Ensemble Learning

Review

- Last week:
  - Evaluating Machine Learning Models Using Cross-
  - Validation Naïve Bayes
  - Support Vector Machines

- Assignments (Canvas):
  - Problem set 4 due tonight
  - Lab assignment 2 due next week
  - Project pre-proposal due in two weeks (finding a partner ideas)

- Questions?

Today’s Topics

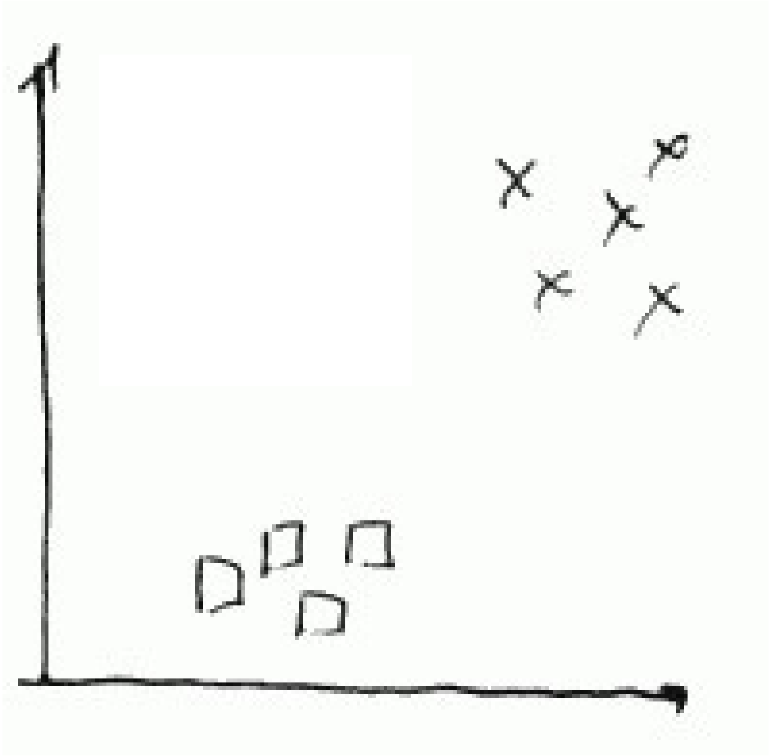
- One-vs-all multiclass classification
- Classifier confidence
- Evaluation: ROC and PR-curves
- Ensemble learning

Today's Topics

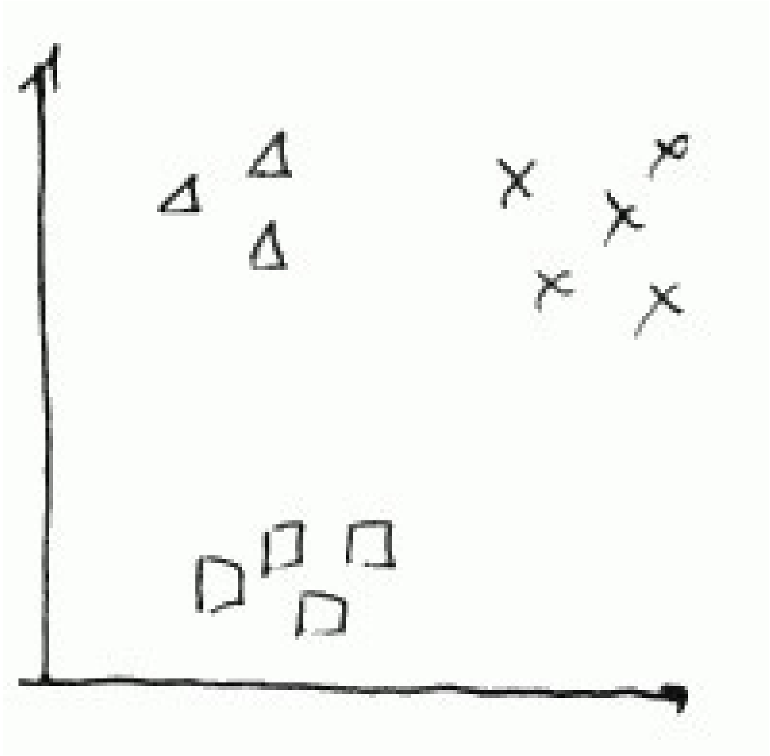
- One-vs-all multiclass classification
- Classifier confidence
- Evaluation: ROC and PR-curves
- Ensemble learning

Recall: Binary vs Multiclass Classification

Binary: distinguish 2 classes



Multiclass: distinguish 3+ classes



Recall: Binary vs Multiclass Classification

Binary: distinguish 2 classes

Perceptron

Adaline

Support Vector Machine

Multiclass: distinguish 3+ classes

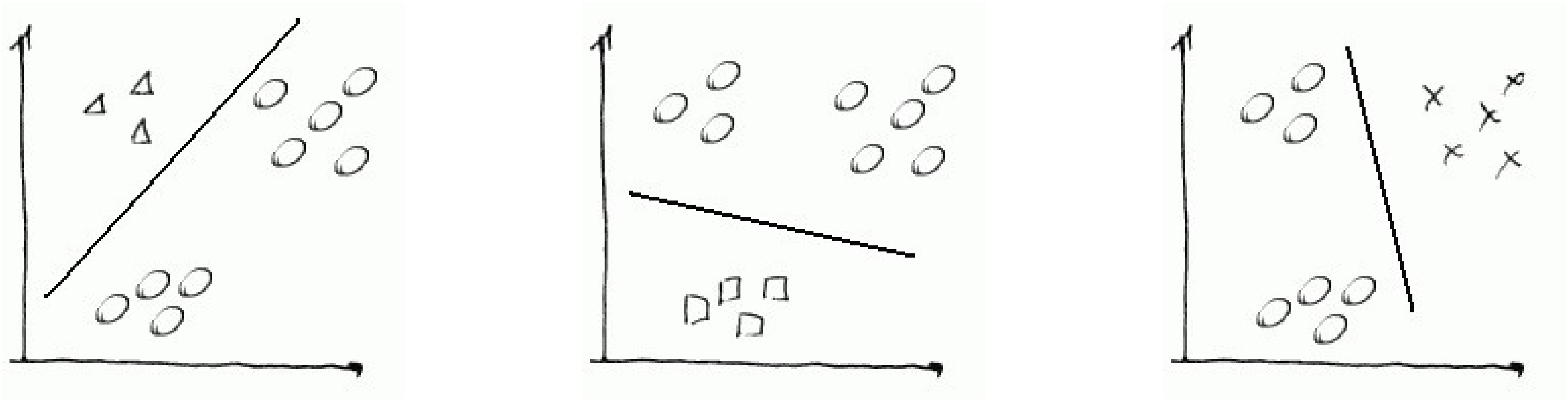
Nearest Neighbor

Decision Tree

Naïve Bayes

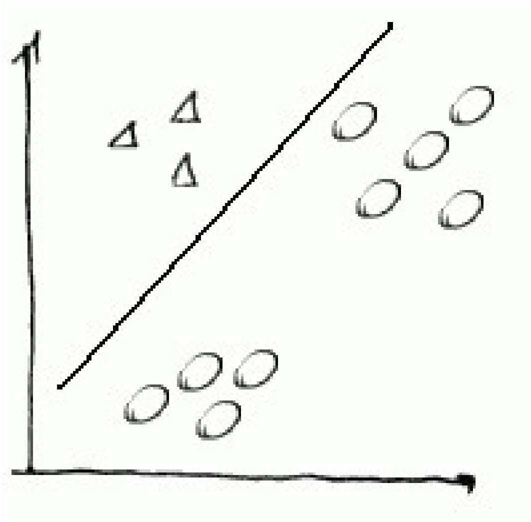
One-vs-All (aka, One-vs-Rest): Applying Binary Classification Methods for Multiclass Classification

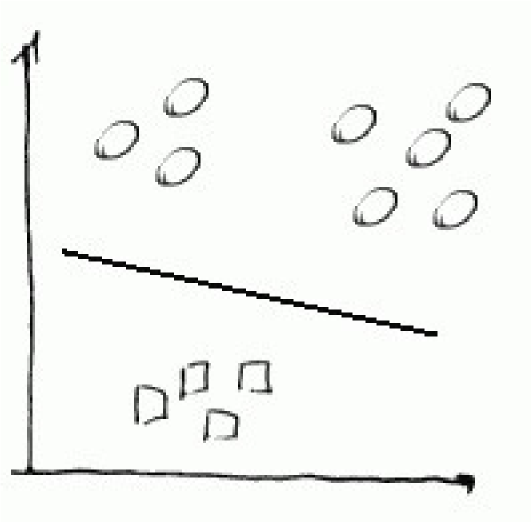
- Given ‘N’ classes, train ‘N’ different classifiers: a single classifier trained per class, with the samples of that class as positive samples and all other samples as negatives; e.g.,

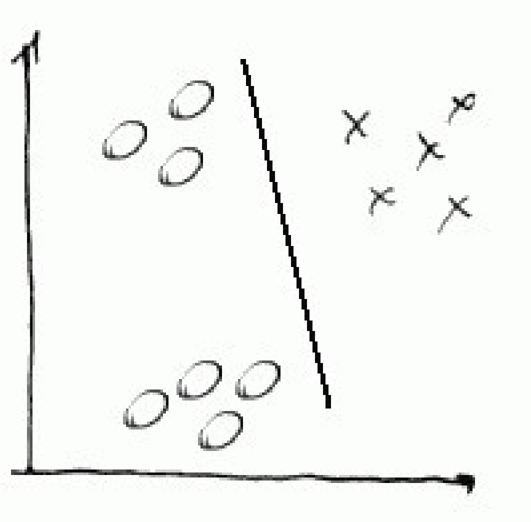


One-vs-All (aka, One-vs-Rest): Limitation

- Often leads to unbalanced distributions during learning; i.e., when the set of negatives is much larger than the set of positives

