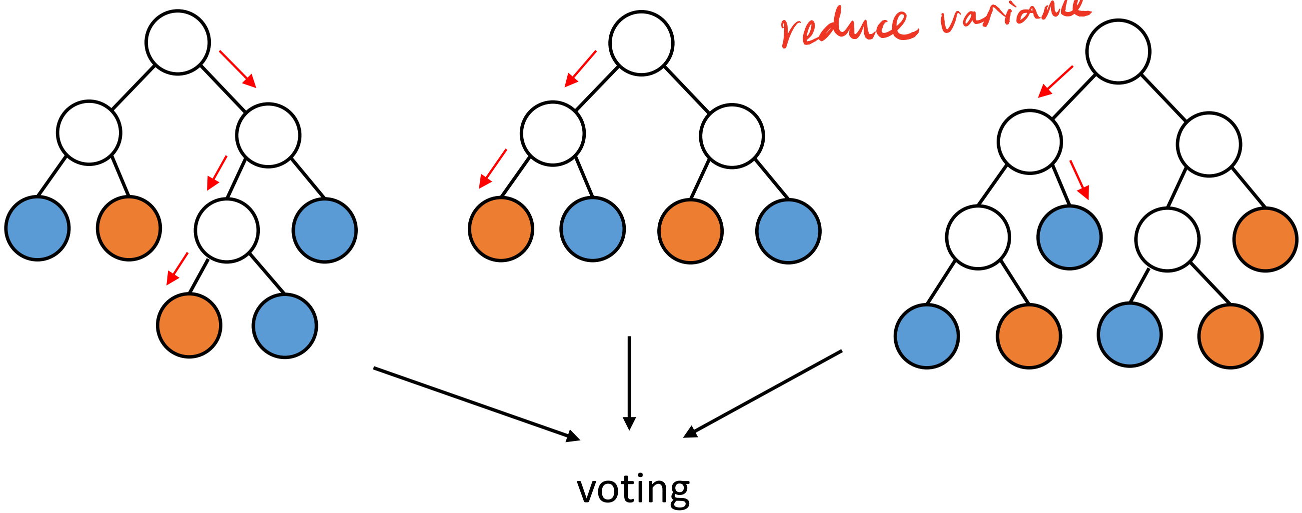
① instance manipulation ⇒ train random trees. with different bagged set

② feature manipulation ⇒ at different nodes.

bootstrape (training set) + random trees

- An ensemble of Random Trees, many trees = forest
  - Each tree is built using a different Bagged training dataset
  - The combined classification is via voting

reduce variance (not the bias)



Random Forests

out-of-bag sample: samples haven't been included in the training for a classifier

- Hyperparameters:
  - number of trees *B*, which can be tuned based on "*out-of-bag*" error
  - feature sub-sample size: as it increases, both the strength and the correlation increase ($\lfloor \log_2 |F| + 1 \rfloor$)

    correlation between tree
- Interpretation:
  - logic behind predictions on individual instances can be followed through the various trees

    hard to interpret

Random Forests

- Practical properties:
  - Generally a very strong performer, efficient to construct
  - Parallelisable
  - Robust to overfitting
  - Interpretability sacrificed

Boosting

instance manipulation

- Intuition: tune base classifiers to focus on the hard-to-classify instances
- Method: iteratively change the distribution and weights of training instances to reflect the performance of the classifier on the previous iteration
  - start with sampling: each training instance having $\frac{1}{N}$ probability of being included in the sample
  - over T iterations, train a classifier and update the weight of each instance according to whether it is correctly classified
  - combine the base classifiers via weighted voting

