Ensemble Classification

AI42001

Different classifiers, different results

$f1(X) = Y1$

$f2(X) = Y2$

………

$fK(X) = YK$

- How to choose a single answer??

Simple Voting

f1(X) = Y1

f2(X) = Y2

………

fK(X) = YK

- How to choose a single answer??
- Classification: mode of all predictions!
- Regression: mean of all predictions!

Simple Voting

- How to choose a single answer??

classifier	f1	f2	f3	f4	f5	f6
Label	1	1	2	1	3	2

Label	Y=1	Y=2	Y=3
Total Votes	3	2	1

- Classification: mode of all predictions!
- Regression: mean of all predictions!
- Final prediction: Y =1 !

Simple Voting

- How to choose a single answer??

classifier	f1	f2	f3	f4	f5	f6
Label	1	1	2	1	3	2

Label	Y=1	Y=2	Y=3
Total Votes	3	2	1

- Classification: mode of all predictions!
- Regression: mean of all predictions!
- Final prediction: Y =1 !

Simple Voting

f1(X) = Y1

f2(X) = Y2

………

fK(X) = YK

- How to choose a single answer??

- Classification: mode of all predictions!
- Regression: mean of all predictions!
- But if all classifiers are not equally reliable????

Weighted Voting

- Attach a “weight” with each classifier
- More reliable classifier -> higher weightage

classifier	f1	f2	f3	f4	f5	f6
Label	1	1	2	1	3	2
Weight	7	5	9	5	7	9

Weighted Voting

- Attach a “weight” with each classifier
- More reliable classifier -> higher weightage

classifier	f1	f2	f3	f4	f5	f6
Label	1	1	2	1	3	2
Weight	7	5	9	5	7	9

Label	Y=1	Y=2	Y=3
Total Weight	17	18	7

- Final prediction: Y = 2!

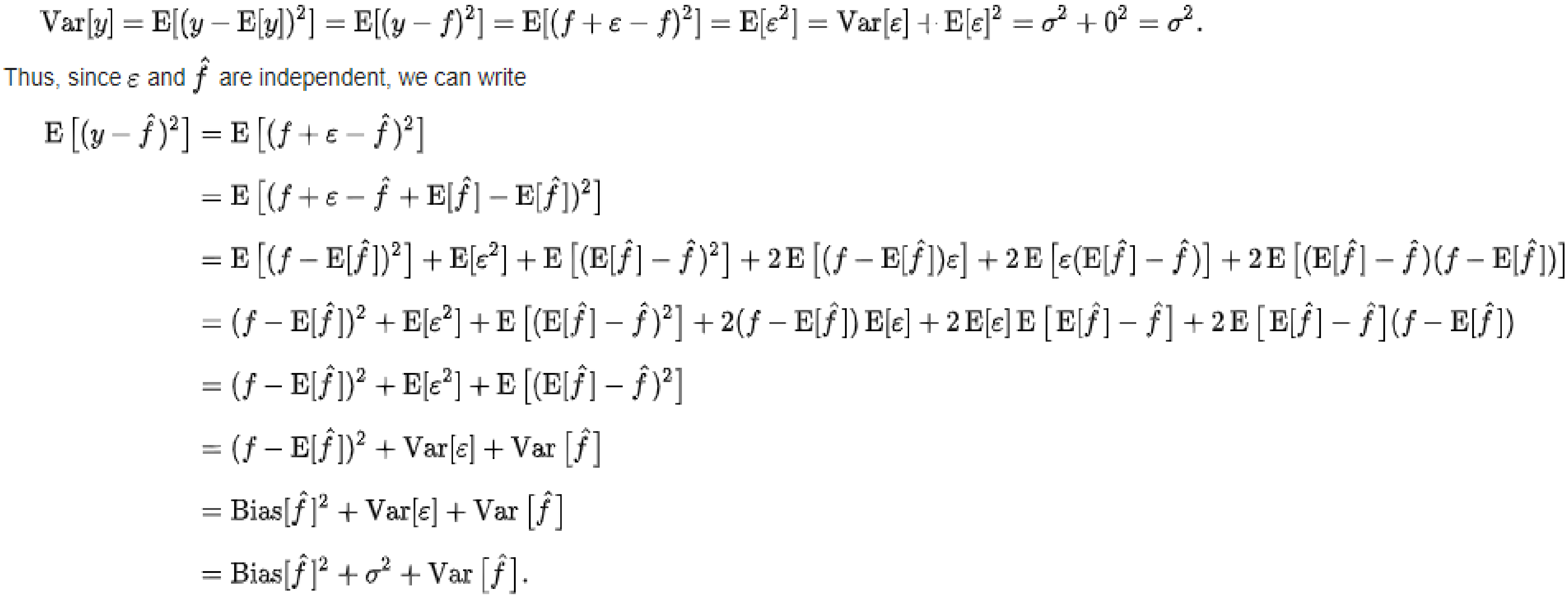
How to have many classifiers?

