CS4104 Applied Machine Learning

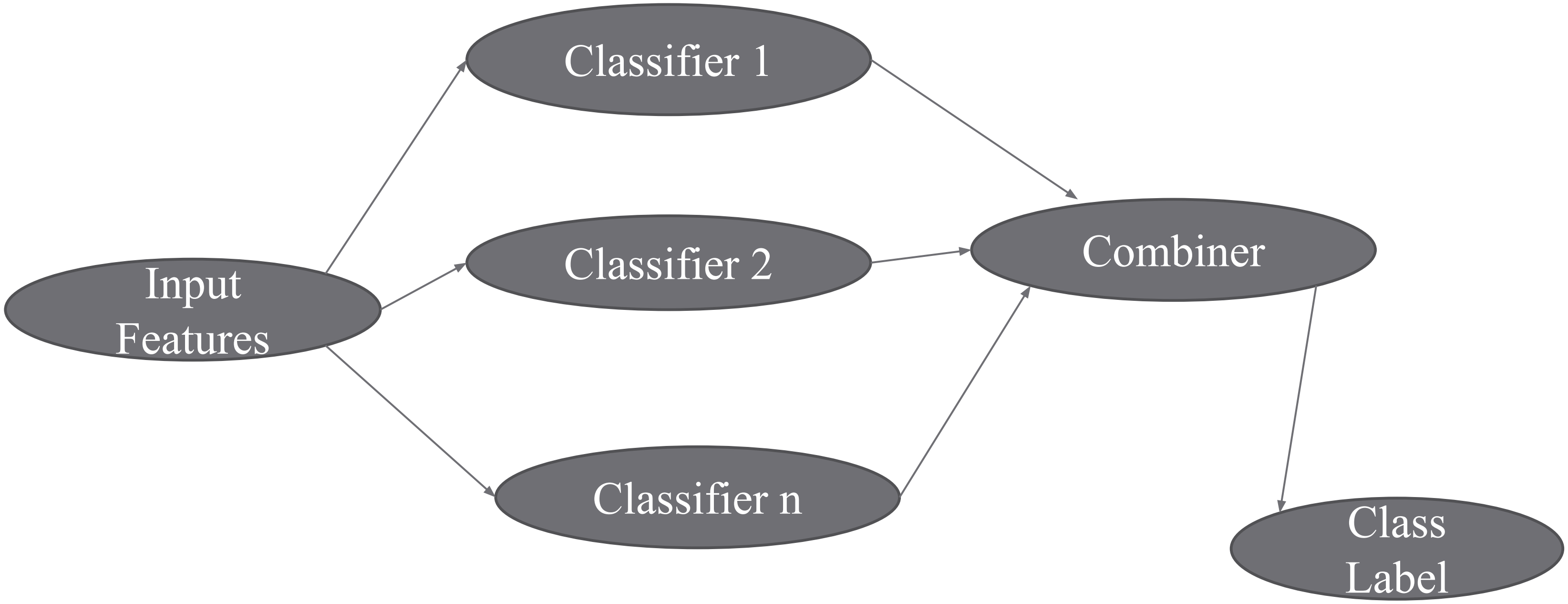
Ensemble Learning

zeshan.khan@nu.edu.pk

Classifiers: Example

- If a dataset produced the accuracy using:
  - KNN: 70%
  - DT: 70%
  - NB: 75%
  - SVM: 76%
- What about combination of the results of all above

Ensemble Learning



Key Ensemble Questions

Which components to combine?

- different learning algorithms
- same learning algorithm trained in different ways
- same learning algorithm trained the same way

How to combine classifications?

- majority vote
- weighted (confidence of classifier) vote
- weighted (confidence in classifier) vote
- learned combiner

What makes a good (accurate) ensemble?

Why Do Ensembles Work?

Hansen and Salamon, 1990

If we can assume classifiers are random in predictions and accuracy > 50%, can push accuracy arbitrarily high by combining more classifiers

Key assumption: classifiers are independent in their predictions

- not a very reasonable assumption
- more realistic: for data points where classifiers predict with > 50% accuracy, can push accuracy arbitrarily high (some data points just too hard)

Why do ensembles work?

Dietterich(2002)

1. *The Statistical Problem*
2. The Computational Problem
3. The Representational Problem

The Statistical Problem

Arises when the hypothesis space is too large for the amount of available data. Hence, there are many hypotheses with the same accuracy on the data and the learning algorithm chooses only one of them! There is a risk that the accuracy of the chosen hypothesis is low on unseen data!