Boosting: Sampling Examples

- Sampling examples with replacement

original training data

1	2	3	4	5	6	7	8	9	10

Iteration 1

error

7	2	6	7	5	4	8	8	1	10

in regression
put more instances
→ the error for that instance
is more important

Iteration 2

1	3	8	4	3	5	10	4	2	9

boosting samples

Iteration 3

4	9	4	2	7	9	3	10	5	4

Boosting Example: AdaBoost

- Base classifiers: $C_1, C_2, \ldots, C_i, \ldots, C_T$
- Training instances $\{(x_j, y_j) | j = 1,2, \ldots, N\}$
- Initial instance weights $\left\{w_j^{(1)} = \frac{1}{N} \middle| j = 1,2, \ldots, N\right\}$

- **Construct classifier $\boldsymbol{C_i}$ in iteration $\boldsymbol{i}$**
  - Compute the error rate for C_i

$$\varepsilon_i = \sum_{j=1}^{N} w_j^{(i)} \delta(C_i(x_j) \neq y_j)$$

where $\delta(\cdot)$ is an indicator function, which is 1 if the condition is true.

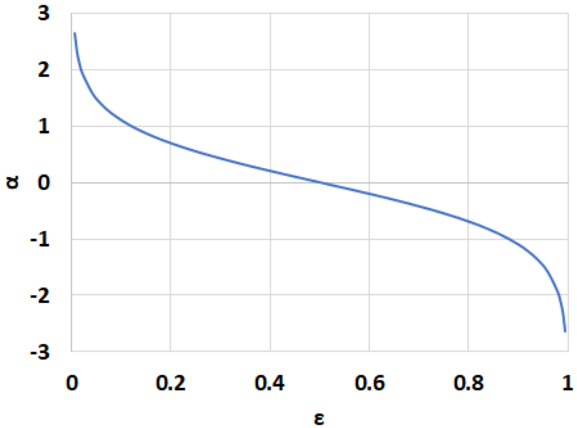
Boosting Example: AdaBoost

- Importance of C_i: α_i the weight associated with the classifiers' votes

$$\alpha_i = \frac{1}{2} ln \frac{1-\varepsilon_i}{\varepsilon_i}$$

weight for classifier

low error ↓ high weight



- Update instance weight (prepare for iteration $i+1$):

weight for instance

$$w_j^{(i+1)} = \frac{w_j^{(i)}}{Z^{(i)}} \times \begin{cases} e^{-\alpha_i} & \text{if } C_i(x_j)=y_j \\ e^{\alpha_i} & \text{if } C_i(x_j)\neq y_j \end{cases}$$

$e^{-\alpha_i} < 1$ $w_j \downarrow$

$e^{\alpha_i} > 1$ $w_j \uparrow$

where $Z^{(i)}$ is the normalisation term.

Boosting Example: AdaBoost

- Continue iterating for $i = 2, \ldots, T$, but reinitialise the instance weights whenever $\varepsilon_i > 0.5$

- $\boxed{\textbf{Classification}}$: combine base classifiers

weighted voting

$$C^*(x) = \arg\max_y \sum_{i=1}^{T} \alpha_i \, \delta(C_i(x) = y)$$

Boosting

- Base classifiers: decision stumps (OneR) or decision trees
- Mathematically complicated but computationally cheap method based on iterative sampling and weighted voting
- The method has guaranteed performance in the form of error bounds over the training data
- More computationally expensive than bagging
- In practical applications, boosting has the tendency to overfit

Comparison

- Bagging/Random Forest vs. Boosting

Bagging/Random Forest	**Boosting**
Parallel sampling	Iterative sampling
Simple voting	Weighted voting
Homogeneous classifiers	Homogeneous classifiers
Minimise variance	Minimise instance bias
Not prone to overfitting	Prone to overfitting

Stacking

- Intuition: smooth errors over **a range of algorithms** with different biases
- Method 1: voting? Which classifier to trust?
- Method 2: train a meta-classifier (level-1 model) over the outputs of the base classifiers (level-0 model)
  - learn which classifiers are the reliable ones, and combine the output of base classifiers
  - train using nested cross validation to reduce bias