- Same training set, different classifier functions
- Split the dataset into multiple parts; train a classifier on each part
- Select different subsets of features; train a classifier for each subset

- Found to reduce overfitting
- Improves “stability”, reduces “variance” (Similar examples assigned similar/same label)

Bias-variance Tradeoff

- y: true label = f + ε [f: true function, ε: random noise]
- f^: predictor function, learnt from a random dataset



$$\operatorname{Var}[y] = \operatorname{E}[(y - \operatorname{E}[y])^2] = \operatorname{E}[(y - f)^2] = \operatorname{E}[(f + \varepsilon - f)^2] = \operatorname{E}[\varepsilon^2] = \operatorname{Var}[\varepsilon] + \operatorname{E}[\varepsilon]^2 = \sigma^2 + 0^2 = \sigma^2.$$

Thus, since ε and $\hat{f}$ are independent, we can write

$$\begin{aligned}
\operatorname{E}\left[(y - \hat{f})^2\right] &= \operatorname{E}\left[(f + \varepsilon - \hat{f})^2\right] \\
&= \operatorname{E}\left[(f + \varepsilon - \hat{f} + \operatorname{E}[\hat{f}] - \operatorname{E}[\hat{f}])^2\right] \\
&= \operatorname{E}\left[(f - \operatorname{E}[\hat{f}])^2\right] + \operatorname{E}[\varepsilon^2] + \operatorname{E}\left[(\operatorname{E}[\hat{f}] - \hat{f})^2\right] + 2\operatorname{E}\left[(f - \operatorname{E}[\hat{f}])\varepsilon\right] + 2\operatorname{E}\left[\varepsilon(\operatorname{E}[\hat{f}] - \hat{f})\right] + 2\operatorname{E}\left[(\operatorname{E}[\hat{f}] - \hat{f})(f - \operatorname{E}[\hat{f}])\right] \\
&= (f - \operatorname{E}[\hat{f}])^2 + \operatorname{E}[\varepsilon^2] + \operatorname{E}\left[(\operatorname{E}[\hat{f}] - \hat{f})^2\right] + 2(f - \operatorname{E}[\hat{f}])\operatorname{E}[\varepsilon] + 2\operatorname{E}[\varepsilon]\operatorname{E}\left[\operatorname{E}[\hat{f}] - \hat{f}\right] + 2\operatorname{E}\left[\operatorname{E}[\hat{f}] - \hat{f}\right](f - \operatorname{E}[\hat{f}]) \\
&= (f - \operatorname{E}[\hat{f}])^2 + \operatorname{E}[\varepsilon^2] + \operatorname{E}\left[(\operatorname{E}[\hat{f}] - \hat{f})^2\right] \\
&= (f - \operatorname{E}[\hat{f}])^2 + \operatorname{Var}[\varepsilon] + \operatorname{Var}\left[\hat{f}\right] \\
&= \operatorname{Bias}[\hat{f}]^2 + \operatorname{Var}[\varepsilon] + \operatorname{Var}\left[\hat{f}\right] \\
&= \operatorname{Bias}[\hat{f}]^2 + \sigma^2 + \operatorname{Var}\left[\hat{f}\right].
\end{aligned}$$

How to have many classifiers?

- Same training set, different classifier functions
- Split the dataset into multiple parts; train a classifier on each part
- Select different subsets of features; train a classifier for each subset

- Bootstrap Aggregation (BAGG-ing)!

