Ensemble Learning

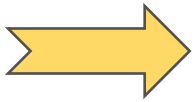
INFO-4604, Applied Machine Learning
University of Colorado Boulder

November 27, 2018

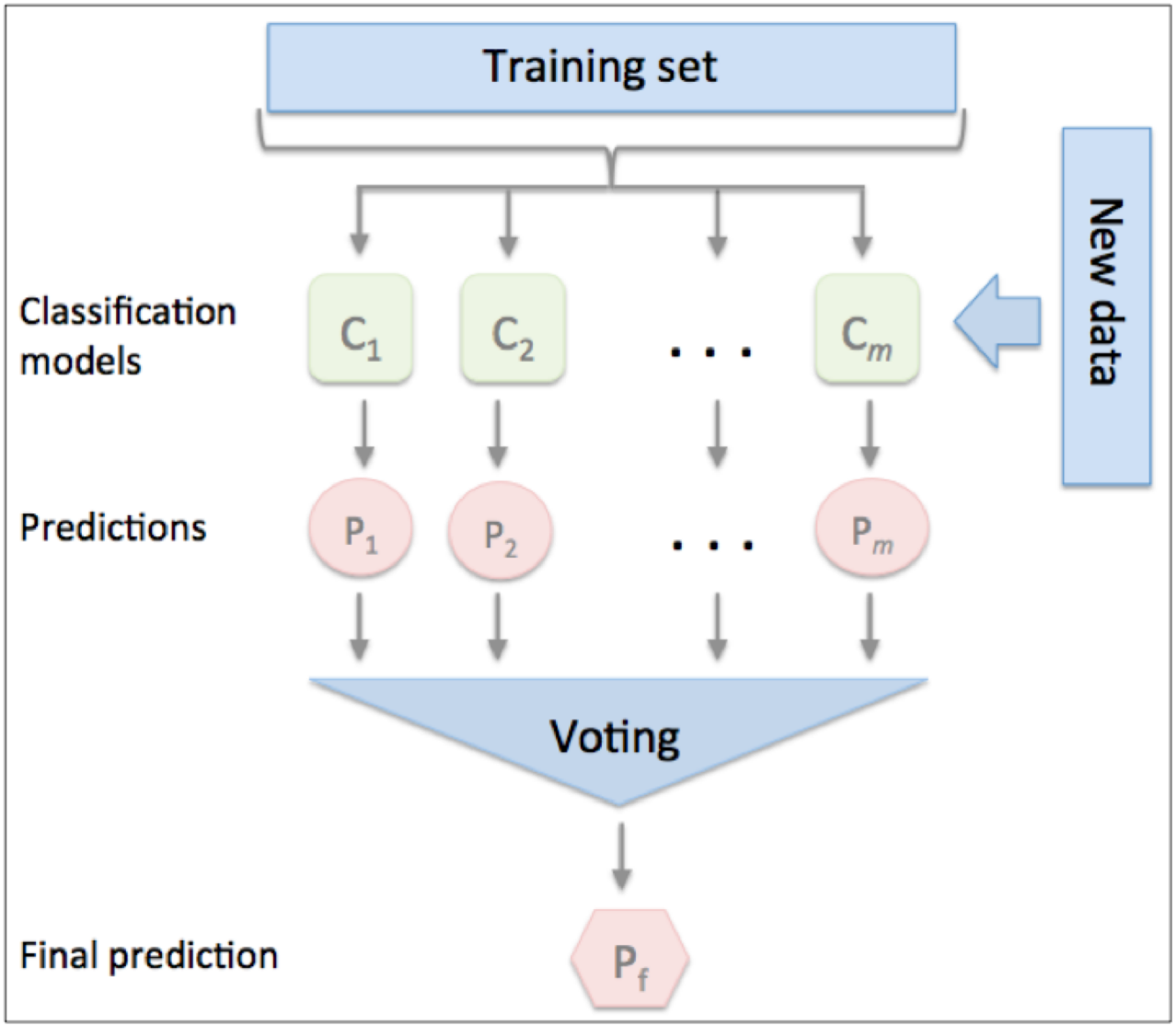