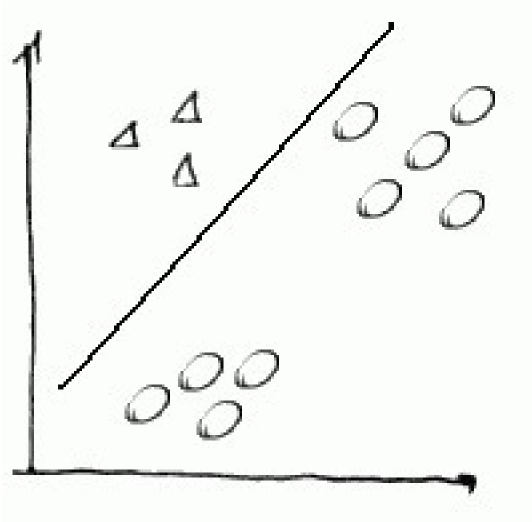


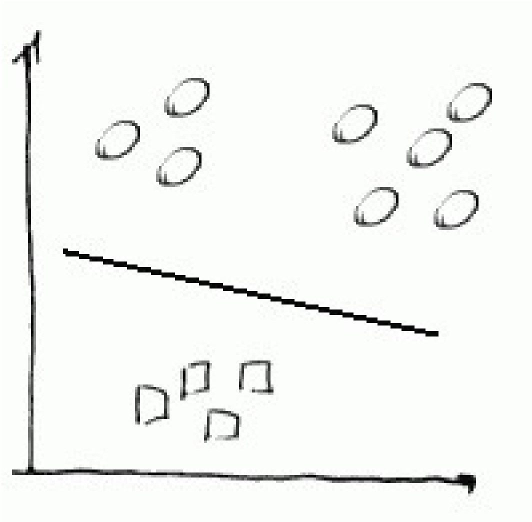


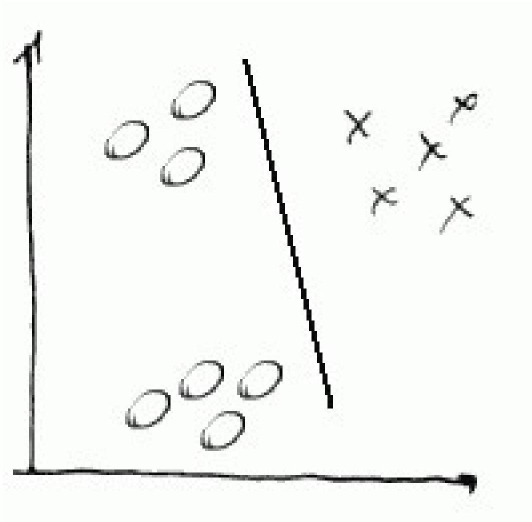


One-vs-All (aka, One-vs-Rest): Class Assignment

- (Imperfect) Approach: use from N classifiers the most confident match; this requires a real-valued confidence score for its decision







Today's Topics

- One-vs-all multiclass classification
- Classifier confidence
- Evaluation: ROC and PR-curves
- Ensemble learning

Classifier Confidence: Beyond Classification

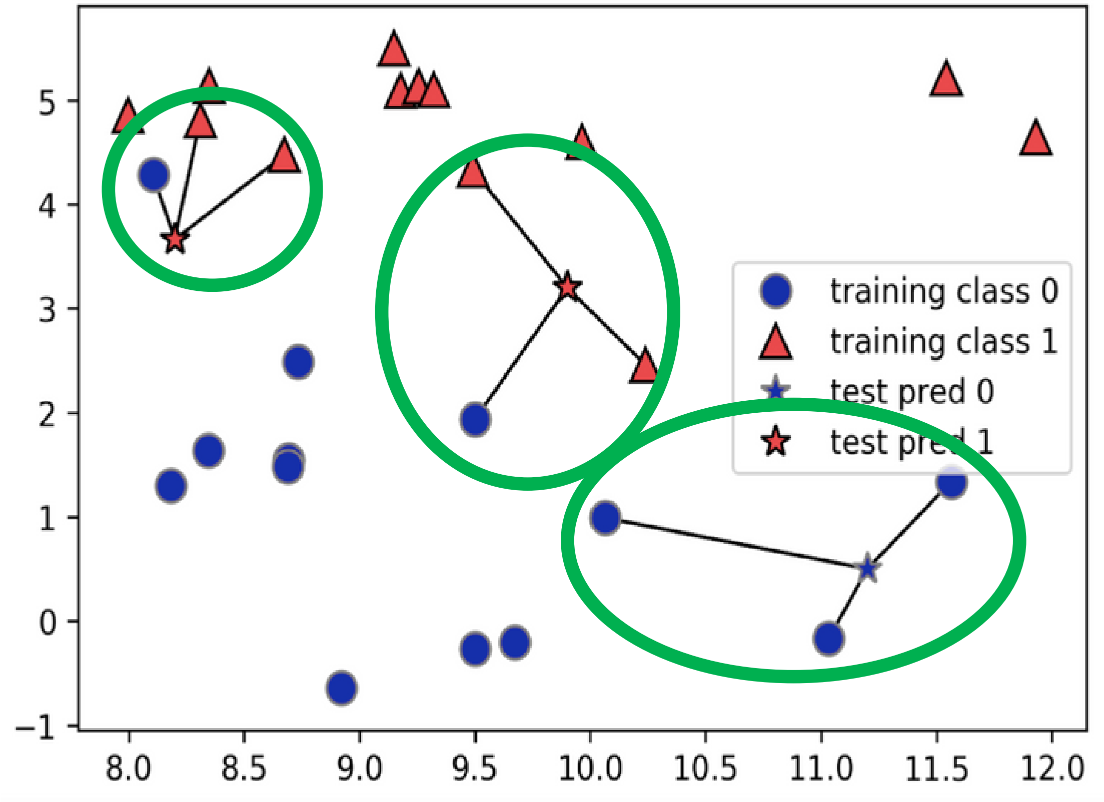
- Indicate both the predicted class and uncertainty about the choice

- When and why might you want to know about the uncertainty?
  - e.g., weather forecast: 25% chance it will rain today
  - e.g., medical treatment: when unconfident, start a patient on a drug at a lower dose and decide later whether to change the medication or dose

Classifier Confidence: How to Measure for K-Nearest Neighbors?

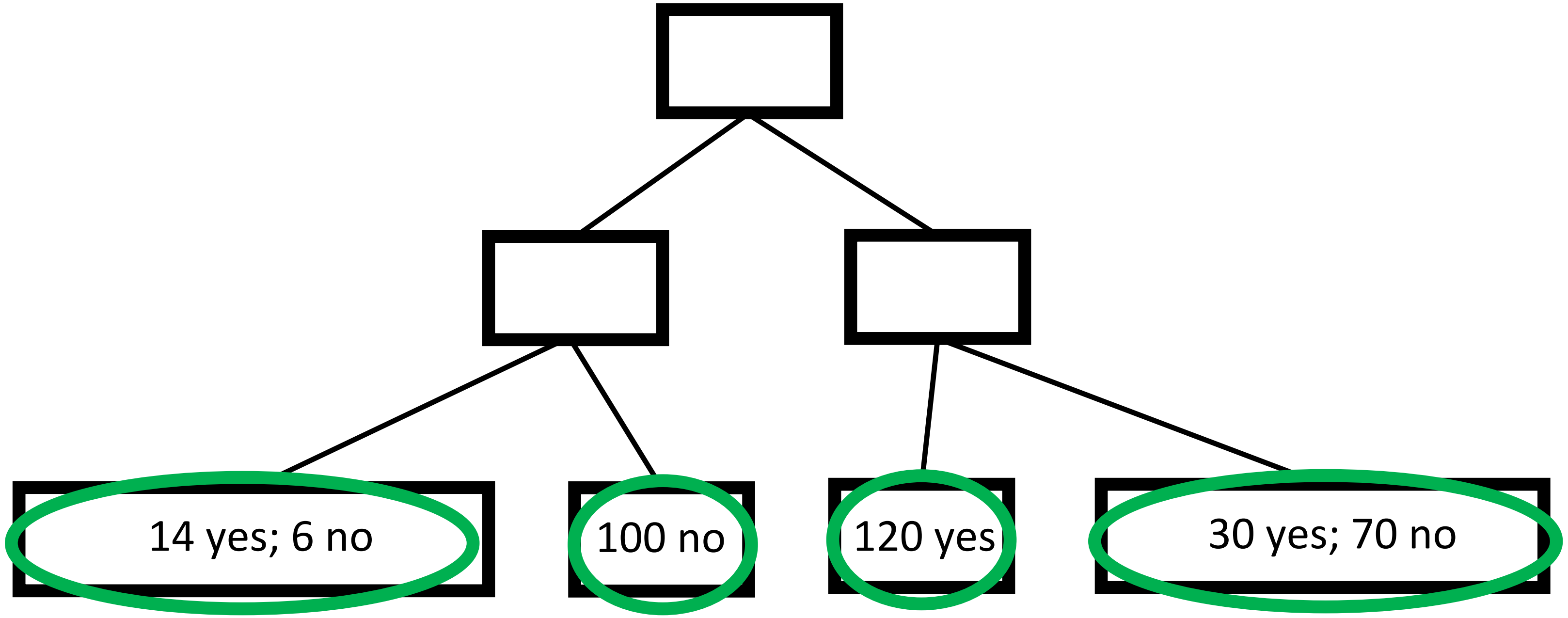
- Proportion of neighbors with label *y*; e.g.,

When K=3:



Classifier Confidence: How to Measure for Decision Trees?

- Proportion of training samples with label *y* in the leaf where for the test sample; e.g.,

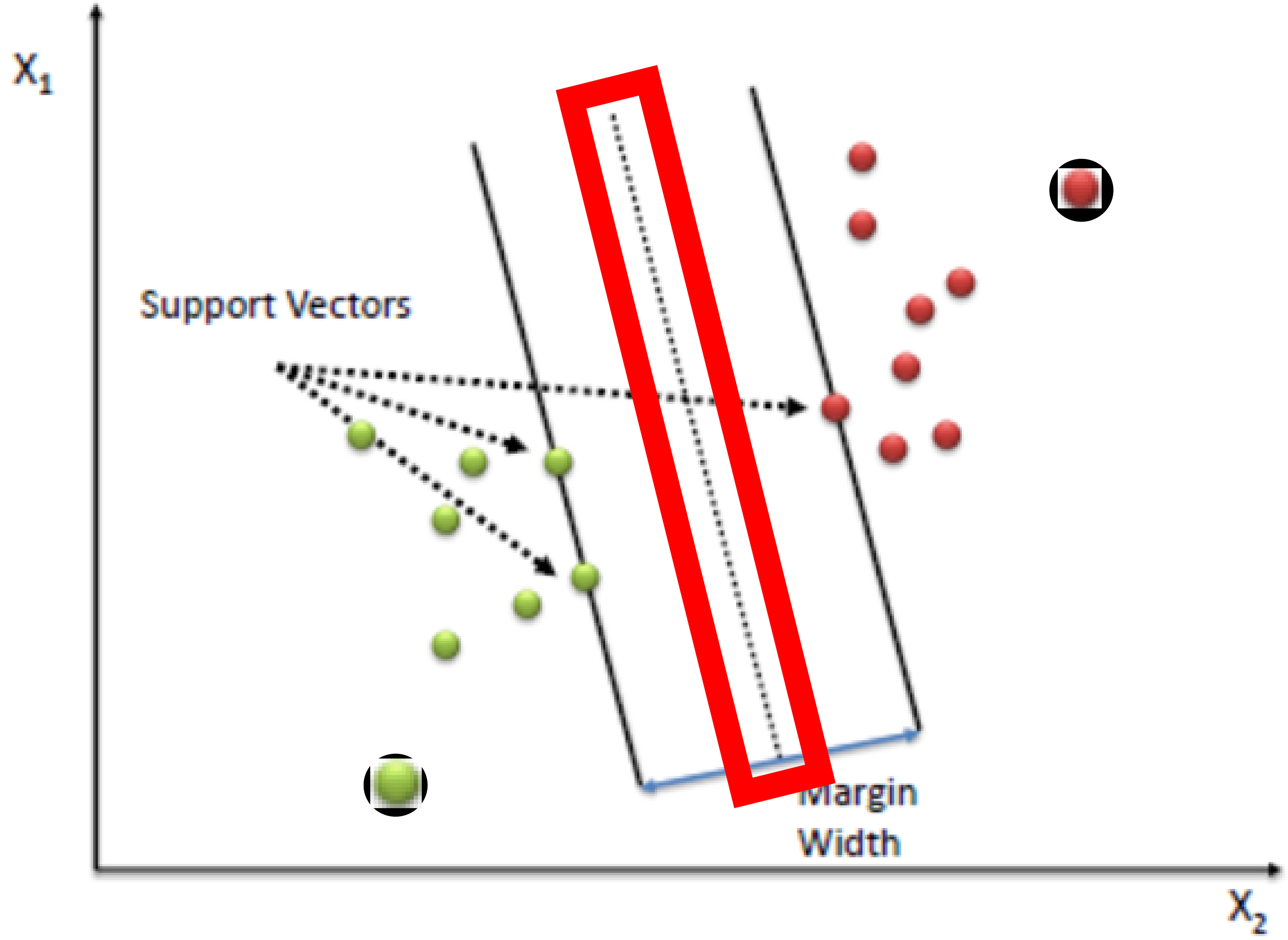


Classifier Confidence: How to Measure for Naïve Bayes?