Bootstrap Aggregation

- Given a training set of size N:
- Choose N examples from it with replacement
- i-th draw: each training example may be chosen with probability 1/N
- Same example may be chosen multiple times!

Bootstrap Aggregation

- Given a training set of size N:
- Choose N examples from it with replacement
- i-th draw: each training example may be chosen with probability 1/N
- Same example may be chosen multiple times!

X1	X2	X3	X4	X5	X6	X7	X8	X9	X10

Bootstrap Aggregation

- Given a training set of size N:
- Choose N examples from it with replacement
- i-th draw: each training example may be chosen with probability 1/N
- Same example may be chosen multiple times!

X1	X2	X3	X4	X5	X6	X7	X8	X9	X10

X5	X9	X3	X4	X5	X1	X1	X6	X7	X9
X6	X3	X4	X10	X3	X7	X3	X6	X8	X2
X4	X7	X9	X5	X3	X6	X2	X6	X1	X6

Bootstrap Aggregation

- Given a training set of size N:
- Choose N examples from it with replacement
- i-th draw: each training example may be chosen with probability 1/N
- Same example may be chosen multiple times!

- Prob(X_j is never chosen) ~ 0.36
- Nearly one-third of original dataset will not be present in new dataset!

Bootstrap Aggregation

- Use BAGG to generate K “versions” of original training set
- Train a classifier on each version
- Give weightage to each classifier according to its accuracy on validation set

BAGGing of features

- Instead of choosing subsets of training examples
- We can choose subsets of the features!

- Each "version" has all the training examples, but only some of the features!

X(1)
X(2)
X(3)
X(4)
X(5)

X(4)
X(2)
X(3)
X(3)
X(2)

X(2)
X(1)
X(4)
X(3)
X(3)

X(4)
X(2)
X(3)
X(2)
X(3)

BAGGing of features

- Instead of choosing subsets of training examples
- We can choose subsets of the features!

- Each “version” has all the training examples, but only some of the features!

- If the classifier used is Decision Tree, then this approach is called Random Forest
- RF is one of the more powerful classifiers around

Weighted Training

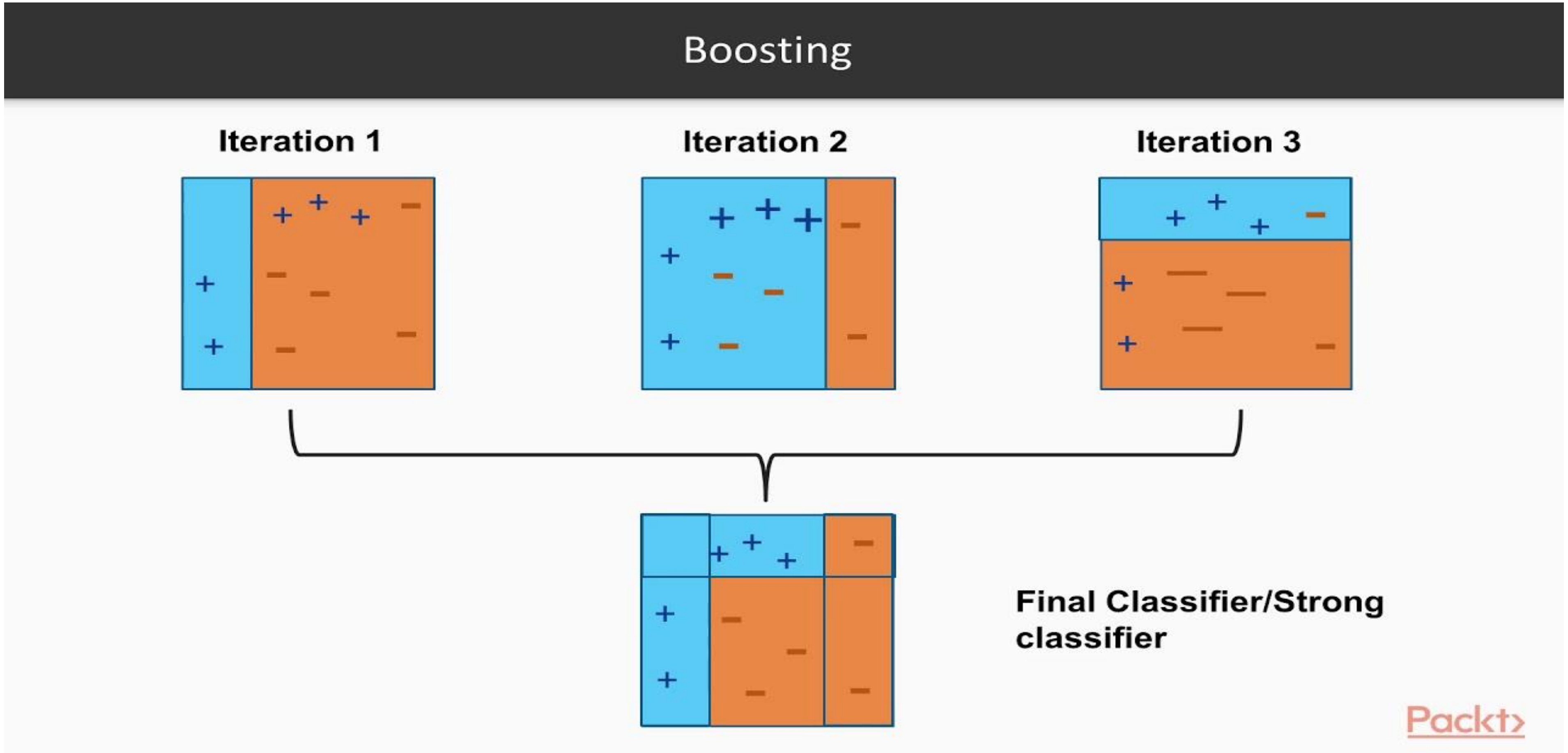
- What if the training samples have different weights?
- Some examples more important than the others!
- The “loss function” has to account for these weights!