The Computational Problem

- Arises when the learning algorithm cannot guarantees finding the best hypothesis.

The Representational Problem

- Arises when the hypothesis space does not contain any good approximation of the target class(es).

Why do ensembles work?

- The statistical problem and computational problem result in the variance component of the error of the classifiers!
- The representational problem results in the bias component of the error of the classifiers!

What Makes a Good Ensemble?

Krogh and Vedelsby, 1995

Can show that the accuracy of an ensemble is mathematically related:

$$Ee = Ec - D$$

Ee is the error of entire ensemble

Ec is the average error of the component classifier

D is the term measuring the diversity of the component

Effective ensembles have accurate and diverse components

Classification Fusion Techniques

- Homogenous Classifiers (Same Classifiers but different training data) e.g. Bagging, Boosting etc
- Heterogeneous Classifiers (Different Classifiers but same training data) e.g. Majority Voting, Mean etc)
- Combination of Homogenous and Heterogeneous Classifiers
  - Homogenous then heterogeneous on output
  - Heterogeneous then homogenous on output

Classification Fusion Techniques: Heterogeneous

Advantage

- Each classifier can concentrate on its own small subproblem instead of trying to cope with the classification problem as a whole, which may be too hard for a single classifier

Disadvantage

- can be lack of diversity among some classifiers

Type I (abstract level): Classifiers Outputs

- This is the lowest level since a classifier provides the least amount of information

- on this level, Classifier output is merely a single class label or an unordered set of candidate classes

Type II (rank level)

- Classifier output on the rank level is an ordered sequence of candidate classes, the so-called n-best list
- The candidate class at the first position is the most likely class, while the class positioned at the end of the list is the most unlikely
- Note that there are no confidence values attached to the class labels on rank level
- Only their position in the n-best list indicates their relative likelihood

Type III (measurement level)

- In addition to the ordered n-best lists of candidate classes on the rank level, classifier output on the measurement level has confidence values assigned to each entry of the n-best list
- These confidences, or scores, can be arbitrary real numbers, depending on the classification architecture used
- The measurement level contains therefore the most information among all three output levels

Voting Techniques

- Majority Voting
- Average of Probabilities
- Product of Probabilities
- Minimum Probability
- Maximum Probability
- Median

Voting in Ensemble Learning

hard voting

- Every individual classifier votes for a class, and the majority wins. In statistical terms, the predicted target label of the ensemble is the mode of the distribution of individually predicted labels

- $Y = \max_{c \in Classifiers} Y_c$

soft voting

- Every individual classifier provides a probability value that a specific data point belongs to a particular target class. The predictions are weighted by the classifier's importance and summed up. Then the target label with the greatest sum of weighted probabilities wins the vote.

- $Y = \sum_{c \in Classifiers} P_c$

Voting in Ensemble Learning

Hard Voting

- Let Assumes that Three classifiers predicts as follow:
  - Classifier 1 predicts class A
  - Classifier 2 predicts class B
  - Classifier 3 predicts class B
- 2/3 classifiers predict class B, so class B is the ensemble decision.

Soft Voting

- Let Assumes that Three classifiers predicts as follow:
  - Classifier 1 predicts class A with Prob 93%
  - Classifier 2 predicts class A with Prob 44%
  - Classifier 3 predicts class A with Prob 40%
- On average the ensemble produces (93+44+40)/3 = 59% probability predict class A, so class A is the ensemble decision.

zeshan.khan@nu.edu.pk

Voting Techniques

C1	C2	C3	Majority Voting
1	1	1	1
1	1	0	1
0	0	0	0
0	0	1	0

Voting Techniques

C1	C2	C3	Average	Product	Minimum	Maximum	Median
0.9	0.5	0.5	0.63	0.23	0.5	0.9	0.5
0.5	0.5	0	0.33	0.00	0	0.5	0.5
0.1	0.1	0.1	0.10	0.00	0.1	0.1	0.1
0.4	0.4	0.6	0.47	0.10	0.4	0.6	0.4

Average = (c1+c2+c3)/3
Product = c1*c2*c3
Minimum = min(c1,c2,c3)
Maximum = max(c1,c2,c3)
Median = median(c1,c2,c3)