- Conditional probability $P\ (Y|X)$ for the most probable class

Classifier Confidence: How to Measure for Support Vector Machines?

- Distance to the hyperplane: e.g.,



Classifier Confidence vs Probability

- Classifiers can make mistakes in estimating their confidence level
- External calibration procedures can address this issue (e.g., using calibration curves/reliability diagrams)

Today’s Topics

- One-vs-all multiclass classification
- Classifier confidence
- Evaluation: ROC and PR-curves
- Ensemble learning

Classification from a Classifier’s Confidence

- Observation: A threshold must be chosen to define the point at which the example belongs to a class or not

- Motivation: how to choose the threshold?
    - Default is 0.5
    - Yet, it can tuned to avoid different types of errors

Review: Confusion Matrix for Binary Classification

		Predicted class	
		P	N
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

Receiver Operating Characteristic (ROC) curve

		Predicted class	
		P	N
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

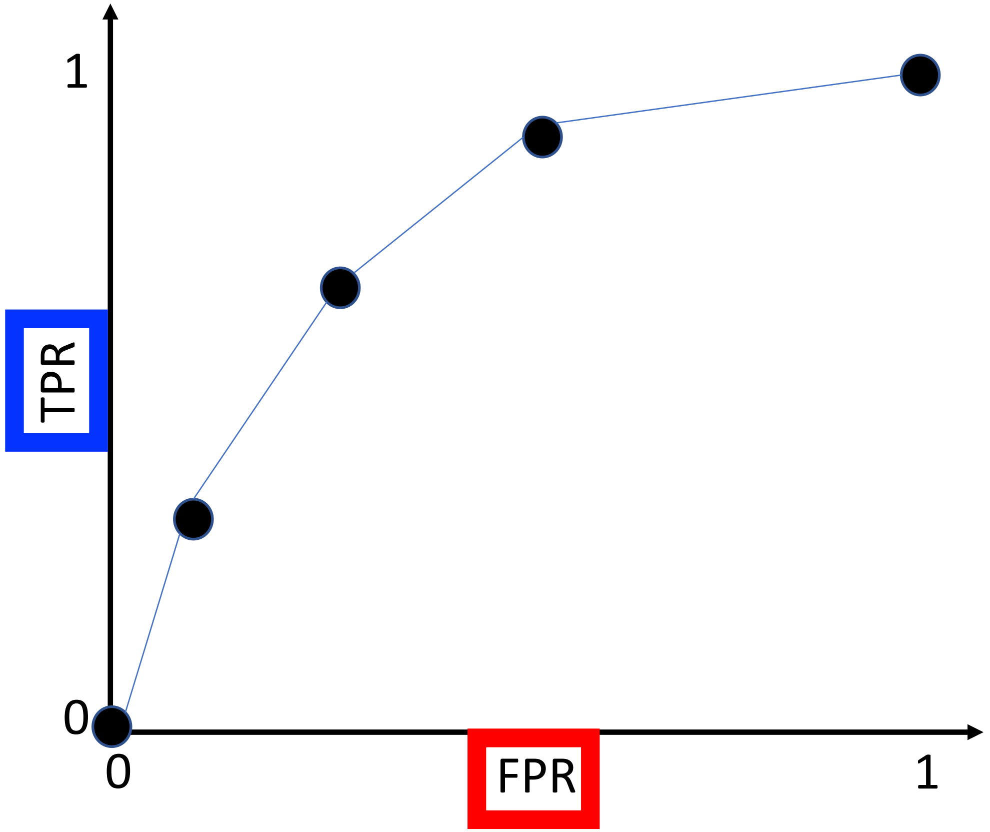
Summarizes performance based on the positive class
-A positive prediction is either correct (TP) or not (FP)

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN}$$

$$REC = TPR = \frac{TP}{P} = \frac{TP}{FN + TP}$$

Receiver Operating Characteristic (ROC) curve

To create, vary prediction threshold and compute TPR and FPR for each threshold



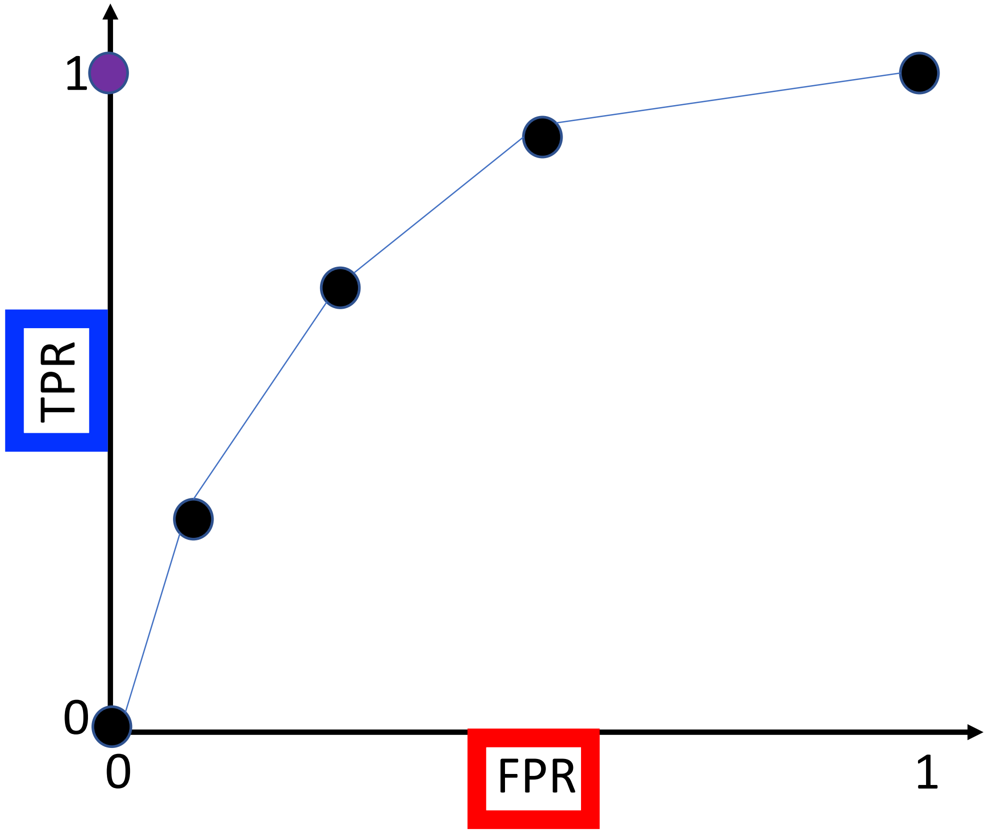
Summarizes performance based on the positive class
-A positive prediction is either correct (TP) or not (FP)

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN}$$

$$REC = TPR = \frac{TP}{P} = \frac{TP}{FN + TP}$$

Receiver Operating Characteristic (ROC) curve

What is the coordinate for a perfect predictor?



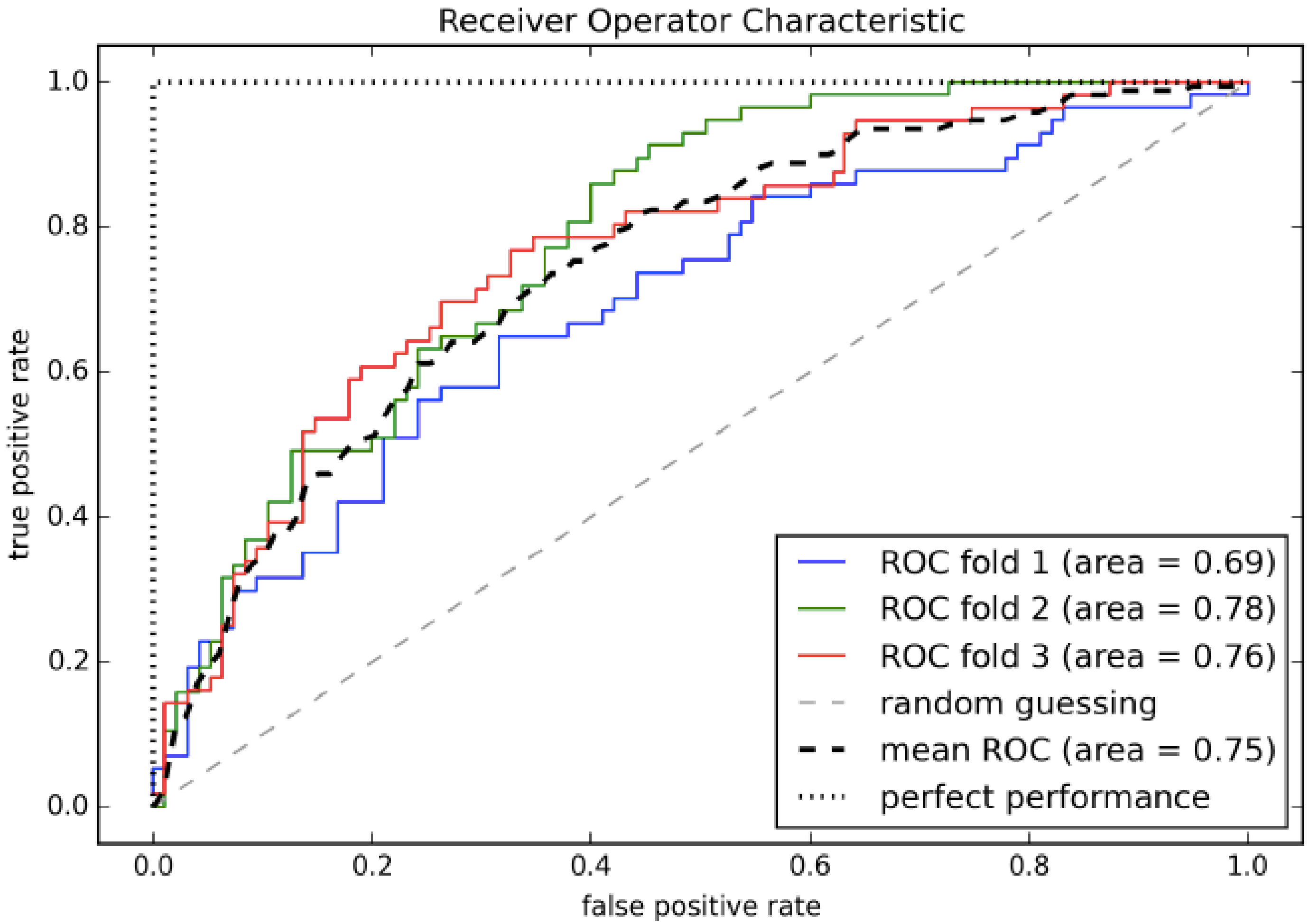
Summarizes performance based on the positive class
-A positive prediction is either correct (TP) or not (FP)

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN}$$

$$REC = TPR = \frac{TP}{P} = \frac{TP}{FN + TP}$$

ROC Curve: Area Under Curve (AUC)

Which of the first three methods performs best overall?