- Weighted K-NN: consider weighted vote of each class label among the K nearest neighbors
- Weighted Decision Tree: at each node, consider “weighted relative frequencies” of each class label

Boosting

- 1) Take a “weak” classifier (just better than random)
- 2) Compute the error of the model on each training example
- 3) Identify the examples on which it makes mistakes
- 4) Increase the “weights” of such examples and retrain classifier
- 5) Add the updated classifier to the ensemble
- 6) Goto 2

Boosting


Boosting
Iteration 1
Iteration 2
Iteration 3
Final Classifier/Strong classifier
Packt>

Adaboost Algorithm

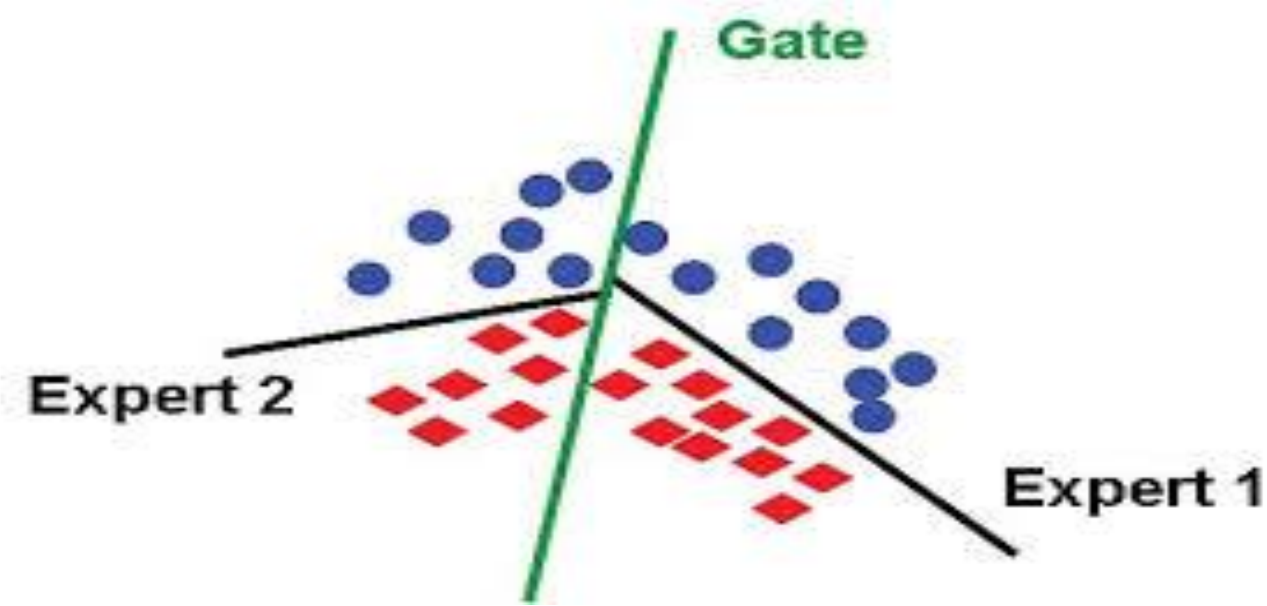
- Initialize weights of each training sample w_i = 1/N
- For iteration t = 1 to max_iter
    - Learn a classifier h_t on training examples according to weights w
    - Calculate weighted error : $e_t = \sum_I w_i \mathbf{I}(h_t(X_i) \mathrel{!=} y_t)$
    - Set weightage of h_t : $a_t = 0.5 \log ((1-e_t) / e_t)$
    - Update the weight of each training example