Prof. Michael Paul

Ensemble Learning

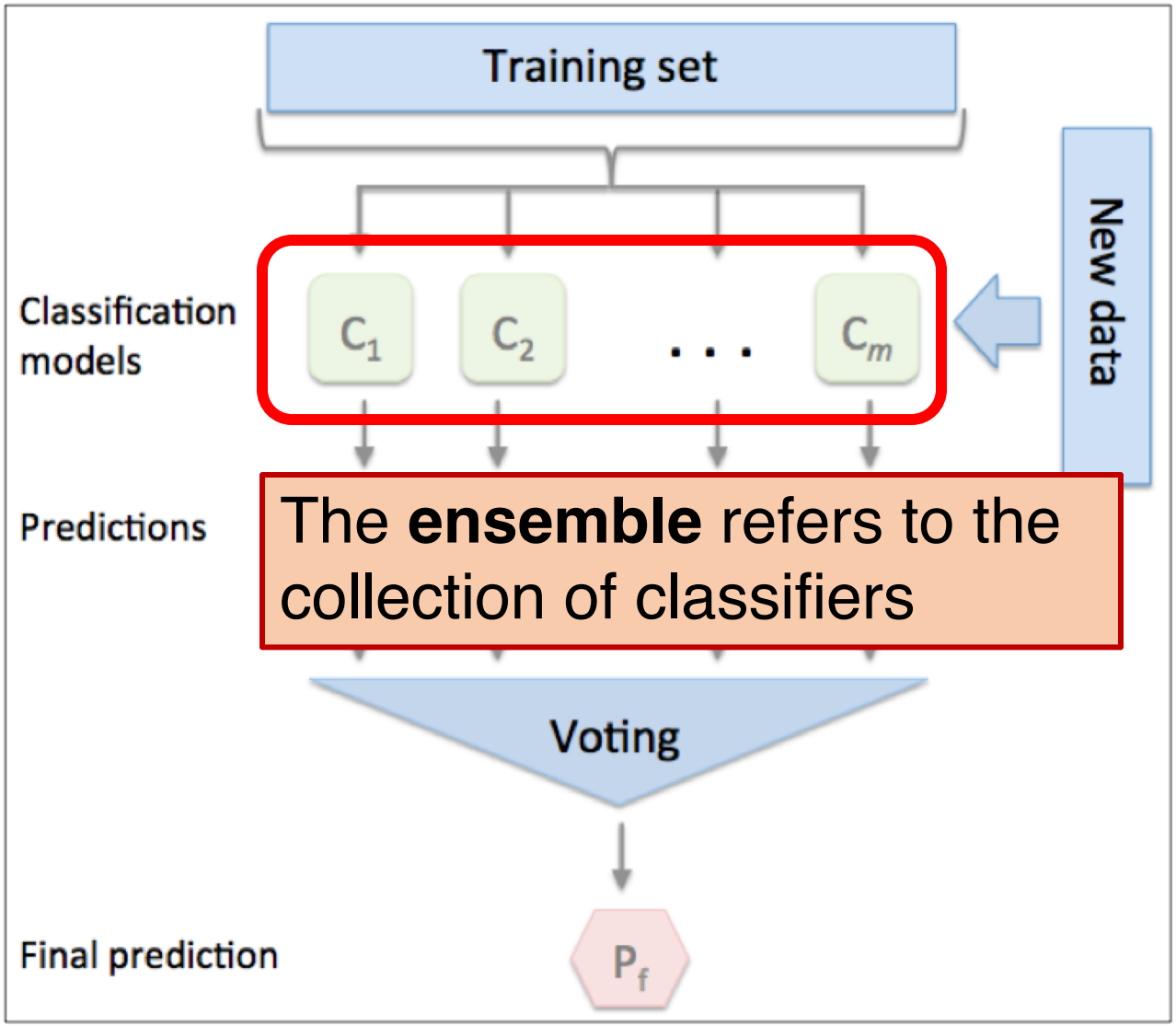
General idea:
Combine multiple classifiers (usually with different strengths) to build a bigger, better classifier