Summarizes performance based on the positive class

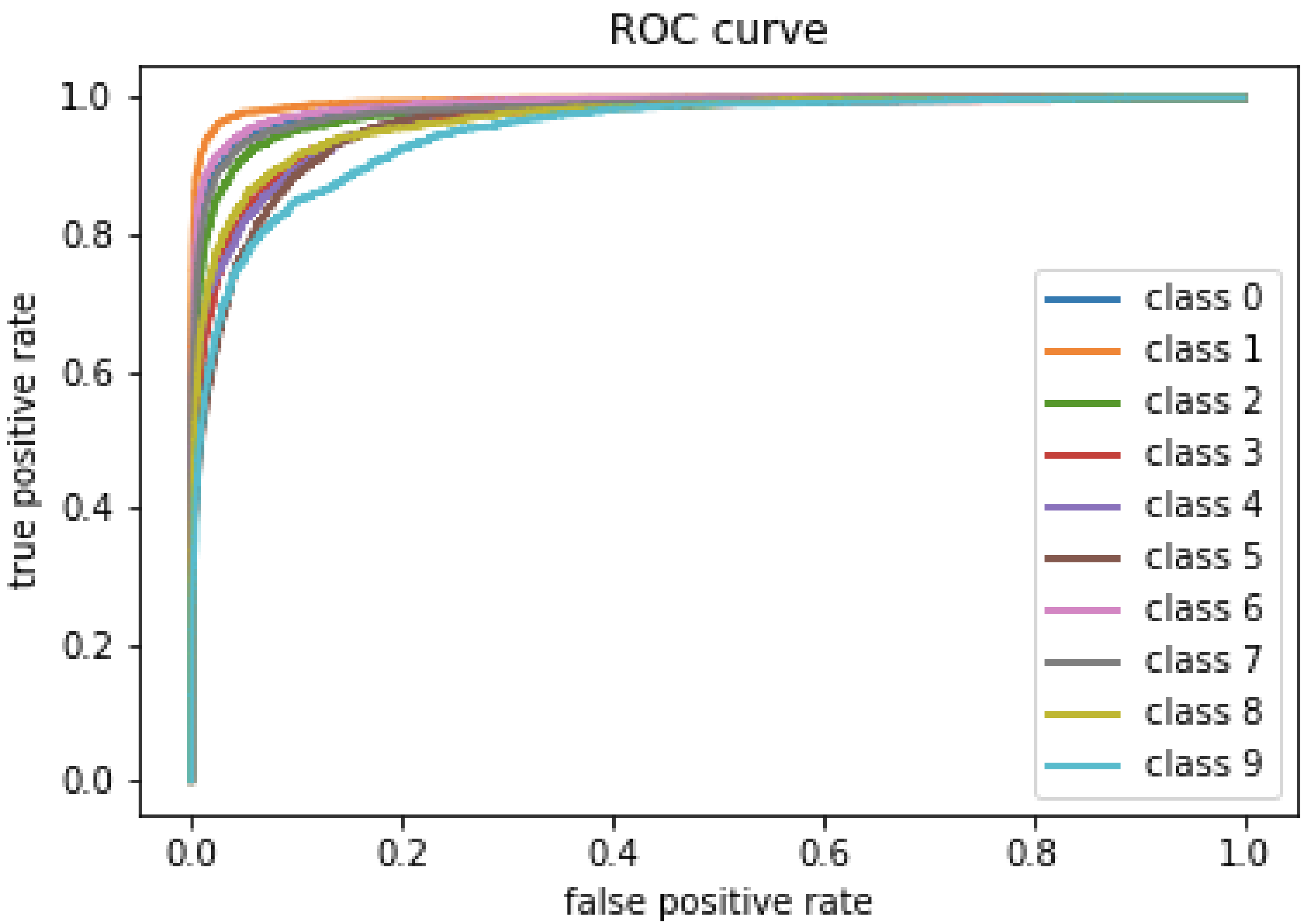
-A positive prediction is either correct (TP) or not (FP)

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN}$$

$$REC = TPR = \frac{TP}{P} = \frac{TP}{FN + TP}$$

ROC Curve: Multiclass Classification

- Plot curve per class:



Precision-Recall (PR) Curve

Predicted class

		P	N
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

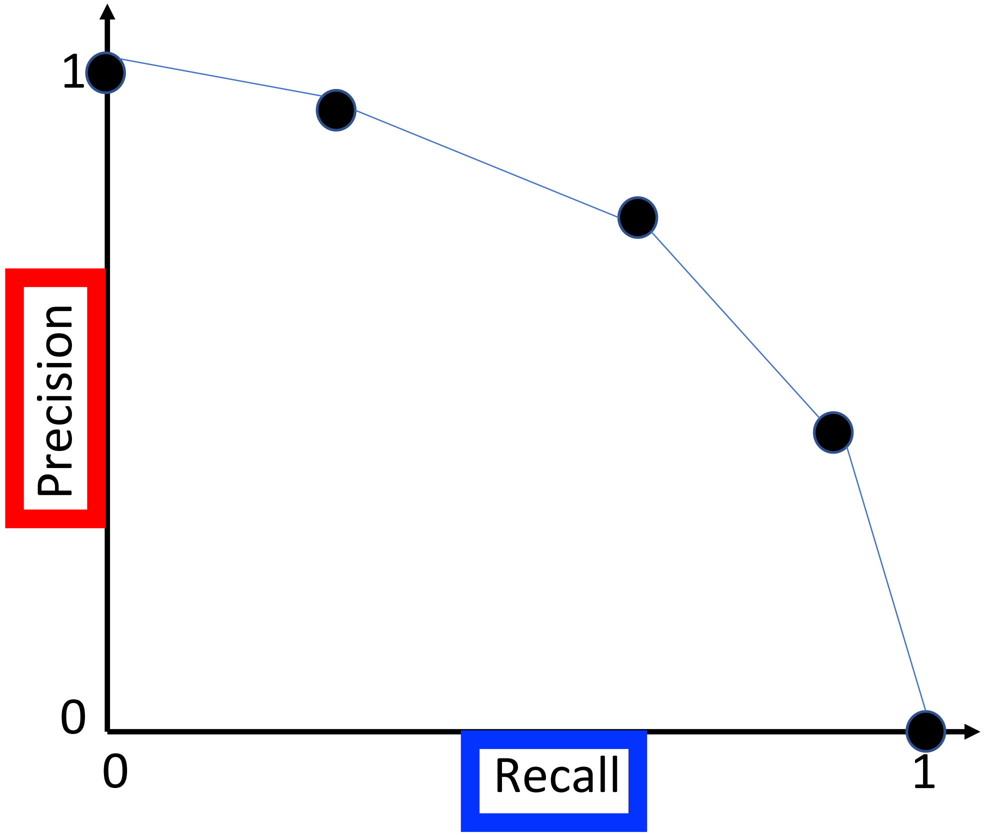
Summarizes performance based only on the positive class (ignores true negatives):

$$PRE = \frac{TP}{TP + FP}$$

$$REC = TPR = \frac{TP}{P} = \frac{TP}{FN + TP}$$

Precision-Recall (PR) Curve

To create, vary prediction threshold and
compute precision and recall for each threshold



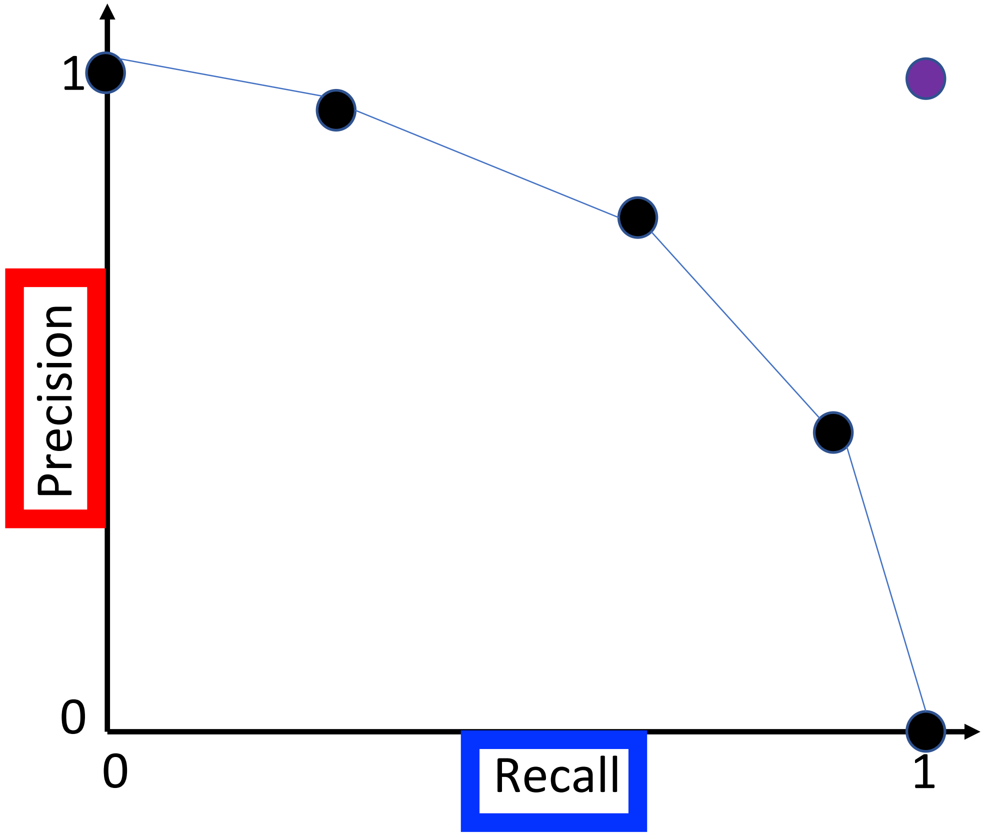
Summarizes performance based only on the positive class (ignores true negatives):

$$PRE = \frac{TP}{TP + FP}$$

$$\boxed{REC = TPR = \frac{TP}{P} = \frac{TP}{FN + TP}}$$

Precision-Recall (PR) Curve

What is the coordinate for a perfect predictor?