      $w_i = w_i \exp(-a_t)$ if $h_t(x_i) = y_i$ (correct classification: decrease weight)

      $w_i = w_i \exp(a_t)$ if $h_t(x_i) \mathrel{!=} y_i$ (wrong classification: increase weight)
    - Normalize the weights so that they add up to 1
- End
- Output: all the classifiers “h” along with their weights “a”

Mixture of Experts

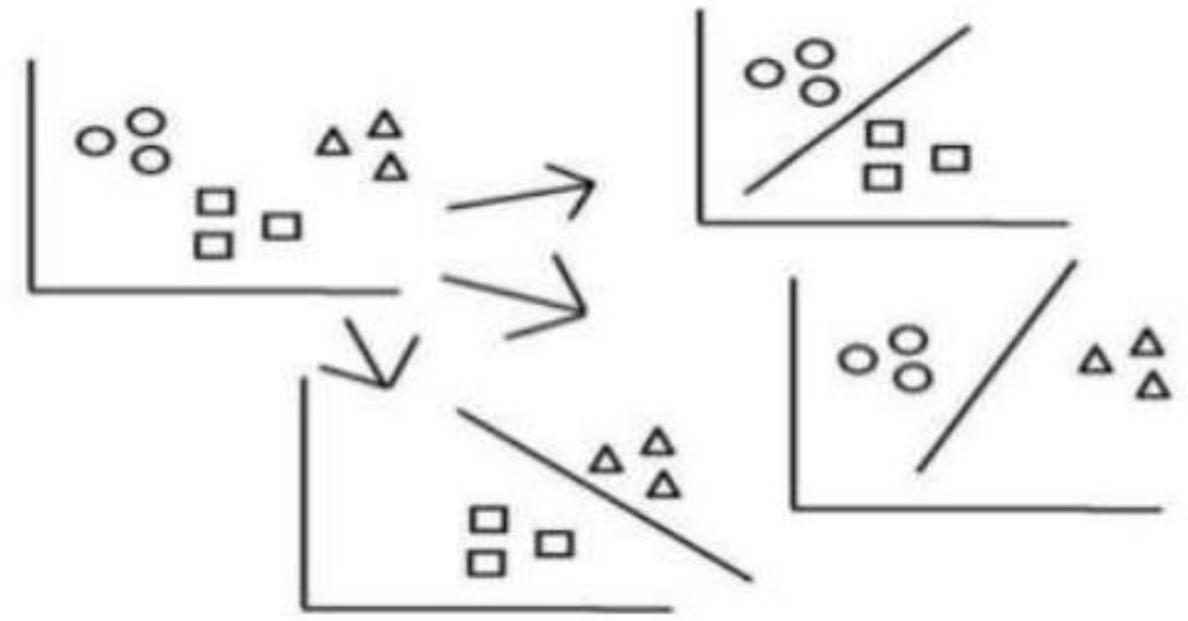
- Different classifiers may be more effective in different parts of feature space
- Weights of classifiers should be dependent on features