Ensemble Learning

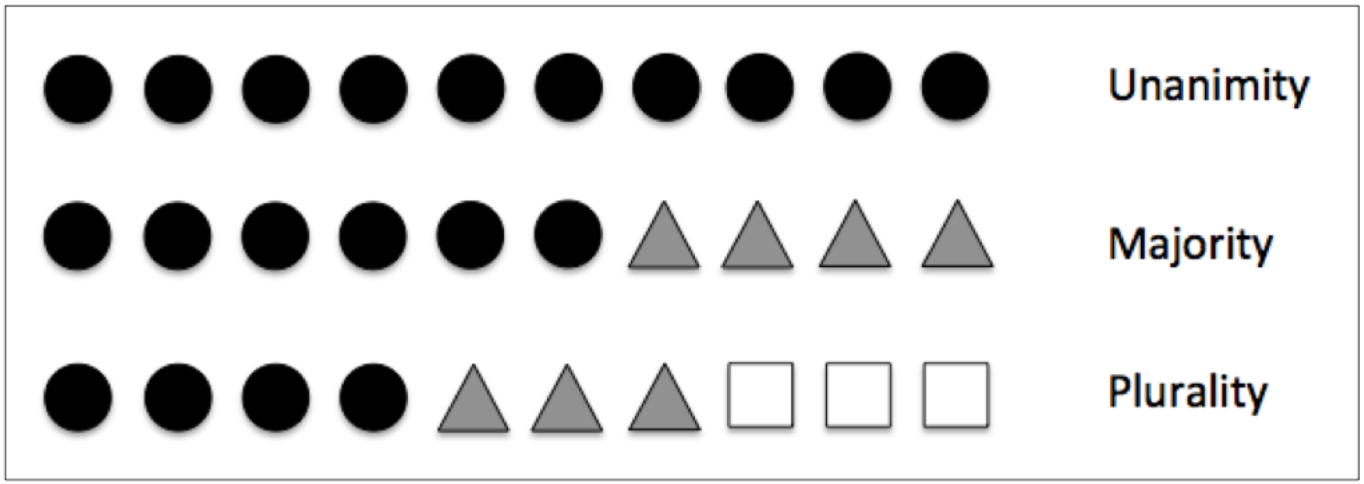


Ensemble Learning



Ensemble Learning

Most ensemble methods use the **majority vote** principle (or **plurality vote** in multiclass setting)



Votes can also be **weighted** so that some classifiers have more influence than others

Ensemble Learning

Why are ensembles helpful?

If the classifiers in the ensemble make different errors (i.e., the errors are not correlated), then the probability of many classifiers making the same error is much smaller than the probability of any one classifier

- In practice, errors are usually somewhat correlated – but various ensemble methods try to learn classifiers with different strengths/weaknesses
- If classifiers are too similar, then an ensemble won't provide a benefit – though ensembles usually don't hurt

Weighting

How to weigh the votes of different classifiers?

- *Learn* the best way to combine classifiers
  - Use the classifier outputs as features in a new classifier, learn weights for the features
  - Sometimes called *stacking*
  - Can use the predicted labels as features, or the predicted probabilities/scores

Weighting

How to weigh the votes of different classifiers?

Methods without learning:

- Weigh the votes by the confidence of the classifier
  - e.g., the score or probability
  - Needs to be the same type of classifier in the ensemble for this weighting to be comparable across classifiers
- Weigh the votes by the accuracy of the classifier
  - Trust accurate classifiers more

Weighting

Note that weighting is not required

- Many methods will use a simple majority vote
- Effective if number of classifiers is large and performance is comparable

Creating an Ensemble

How to obtain multiple classifiers?

- Different algorithms (e.g., SVM, kNN)
- Train on different datasets or feature sets
  - Random sampling the data
- Independently created systems
  - e.g., combine multiple entries in a competition

Creating an Ensemble

How to obtain multiple classifiers?

- Different algorithms (e.g., SVM, kNN)
- **Train on different datasets or feature sets**
  - **Random sampling the data**
- Independently created systems
  - e.g., combine multiple entries in a competition

Bagging

Bagging involves creating different classifiers on different random samples of training data

- Training instances are sampled *with replacement* (can sample the same instance more than once) to create a new training set with the same number of instances
- This process is repeated *K* times to create *K* classifiers

The sampling process is called *bootstrap* sampling

- Traditionally used for constructing confidence intervals
- Bagging = **B**ootstrap **agg**regation

Bagging

Another version of bagging is to randomly sample *features* (creating different classifiers on different versions of the feature set)

- Referred to as *feature bagging*
- Less common than instance bagging

Bagging

Remember *random forests*:

- Create different decision trees on random subsets of training data, then take majority vote
- Random forests are an example of bagging
  - Random forests often work well, even though individual decision trees often work poorly
  - Hard to create a decision tree that is accurate without overfitting – ensembles can keep the strengths while reducing the weakness

Bagging

Bagging reduces *variance*, but not bias

- Even if individual classifiers are overfitted, they tend to be overfitted in different ways, so the final ensemble reduces overfitting
- If the individual classifiers are underfitted due to the same bias, the ensemble will also be underfitted

Therefore, classifiers in an ensemble typically are trained to have high variance (e.g., less regularization, deeper decision trees)

Boosting

Rather than training multiple classifiers independently, **boosting** works *iteratively*

- In each iteration, a new classifier is trained to try to correctly classify instances that the previous classifier got wrong
- In doing so, the new classifier might now get wrong instances that the previous classifier got correct
- The goal is to learn classifiers that perform better on different instances