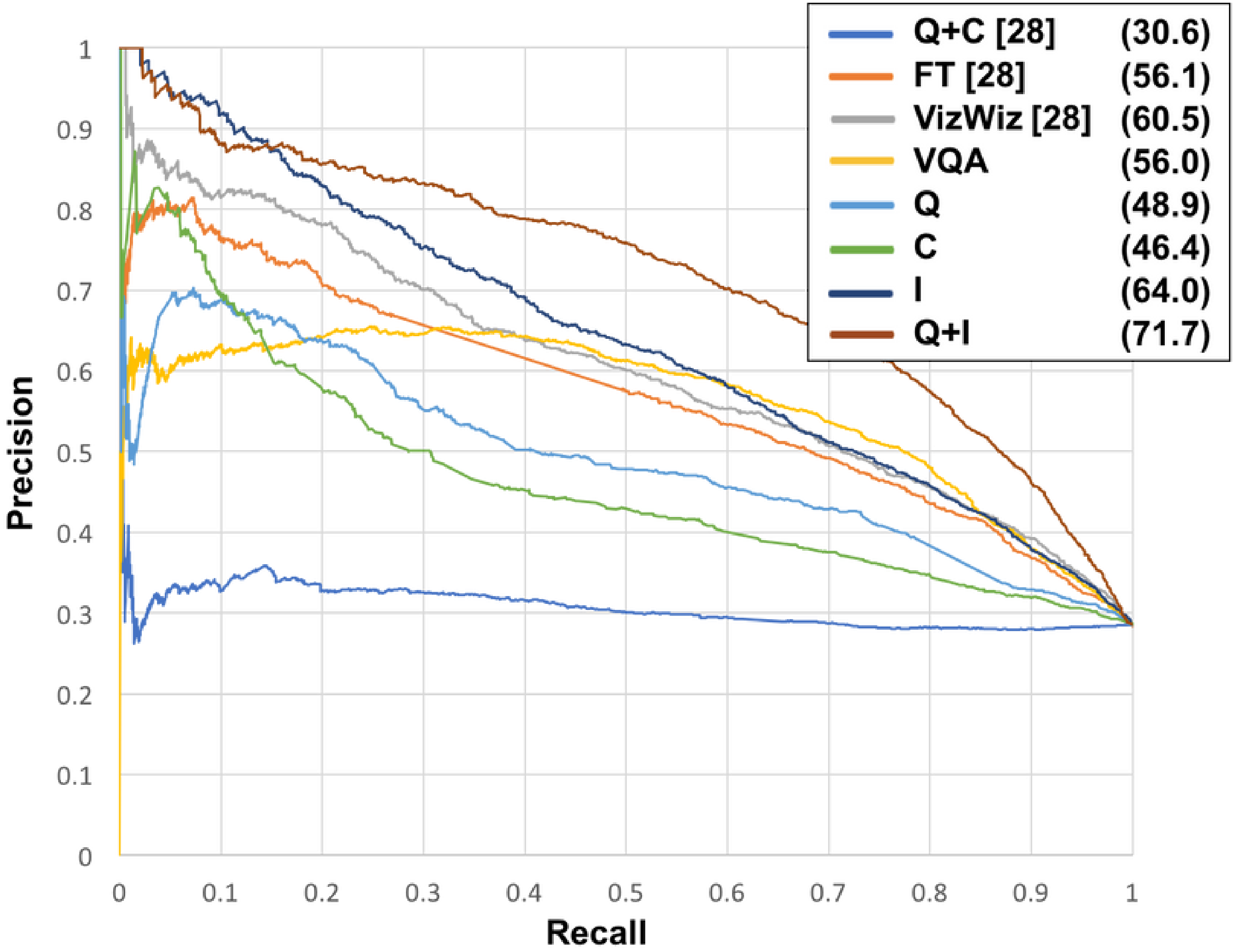


Summarizes performance based only on the positive class (ignores true negatives):

$$PRE = \frac{TP}{TP + FP}$$

$$\boxed{REC = TPR = \frac{TP}{P} = \frac{TP}{FN + TP}}$$

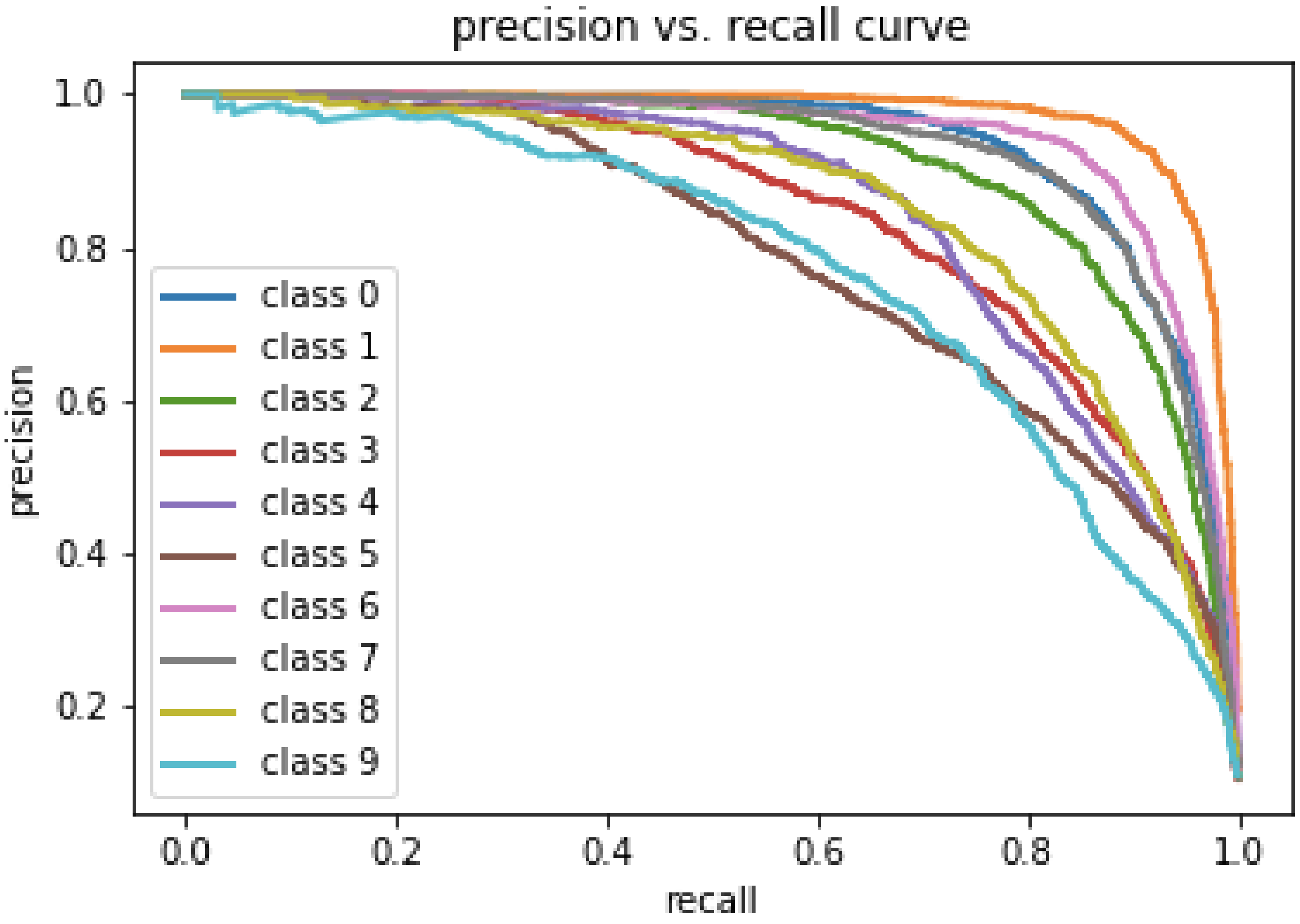
PR Curve: Area Under Curve (AUC)



- Which classifier is the best?

PR Curve: Multiclass Classification

- Plot curve per class:



Group Discussion: Evaluation Curves

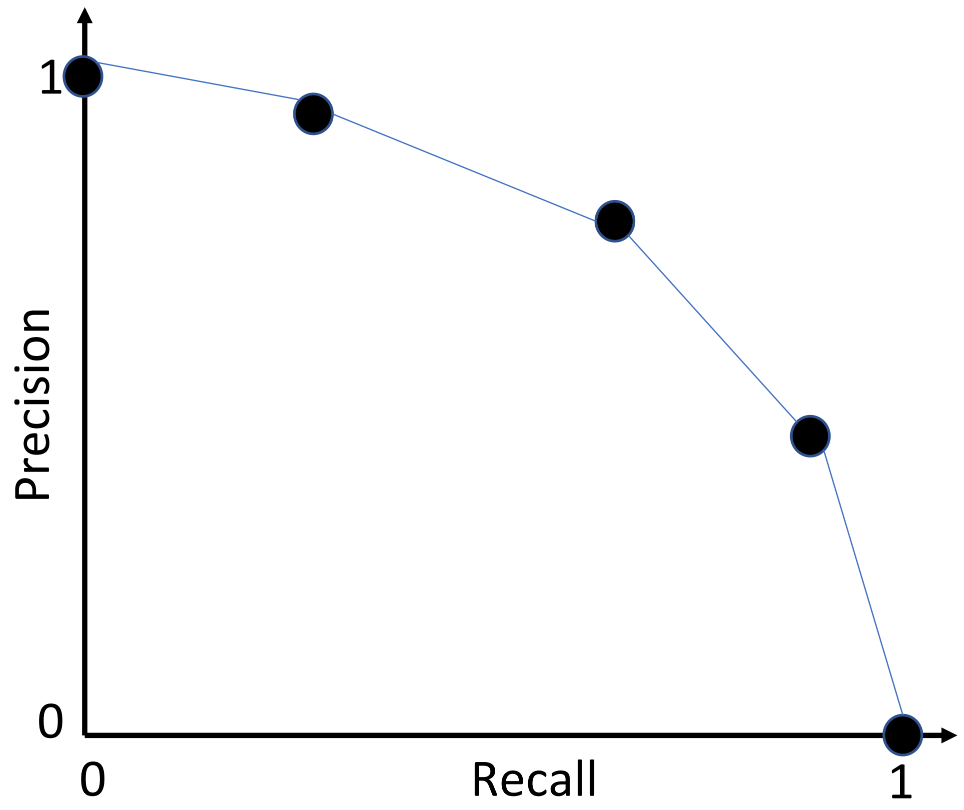
1. Assume you are building a classifier for these applications:
   - Detecting offensive content online
   - Medical diagnoses
   - Detecting shoplifters
   - Deciding whether a person is guilty of a crime

What classifier threshold would you choose for each application and why?