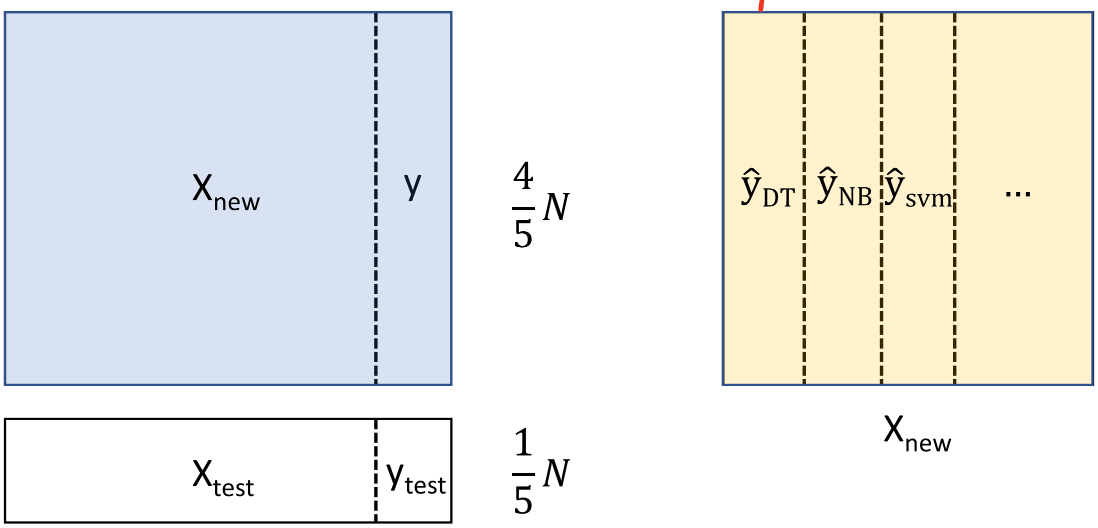
Stacking

- **Level-0:** base classifiers
  - Given training dataset (X, y)
  - Train different classifiers: e.g. SVM, Naïve Bayes, DT
- **Level-1:** combination
  - Construct new attributes based on Level-0 classifiers
    - Each attribute contains the predictions of a level-0 classifier. If there are *M* level-0 classifiers, add *M* attributes.
    - Discard or keep original data *X*
    - Consider other data if available (NB probability scores, weights of SVM)
  - Train meta-classifier (e.g. Logistic Regression) to make final prediction.

Stacking

- Nested cross validation
  - Example: 2 layers of CV

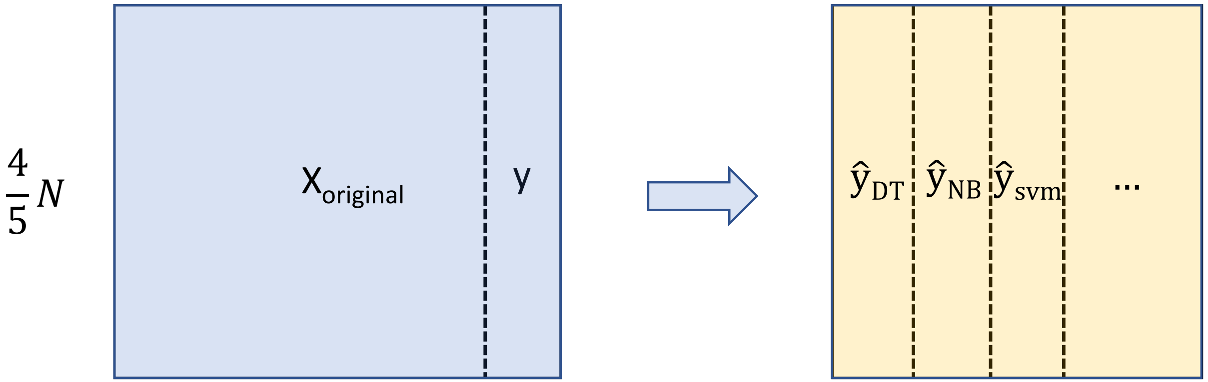
Level 1: for meta-classifier, **outer CV** fold = 5 prediction