Voting Techniques (Exercise)

C1	C2	C3	Average	Product	Minimum	Maximum	Majority
0.1	0.2	0.9					
0.9	0.7	0.5					
0.4	0.5	0.6					
0.2	0.2	0.8					

Voting Techniques (Exercise)

C1	C2	C3	Average	Product	Minimum	Maximum	Majority
0.1	0.2	0.9	0.4	0.018	0.1	0.9	0
0.9	0.7	0.5	0.7	0.315	0.5	0.9	1
0.4	0.5	0.6	0.5	0.12	0.4	0.6	1
0.2	0.2	0.8	0.4	0.032	0.2	0.8	0

Voting Techniques

C1	C2	C3	C4	C5	Average	Majority
.99	.99	.49	.49	.49	0.69	0
1	1	0	0	0	0.4	0

Voting Techniques (Exercise)

C1	C2	C3	Average	Product	Minimum	Maximum	Majority

From Previous Table, Assume higher value of probability indicates belongs to Class 1

Any value < 0.5, belongs to 0 and >=0.5, belongs to 1
Different voting techniques give different predictions

Voting Techniques (Exercise)

C1	C2	C3	Average	Product	Minimum	Maximum	Majority
0	0	1	0	0	0	1	0
1	1	1	1	0	1	1	1
0	1	1	1	0	0	1	1
0	0	1	0	0	0	1	0

From Previous Table, Assume higher value of probability indicates belongs to Class 1

Any value < 0.5, belongs to 0 and >=0.5, belongs to 1
Different voting techniques give different predictions

Code

- import numpy as np
- from sklearn.linear_model import LogisticRegression
- from sklearn.naive_bayes import GaussianNB
- from sklearn.ensemble import RandomForestClassifier, VotingClassifier
- X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])
- y = np.array([1, 1, 1, 2, 2, 2])
- clf1 = LogisticRegression()
- clf2 = RandomForestClassifier(n_estimators=50)
- clf3 = GaussianNB()
- eclf1 = VotingClassifier(estimators=[('lr', clf1), ('rf', clf2), ('gnb', clf3)], voting='hard')
- eclf1 = eclf1.fit(X, y)
- print(eclf1.predict(X))

Others Heterogeneous Classifiers

- Weighted Majority Vote
- Naïve Bayes Combination
- Fuzzy Integral
- Dempster-Shafer Combination (Probability Combination)
- Many More

Homogenous Ensemble Classifiers

- Same classifier but different training data
  - Bagging
  - Boosting
  - Random Forest
  - Others

Bagging

- Employs simplest way of combining predictions that belong to the same type.
- Combining can be realized with voting or averaging
- Each model receives equal weight
- "Idealized" version of bagging:
  - Sample several training sets of size *n* (instead of just having one training set of size *m where m>>n*)
  - Build a classifier for each training set
  - Combine the classifier's predictions
- This improves performance in almost all cases if learning scheme is *unstable* (i.e. decision trees)

Bagging classifiers

Classifier generation

Let *n* be the size of the training set.

For each of *t* iterations:

- Sample *m* instances with replacement from the training set.
- Apply the learning algorithm to the sample.
- Store the resulting classifier

Classification

For each of the *t* classifiers:

- Predict class of instance using classifier.
- Return class that was predicted most often.

Why does bagging work?

- Bagging reduces variance by voting/ averaging, thus reducing the overall expected error
  - In the case of classification there are pathological situations where the overall error might increase
  - Usually, the more classifiers the better