2. When would you choose to evaluate with a PR curve versus a ROC curve?

3. What is the area under the ROC and PR curves for a perfect classifier?

Assume the following thresholds were used to create the curve: 0, 0.25, 0.5, 0.75, 1.



Today’s Topics

- One-vs-all multiclass classification
- Classifier confidence
- Evaluation: ROC and PR-curves
- Ensemble learning

Idea: How Many Predictors to Use?

More than 1: Ensemble

Why Choose Ensemble Instead of an Algorithm?

- Reduces probability for making a wrong prediction, assuming:
  - Classifiers are independent (not true in practice!)
- Suppose:
  - n classifiers for binary classification task
  - Each classifier has same error rate ε
  - Probability mass function indicates the probability of error from an ensemble:

Number of classifiers

Classifier error rate

$$P(y \geq k) = \sum_{k}^{\boxed{n}} \boxed{\binom{n}{k}} \varepsilon^{k} (1 - \boxed{\varepsilon})^{n-k} = \boxed{\varepsilon_{ensemble}}$$

Error probability

ways to choose *k* subsets from set of size *n*

- e.g., n = 11, ε = 0.25; k = 6: probability of error is ~0.034 which is much lower than probability of error from a single algorithm (0.25)

Why Choose Ensemble Instead of an Algorithm?

- Reduces probability for making a wrong prediction, assuming:
  - Classifiers are independent (not true in practice!)
- Suppose:
  - n classifiers for binary classification task
  - Each classifier has same error rate ε
  - Probability mass function indicates the probability of error from an ensemble:

How to Get Diverse Classifiers?

$$P(y \ge k) = \sum_{k}^{n} \binom{n}{k} \varepsilon^{k} (1-\varepsilon)^{n-k} = \varepsilon_{ensemble}$$

- e.g., n = 11, ε = 0.25; k = 6: probability of error is ~0.034 which is much lower than probability of error from a single algorithm (0.25)

Why Choose Ensemble Instead of an Algorithm?

- Reduces probability for making a wrong prediction, assuming:
  - Classifiers are independent (not true in practice!)
- Suppose:
  - n classifiers for binary classification task
  - Each classifier has same error rate
  - Probability mass function indicates the probability of error from an ensemble:

1. Use different algorithms

2. Use different features

$$P(y \geq k) = \sum_{k}^{n} \binom{n}{k} \varepsilon^{k} (1-\varepsilon)^{n-k} = \varepsilon_{ensemble}$$

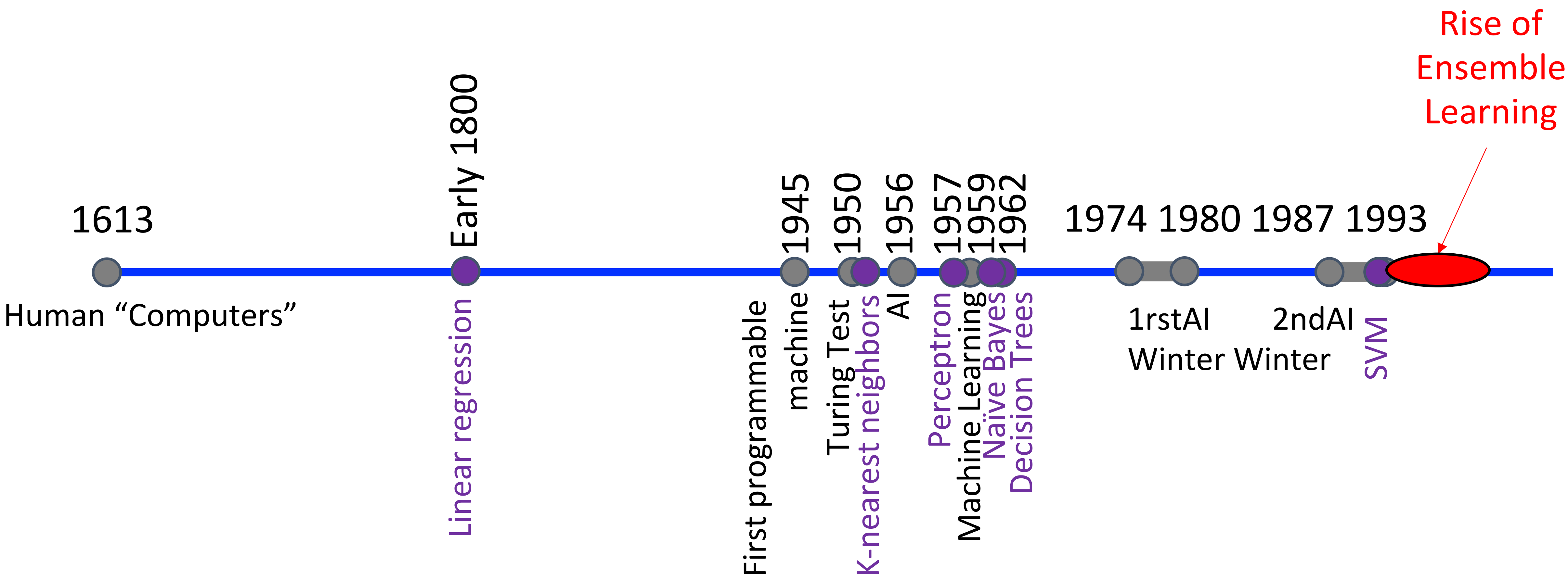
2. Use different training data

- e.g., n = 11, ε = 0.25; k = 6: probability of error is ~0.034 which is much lower than probability of error from a single algorithm (0.25)

How to Predict with an Ensemble?

- Majority Voting
  - Return most popular prediction from multiple prediction algorithms
- Bootstrap Aggregation, aka Bagging
  - Resample data to train algorithm on different random subsets
- Boosting
  - Reweight data to train algorithms to specialize on different “hard” examples
- Stacking
  - Train a model that learns how to aggregate classifiers’ predictions