Stacking

- Nested cross validation
  - Example: 2 layers of CV

Level 0: for base classifiers, if **no CV** (fold = 1)

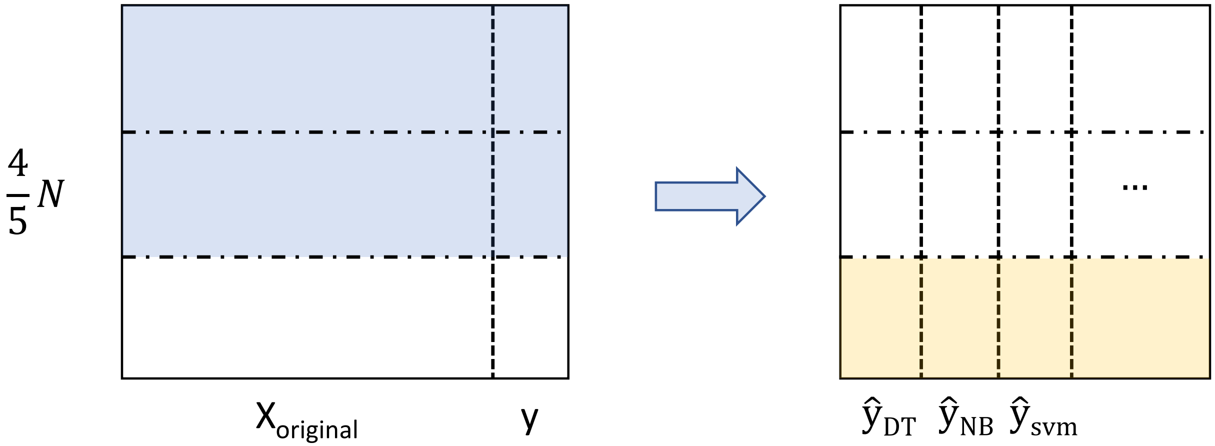


problem: see all traing instances, overfit.

Stacking

- Nested cross validation
  - Example: 2 layers of CV

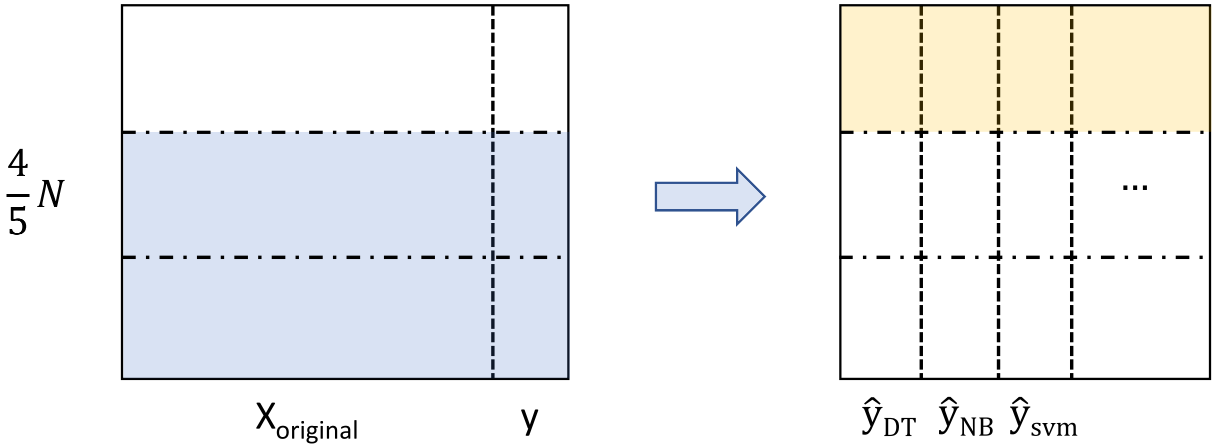
Level 0: for base classifiers, **inner CV** fold = 3