Random Forest

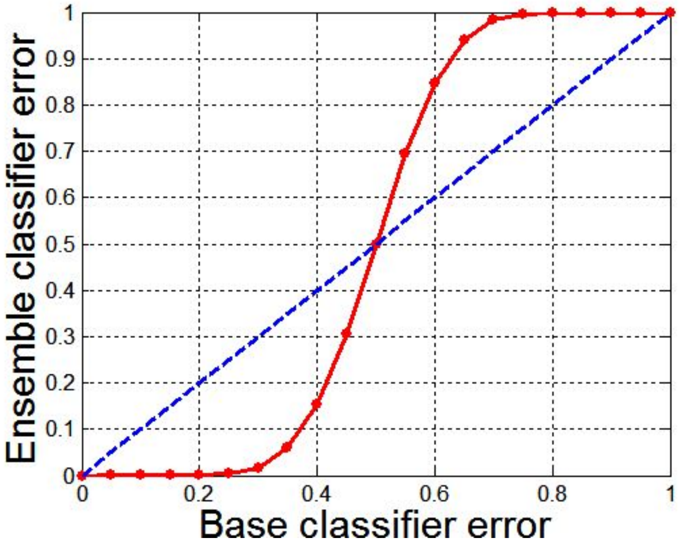
```
$Random\ Forest\ (S, F, B)$
    $H \leftarrow \emptyset$
    $for\ i \in (1, B):$
        $S_i \leftarrow sample(S)$
        $h_i \leftarrow RTree(S_i, F)$
        $H = H \cup h_i$
    $return\ H$
```

```
$RTree(S, F):$
    $for\ n \in Node:$
        $f \leftarrow subset(F):$
        $split\ on\ f_{best}$
    $return\ Tree$
```

- S is training Data
- F is the features set
- B is the number of trees

Bagging and Random Forest

- Bagging usually improves decision trees.
- Random forest usually outperforms bagging due to the fact that errors of the decision trees in the forest are less correlated.



Randomization Injection

- Inject some randomization into a standard learning algorithm (usually easy):
  - –Neural network: random initial weights
  - –Decision tree: when splitting, choose one of the top *N* attributes at random (uniformly)
- Dietterich (2000) showed that 200 randomized trees are statistically significantly better than C4.5 for over 33 datasets!

Feature-Selection Ensembles

- ***Key idea:*** Provide a different subset of the input features in each call of the learning algorithm.
- ***Example:*** Venus&Cherkauer (1996) trained an ensemble with 32 neural networks. The 32 networks were based on 8 different subsets of 119 available features and 4 different algorithms. The ensemble was significantly better than any of the neural networks!

Boosting

- Also uses voting/averaging but models are weighted according to their performance

- Iterative procedure: new models are influenced by performance of previously built ones
  - New model is encouraged to become expert for instances classified incorrectly by earlier models
  - Intuitive justification: models should be experts that complement each other

- There are several variants of this algorithm

AdaBoost.M1

classifier generation

```
Assign equal weight to each training instance.
For each of t iterations:
  Learn a classifier from weighted dataset.
  Compute error e of classifier on weighted dataset.
  If e equal to zero, or e greater or equal to 0.5:
    Terminate classifier generation.
  For each instance in dataset:
    If instance classified correctly by classifier:
       Multiply weight of instance bye / (1 - e).
  Normalize weight of all instances.
```

classification

```
Assign weight of zero to all classes.
For each of the t classifiers:
  Add -log(e / (1 - e)) to weight of class predicted
 by the classifier.
Return class with highest weight.
```

Remarks on Boosting

- Boosting can be applied without weights using re-sampling with probability determined by weights;
- Boosting decreases exponentially the training error in the number of iterations;
- Boosting works well if base classifiers are not too complex and their error doesn’t become too large too quickly!
- Boosting reduces the bias component of the error of simple classifiers!

Example

Ind ex	0	1	2	3	4	5	6	7	8	9
X	0	1	2	3	4	5	6	7	8	9
Y	1	1	1	-1	-1	-1	1	1	1	-1

Example

Index	0	1	2	3	4	5	6	7	8	9
X	0	1	2	3	4	5	6	7	8	9
Y	1	1	1	-1	-1	-1	1	1	1	-1
P	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

- T=1
- Compute the error (E) at all hypothesis ($val < threshold \; or \; val < threshold$)

Example (t1)

Index	0	1	2	3	4	5	6	7	8	9	
X	0	1	2	3	4	5	6	7	8	9	
Y	1	1	1	-1	-1	-1	1	1	1	-1	
P	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	
$x<0.5$	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0.5
$x<1.5$	1	1	-1	-1	-1	-1	-1	-1	-1	-1	0.4
$x<2.5$	1	1	1	-1	-1	-1	-1	-1	-1	-1	0.3
$x<3.5$	1	1	1	1	-1	-1	-1	-1	-1	-1	0.4
$x<4.5$	1	1	1	1	1	-1	-1	-1	-1	-1	0.5
$x<5.5$	1	1	1	1	1	1	-1	-1	-1	-1	0.6

Example (t1)