Boosting

In boosting, the classifiers are usually *weak learners*

- Classifiers that perform slightly better than random

A common type of weak learner is a *decision stump*

- A decision tree with only one node (prediction based on one feature)

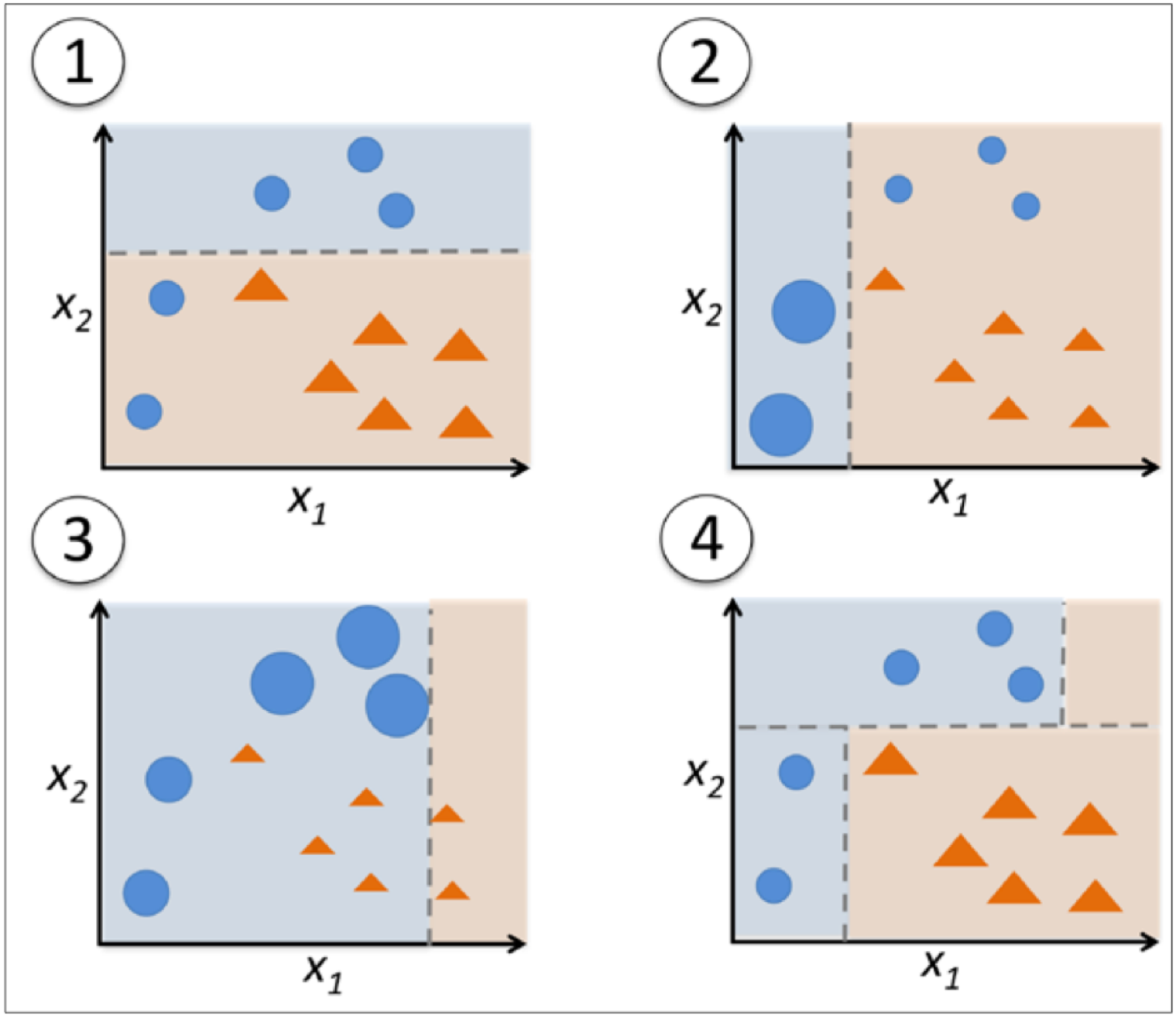
Why weak instead of strong learners?

- Mainly efficiency – boosting theory shows that weak learners are just as good, so no need for large models
- Weak learners also more diverse (better for ensemble)

Boosting

AdaBoost is a common boosting algorithm

- Uses all training data in every iteration
- But weighs the training instances differently in each iteration, so the classifier is encouraged to get certain instances correct over others


1
x_2
x_1
2
x_2
x_1
3
x_2
x_1
4
x_2
x_1