Stacking

- Nested cross validation
  - Example: 2 layers of CV

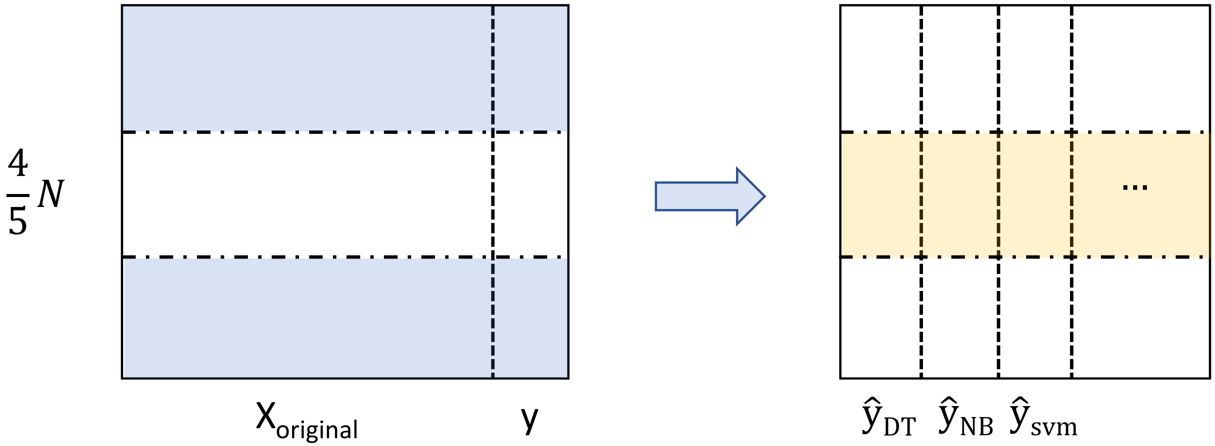
Level 0: for base classifiers, **inner CV** fold = 3



Stacking

- Nested cross validation
  - Example: 2 layers of CV

Level 0: for base classifiers, **inner CV** fold = 3



Stacking

pro

- Able to combine heterogeneous classifiers with varying performance

con

- Mathematically simple but computationally expensive method
- Generally, stacking results in as good or better results than the best of the base classifiers

Summary

- What is classifier combination?
- What is the basic idea behind:
  - Bagging
  - Random Forest
  - Boosting
  - Stacking
- How to compare different models, e.g. bagging vs. boosting?

References

- Leo Breiman. Random Forests. Machine Learning, 45(1):5–32, 2001.
- Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, and Vipin Kumar. Introduction to Data Mining. Pearson, 2018.
- Ian Witten, Eibe Frank, and Mark A. Hall. Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, 3rd edition, 2011.

Calculate total error rate

012.

Question 5 0 / 1 pts

Suppose there are 3 independent binary classifiers C_1, C_2, and C_3, with error rates 0.3, 0.2, and 0.2 respectively. If the classifiers are combined by majority voting, what is the error rate of the combined classifier?

- 0.012
- 0.124
- You Answered: 0.164
- Correct Answer: 0.136

To make an error by the combined classifier, at least two classifiers should make errors. There are four scenarios to generate wrong predictions:

incorrect classifiers	error rate
$\{C_1, C_2\}$	0.3*0.2*(1-0.2) = 0.048
$\{C_2, C_3\}$	0.3*(1-0.2)*0.2 = 0.048
$\{C_1, C_3\}$	(1-0.3)*0.2*0.2 = 0.028
$\{C_1, C_2, C_3\}$	0.3*0.2*0.2 = 0.012

Thus, the error rate of the combined classifier is 0.048+0.048+0.028+0.012=0.136