Index	0	1	2	3	4	5	6	7	8	9	
X	0	1	2	3	4	5	6	7	8	9	
Y	1	1	1	-1	-1	-1	1	1	1	-1	
P	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	
x<2.5	1	1	1	-1	-1	-1	-1	-1	-1	-1	0.3

- T=1
- Compute the error (E) at all hypothesis $(val < threshold \ or \ val < threshold)$
- Compute $\alpha = \frac{1}{2}\ln(\frac{1-\epsilon}{\epsilon})$=0.4236
- $Q_i = e^{-\alpha * Y_i * X_i}$(1.527 False,0.654 True)
- $Z = Sum(Q)$

Example (t1)

Index	0	1	2	3	4	5	6	7	8	9	
X	0	1	2	3	4	5	6	7	8	9	
Y	1	1	1	-1	-1	-1	1	1	1	-1	
P	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	
x<2.5	1	1	1	-1	-1	-1	-1	-1	-1	-1	0.3
Q	0.65	0.65	0.65	0.65	0.65	0.65	1.5	1.5	1.5	0.65	
Pnew	0.065	0.065	0.065	0.065	0.065	0.065	0.15	0.15	.015	0.065	

- Compute $\alpha = \frac{1}{2}\ln(\frac{1-\epsilon}{\epsilon})$=0.4236
- $Q_i = e^{-\alpha * Y_i * X_i}$(1.527 False,0.654 True)
- $Z = Sum(Q)$=0.9165

Example (t1)

Index	0	1	2	3	4	5	6	7	8	9	
X	0	1	2	3	4	5	6	7	8	9	
Y	1	1	1	-1	-1	-1	1	1	1	-1	
P	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	
x<2.5	1	1	1	-1	-1	-1	-1	-1	-1	-1	0.3
Pnew	0.065	0.065	0.065	0.065	0.065	0.065	0.15	0.15	.015	0.065	
P	0.07	0.07	0.07	0.07	0.07	0.07	0.167	0.167	0.167	0.07	

- $Z = Sum(Q)$=0.9165
- $f(x) = 0.423 * I(x < 2.5)$ 3 mistakes

Example (t2)

Index	0	1	2	3	4	5	6	7	8	9	
X	0	1	2	3	4	5	6	7	8	9	
Y	1	1	1	-1	-1	-1	1	1	1	-1	
P	0.07	0.07	0.07	0.07	0.07	0.07	0.167	0.167	0.167	0.07	
x<8.5	1	1	1	1	1	1	1	1	1	-1	0.2

- T=1
- Compute the error (E) at all hypothesis ($val < threshold$ or $val < threshold$)
- Compute $\alpha = \frac{1}{2}\ln(\frac{1-\epsilon}{\epsilon})$=0.6496
- $Q_i = e^{-\alpha * Y_i * X_i}$
- $Z = Sum(Q)$

Example (t2)

Index	0	1	2	3	4	5	6	7	8	9	
X	0	1	2	3	4	5	6	7	8	9	
Y	1	1	1	-1	-1	-1	1	1	1	-1	
P	0.07	0.07	0.07	0.07	0.07	0.07	0.167	0.167	0.167	0.07	
x<8.5	1	1	1	1	1	1	1	1	1	-1	0.2
Pnew	0.37	0.37	0.37	0.137	0.137	0.137	0.87	0.87	0.87	0.37	

- Compute $\alpha = \frac{1}{2}\ln(\frac{1-\epsilon}{\epsilon})$= 0.6496
- $Q_i = e^{-\alpha * Y_i * X_i}$
- $Z = Sum(Q)$=0.82

zeshan.khan@nu.edu.pk

Example (t2)

Index	0	1	2	3	4	5	6	7	8	9	
X	0	1	2	3	4	5	6	7	8	9	
Y	1	1	1	-1	-1	-1	1	1	1	-1	
P	0.07	0.07	0.07	0.07	0.07	0.07	0.167	0.167	0.167	0.07	
x<8.5	1	1	1	1	1	1	1	1	1	-1	0.2
Pnew	0.37	0.37	0.37	0.137	0.137	0.137	0.87	0.87	0.87	0.37	
P	0.45	0.45	0.45	0.167	0.167	0.167	0.106	0.106	0.106	0.045	

- $Z = Sum(Q)$=0.9165
- $f(x) = 0.423 * I(x < 2.5) + 0.649 * I(x < 8.5)$ 3 mistakes