Summary

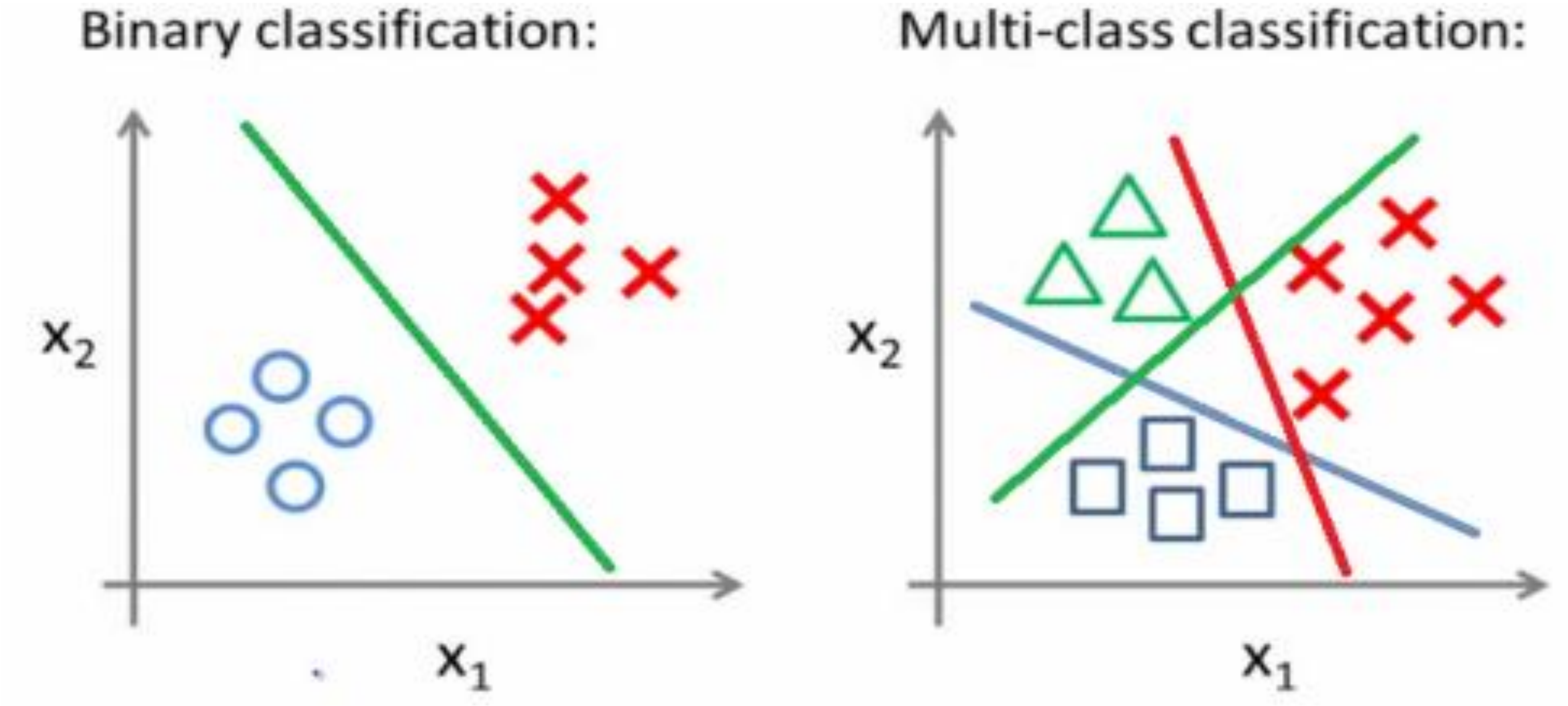
- Complex learning problems can often be solved by effectively combining multiple weak model via ensemble learning methods
- Simple ones: Voting/averaging or stacking
- Bagging: Random Forests
- Boosting: Adaboost
- Mixture of Experts
- These models help reduce variance or overfitting, and may have computational benefits over more complex classification algorithms

One-vs-one Classification



One-vs-One (OVO)

One-vs-all Classification


Binary classification:
Multi-class classification:
x_2
x_1
x_2
x_1