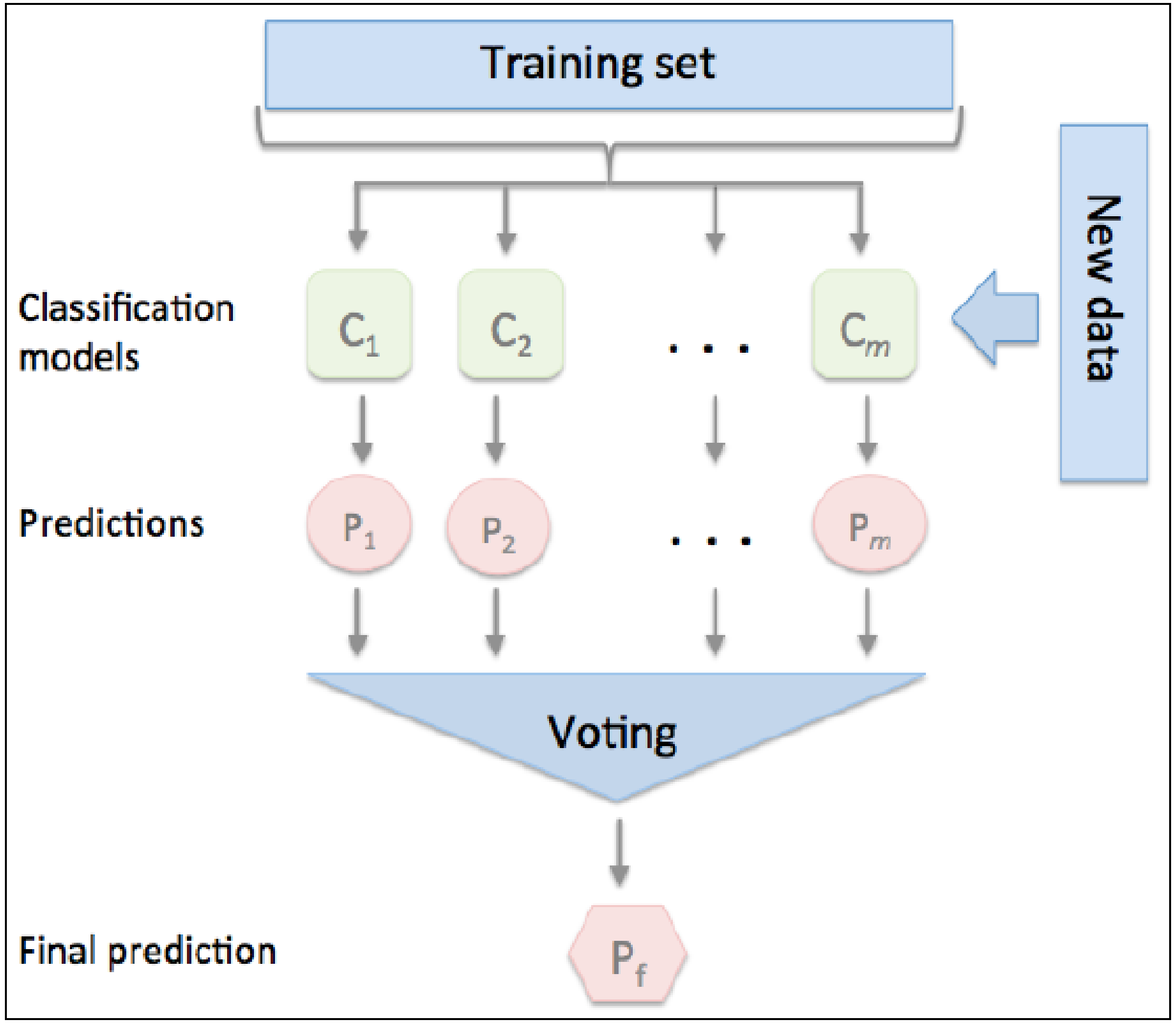
Historical Context of ML Models



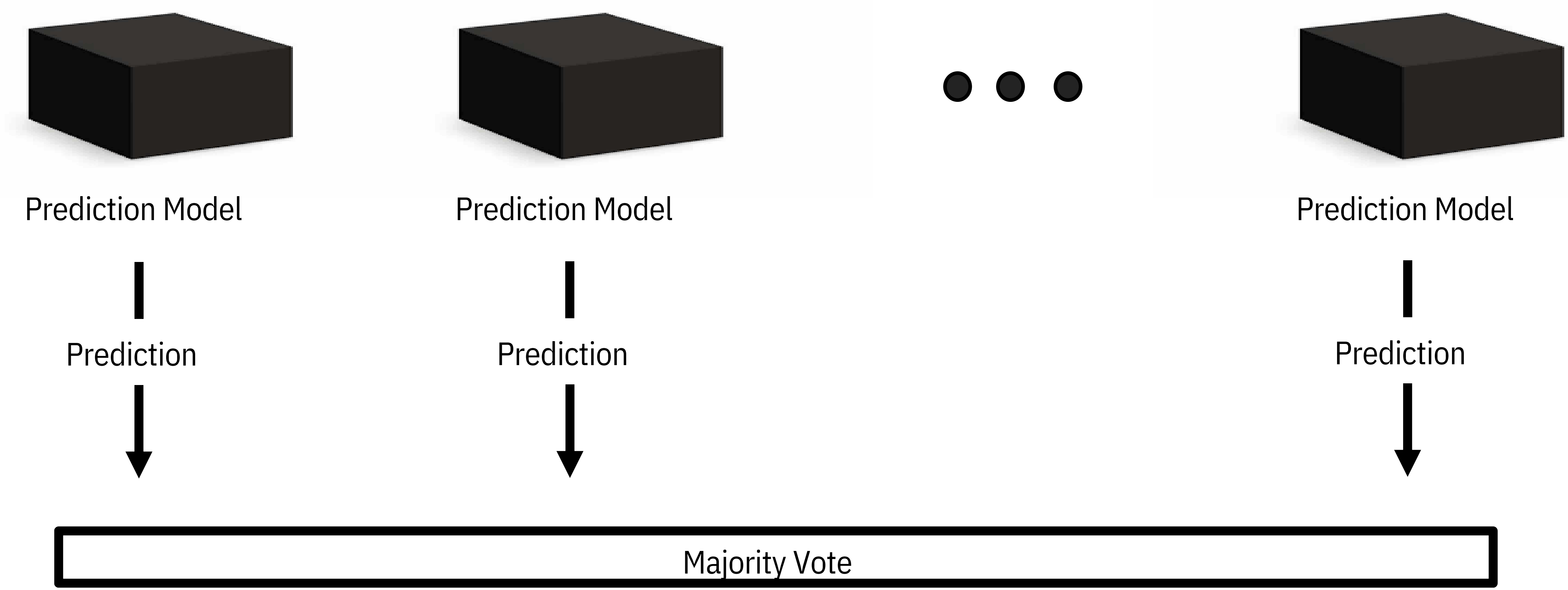
How to Predict with an Ensemble of Algorithms?

- Majority Voting
  - Return most popular prediction from multiple prediction algorithms
- Bootstrap Aggregation, aka Bagging
  - Train algorithm repeatedly on different random subsets of the training set
- Boosting
  - Train algorithms that each specialize on different “hard” training examples
- Stacking
  - Train a model that learns how to aggregate classifiers’ predictions

Majority Voting

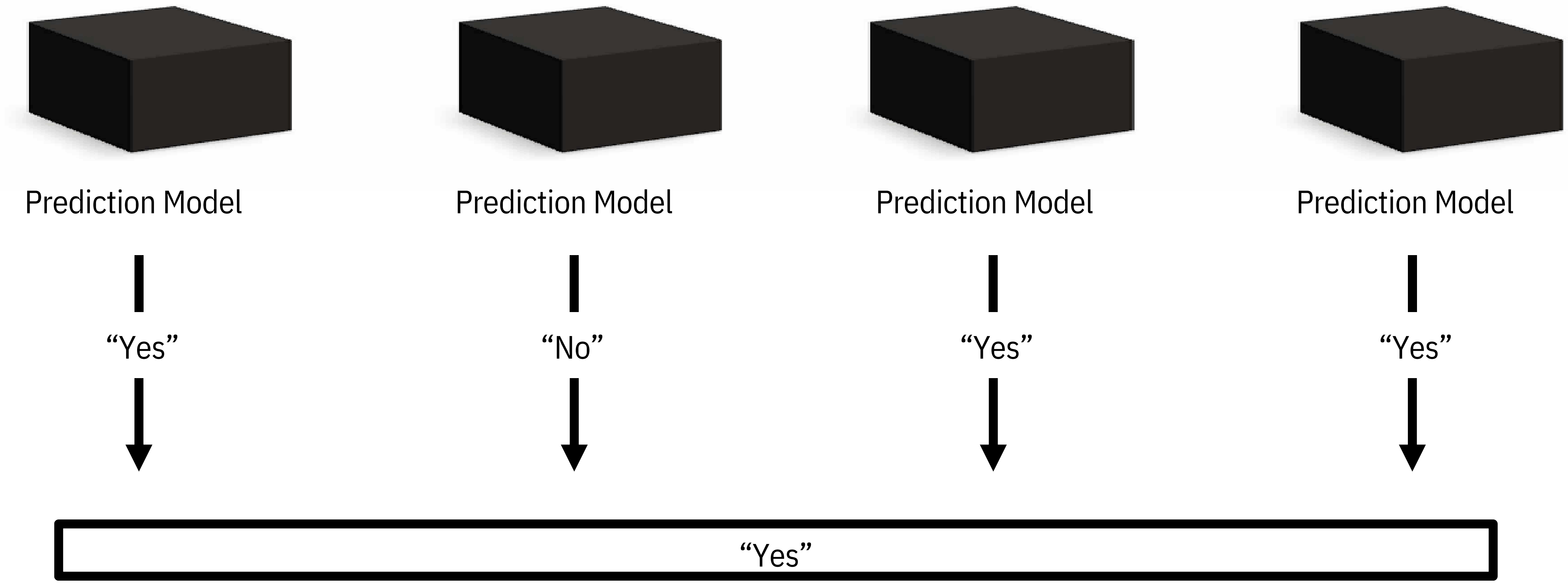


Majority Voting

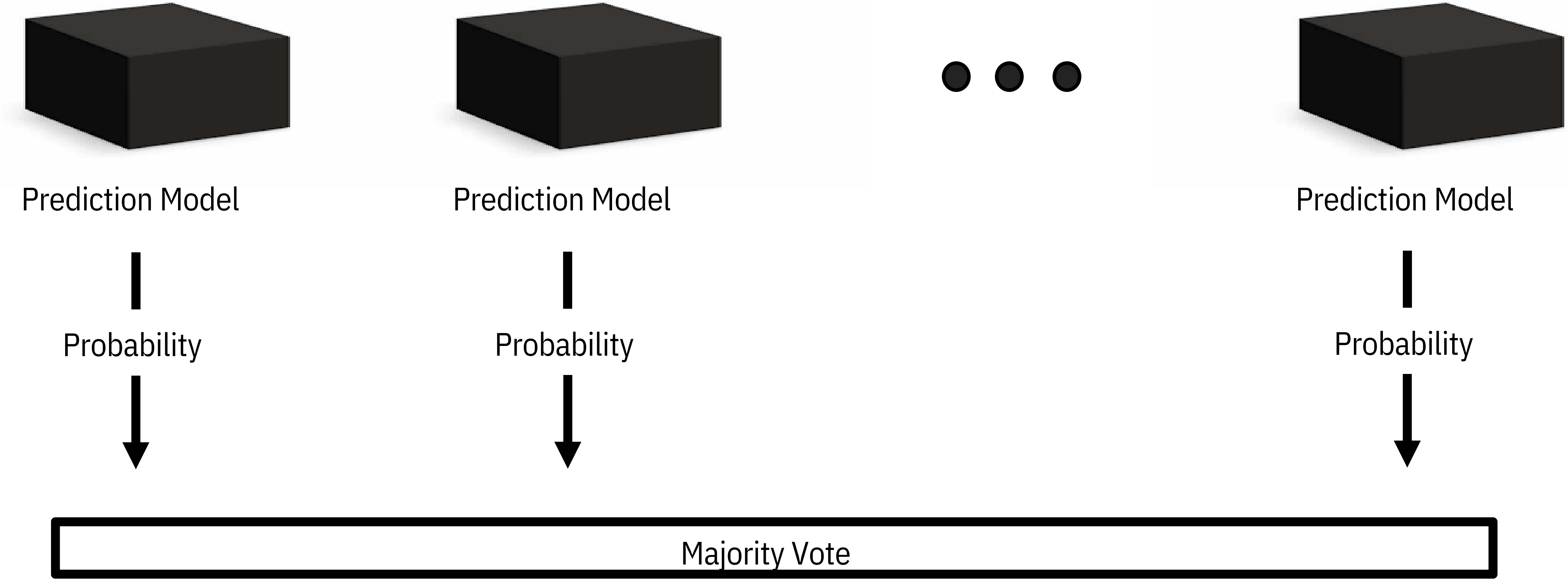


Majority Voting: Binary Task

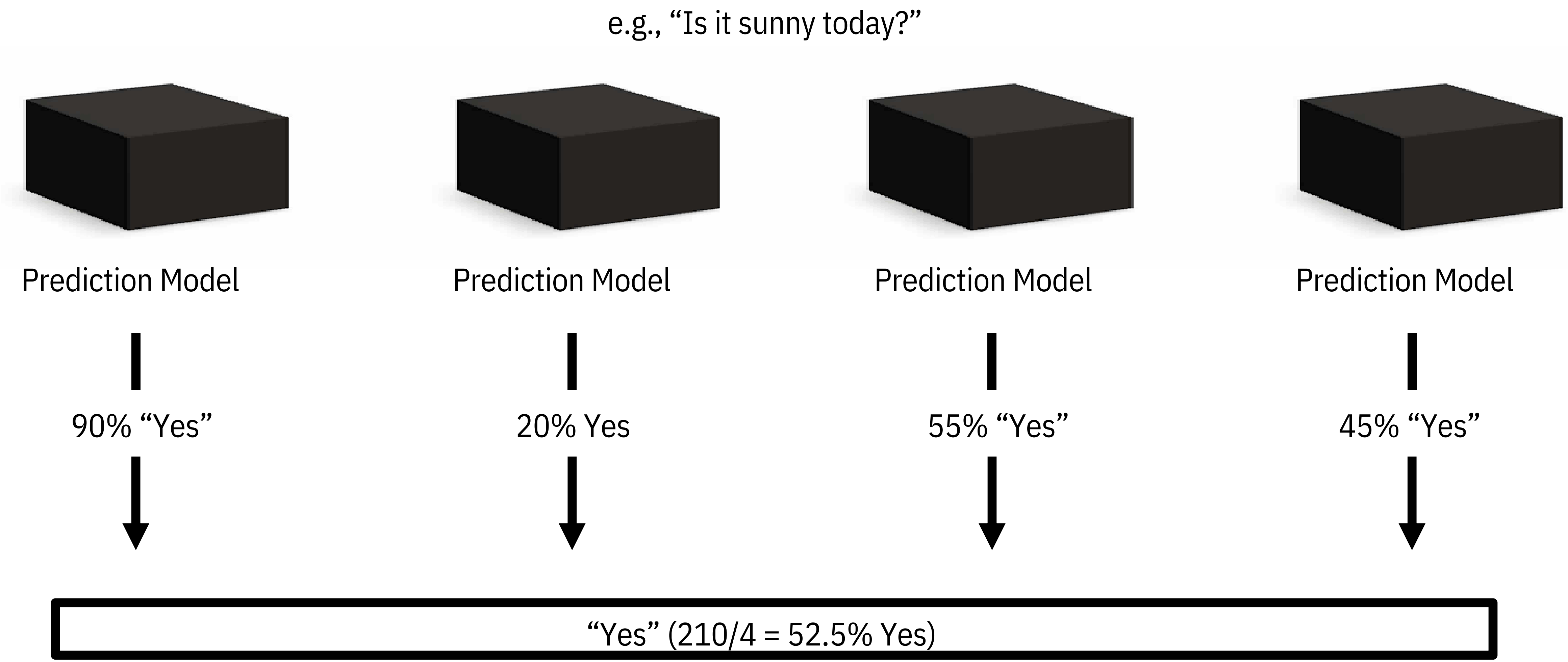
e.g., “Is it sunny today?”