Example (t3)

Index	0	1	2	3	4	5	6	7	8	9	
X	0	1	2	3	4	5	6	7	8	9	
Y	1	1	1	-1	-1	-1	1	1	1	-1	
P	0.45	0.45	0.45	0.167	0.167	0.167	0.106	0.106	0.106	0.045	
X>5.5	-1	-1	-1	-1	-1	-1	1	1	1	1	0.18

- T=3
- Compute the error (E) at all hypothesis ($val < threshold\ or\ val < threshold$)
- Compute $\alpha = \frac{1}{2}\ln(\frac{1-\epsilon}{\epsilon})$=7520
- $Q_i = e^{-\alpha * Y_i * X_i}$
- $Z = Sum(Q)$

Example (t3)

Index	0	1	2	3	4	5	6	7	8	9	
X	0	1	2	3	4	5	6	7	8	9	
Y	1	1	1	-1	-1	-1	1	1	1	-1	
P	0.45	0.45	0.45	0.167	0.167	0.167	0.106	0.106	0.106	0.045	
X>5.5	-1	-1	-1	-1	-1	-1	1	1	1	1	0.18
Pnew	0.096	0.096	0.096	0.078	0.078	0.078	0.05	0.05	0.05	0.096	

- Compute $\alpha = \frac{1}{2}\ln(\frac{1-\epsilon}{\epsilon})$=0.4236
- $Q_i = e^{-\alpha * Y_i * X_i}$(1.527 False,0.654 True)
- $Z = Sum(Q)$=0.9165

Example (t3)

Index	0	1	2	3	4	5	6	7	8	9	
X	0	1	2	3	4	5	6	7	8	9	
Y	1	1	1	-1	-1	-1	1	1	1	-1	
P	0.45	0.45	0.45	0.167	0.167	0.167	0.106	0.106	0.106	0.045	
X>5.5	-1	-1	-1	-1	-1	-1	1	1	1	1	0.18
Pnew	0.096	0.096	0.096	0.078	0.078	0.078	0.05	0.05	0.05	0.096	
P	0.125	0.125	0.125	0.102	0.102	0.102	0.064	0.064	0.064	0.125	

- $Z = Sum(Q)$=0.771
- $f(x) = 0.423 * I(x < 2.5) + 0.649 * I(x < 8.5) + 0.752I(x > 5.5)$ 0 mistakes

Stacking

- Uses *meta learner* instead of voting to combine predictions of base learners
  - Predictions of base learners (*level-0 models*) are used as input for meta learner (*level-1 model)*
- Base learners usually different learning schemes
- Hard to analyze theoretically: “black magic”

Some Practical Advices

- If the classifier is unstable (high variance), then apply bagging!
- If the classifier is stable and simple (high bias) then apply boosting!
- If the classifier is stable and complex then apply randomization injection!
- If you have many classes and a binary classifier then try error-correcting codes! If it does not work then use a complex binary classifier!

Diversity Measures

- Most Popular
  - Plain Disagreement Measure
  - Entropy

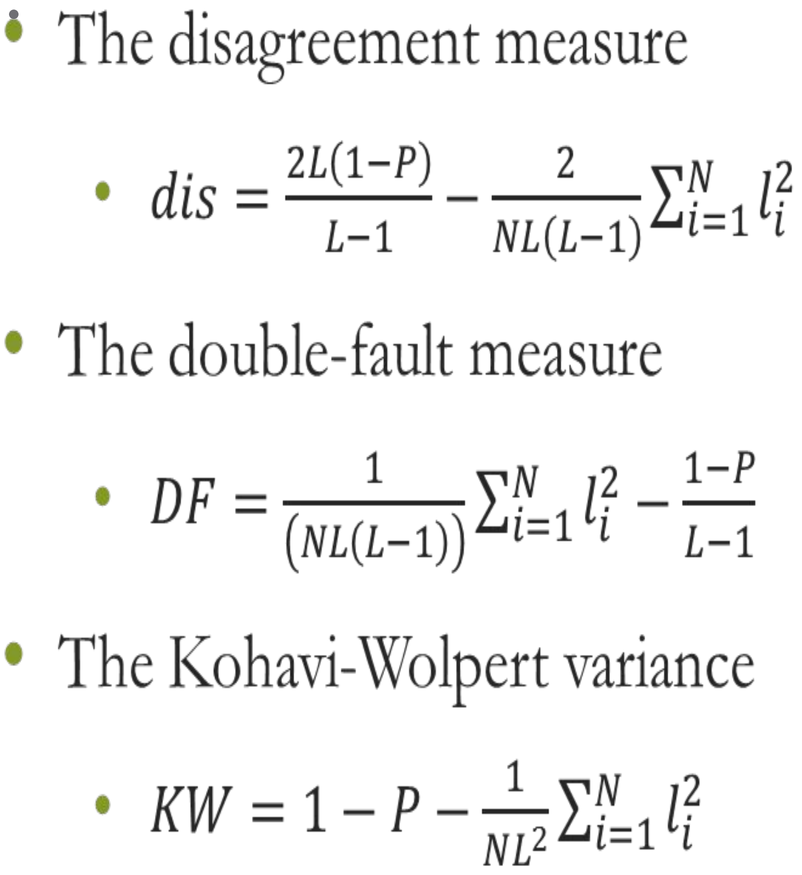
Diversity Measure

- Two classifiers A and B the disagreement is D as:
- $D = \left(\frac{1}{N_s}\right) \sum_{k=1}^{N_s} Diff(C_a(S_k), C_b(S_k),)$
- Where
  - N_s is the number of samples in dataset
  - $C_i(S_k)$ is the class assigned by classifier I to sample k
  - $Diff(a, b) = int(a \neq b)$
    - if a==b then 0 otherwise 1

Diversity Measures

- L: total number of base classifiers
- N: total number of training samples
- m_i: margin of an ensemble on the training sample x_i
- P: average classification accuracy of the base classifiers on the training data
- p_j: classification accuracy of the base classifier h_j
- l_i: product of L and sum of the weights of the base classifiers that classify the training sample xi incorrectly, $l_i = L \sum_{O_{ij}=-1} w_j$
- O_{ij}: oracle output of the classifier h_j on the training sample x_i
- div: diversity among the base classifiers in the ensemble
- E. K. Tang, P. N. Suganthan, X. Yao “An analysis of diversity measures” Springer Science + Business Media, LLC 2006.

Diversity Measures



- The disagreement measure
  - $dis = \frac{2L(1-P)}{L-1} - \frac{2}{NL(L-1)} \sum_{i=1}^{N} l_i^2$
- The double-fault measure
  - $DF = \frac{1}{(NL(L-1))} \sum_{i=1}^{N} l_i^2 - \frac{1-P}{L-1}$
- The Kohavi-Wolpert variance
  - $KW = 1 - P - \frac{1}{NL^2} \sum_{i=1}^{N} l_i^2$

Diversity Measures

- The measurement of inter-rater agreement
  - $K = \frac{LP-P-L}{LP-P} + \frac{\sum_{i=1}^{N} l_i^2}{NL(L-1)P(1-P)}$
- The generalized diversity
  - $GD = \frac{L}{L-1} - \frac{\sum_{i=1}^{N} l_i^2}{NL(L-1)\ (1-P)}$
- The measure of Difficulty
  - $diff\ = \frac{1}{NL}\sum_{i=1}^{N} l_i^2 - L(1-P)^2$

If S is the number of base classifiers, then the entropy is defined as:

$$\text{Entropy} = \frac{1}{N_s}\sum_{a=1}^{N_s}\sum_{b=1}^{C} -\frac{N_b^a}{S} * log(\frac{N_b^a}{S})$$

where N_s is the number of samples in the data set, C is the number of classes and N_b^a is the number of base classifiers that assign sample a to class b. In order to keep this measure of diversity within the range [0,1] the logarithm should be taken to the base C.

Sklearn Ensemble

- $sklearn \rightarrow ensemble \rightarrow \ RandomForestClassifier$
- $sklearn \rightarrow ensemble \rightarrow ExtraTreesClassifier$
- $sklearn \rightarrow ensemble \rightarrow AdaBoostClassifier$
- $sklearn \rightarrow ensemble \rightarrow GradientBoostingClassifier$
- $sklearn \rightarrow ensemble \rightarrow HistGradientBoostingClassifier$
- $sklearn \rightarrow ensemble \rightarrow VotingClassifier$
- $sklearn \rightarrow ensemble \rightarrow VotingRegressor$
- $sklearn \rightarrow ensemble \rightarrow StackingRegressor$