Majority Voting: “Soft” (not “Hard”)

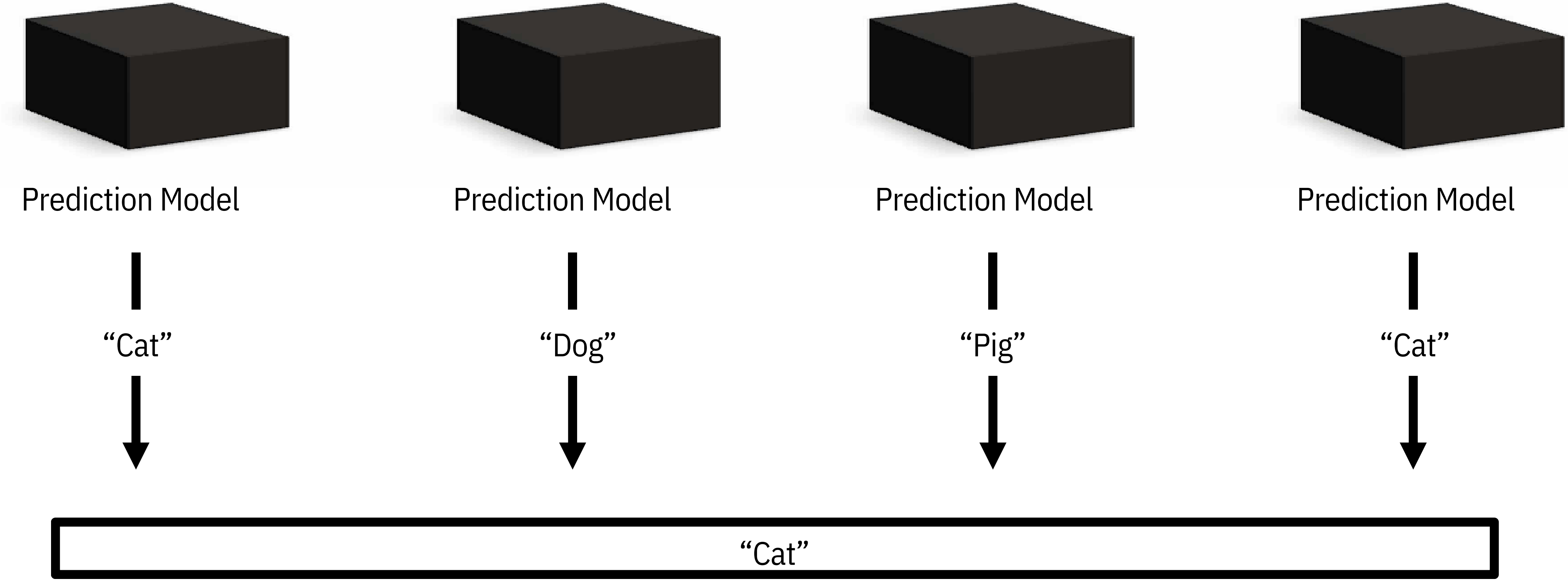


Majority Voting: Soft Voting on Binary Task



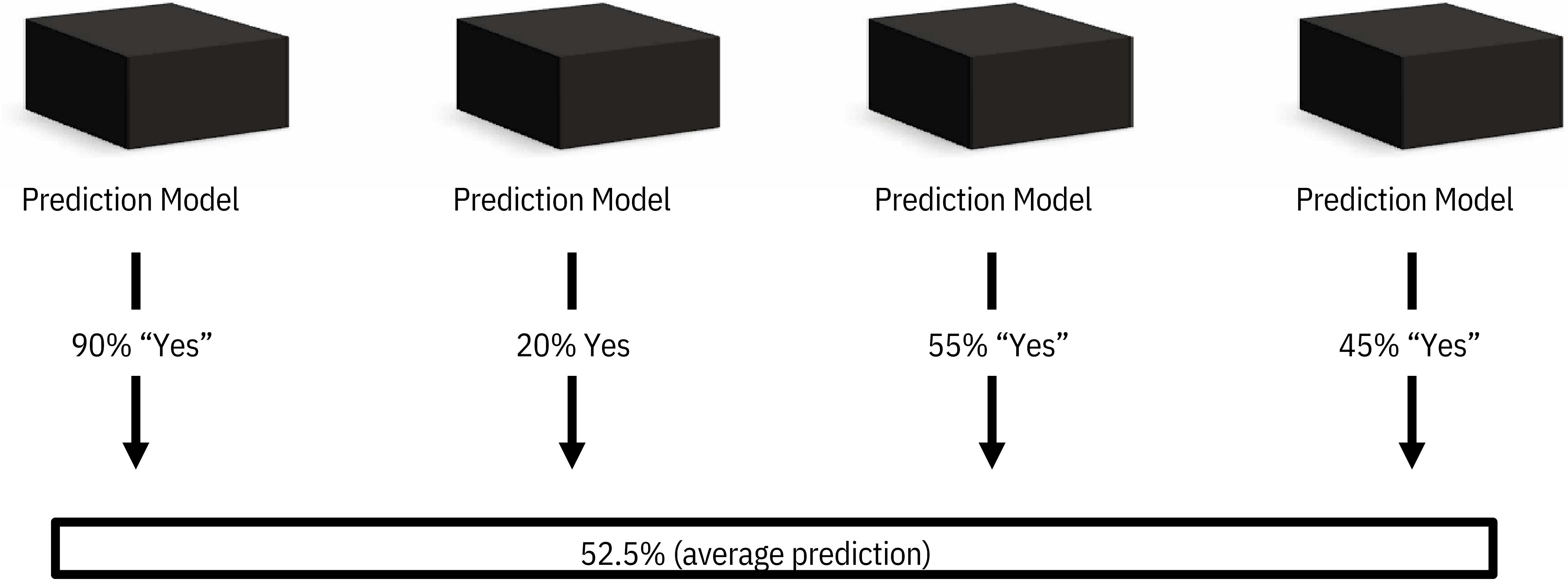
Plurality Voting: Non-Binary Task

e.g., “What object is in the image?”

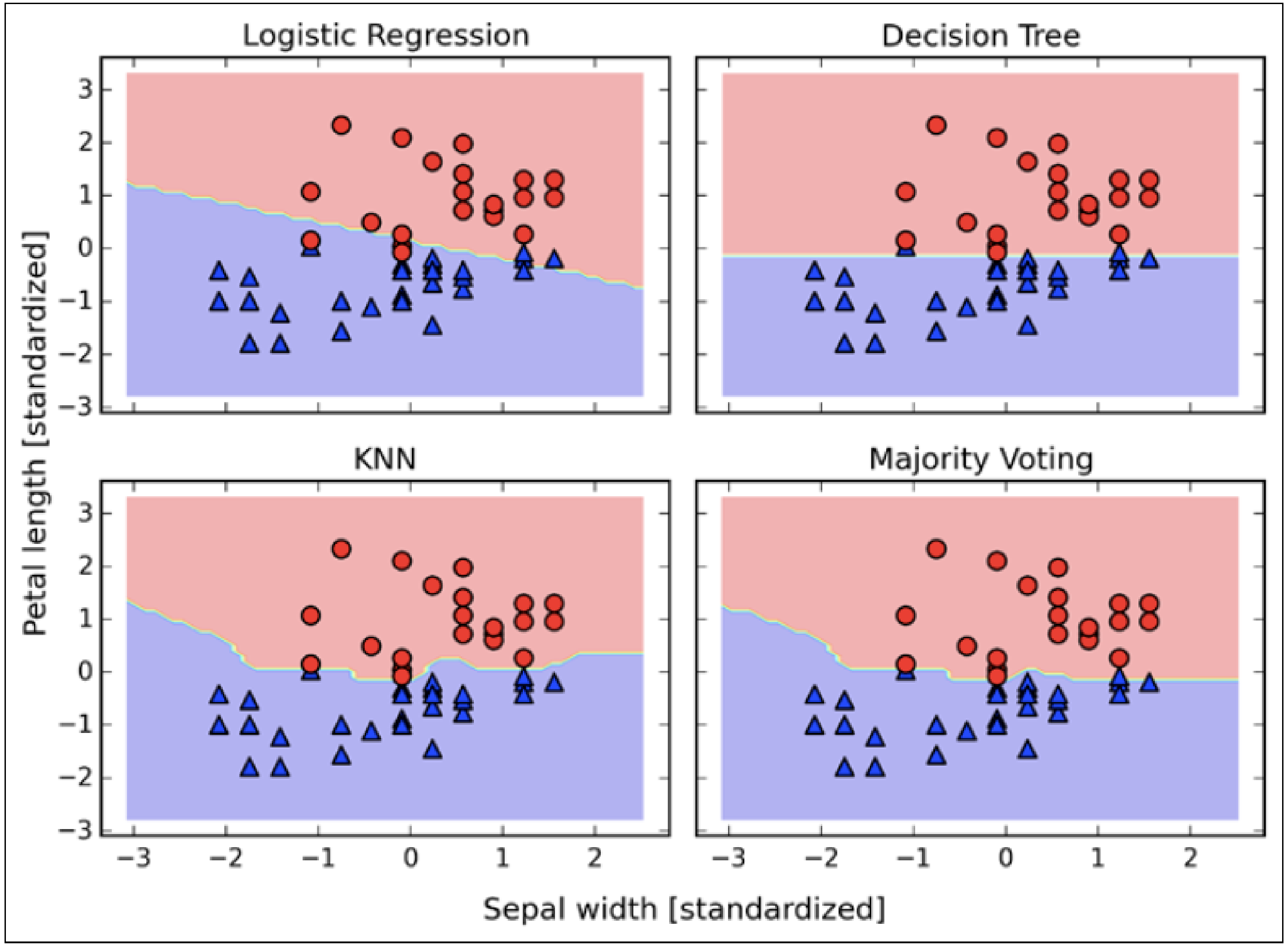


Majority Voting: Regression

e.g., “Is it sunny today?”



Majority Voting: Example of Decision Boundary



How to Predict with an Ensemble of Algorithms?

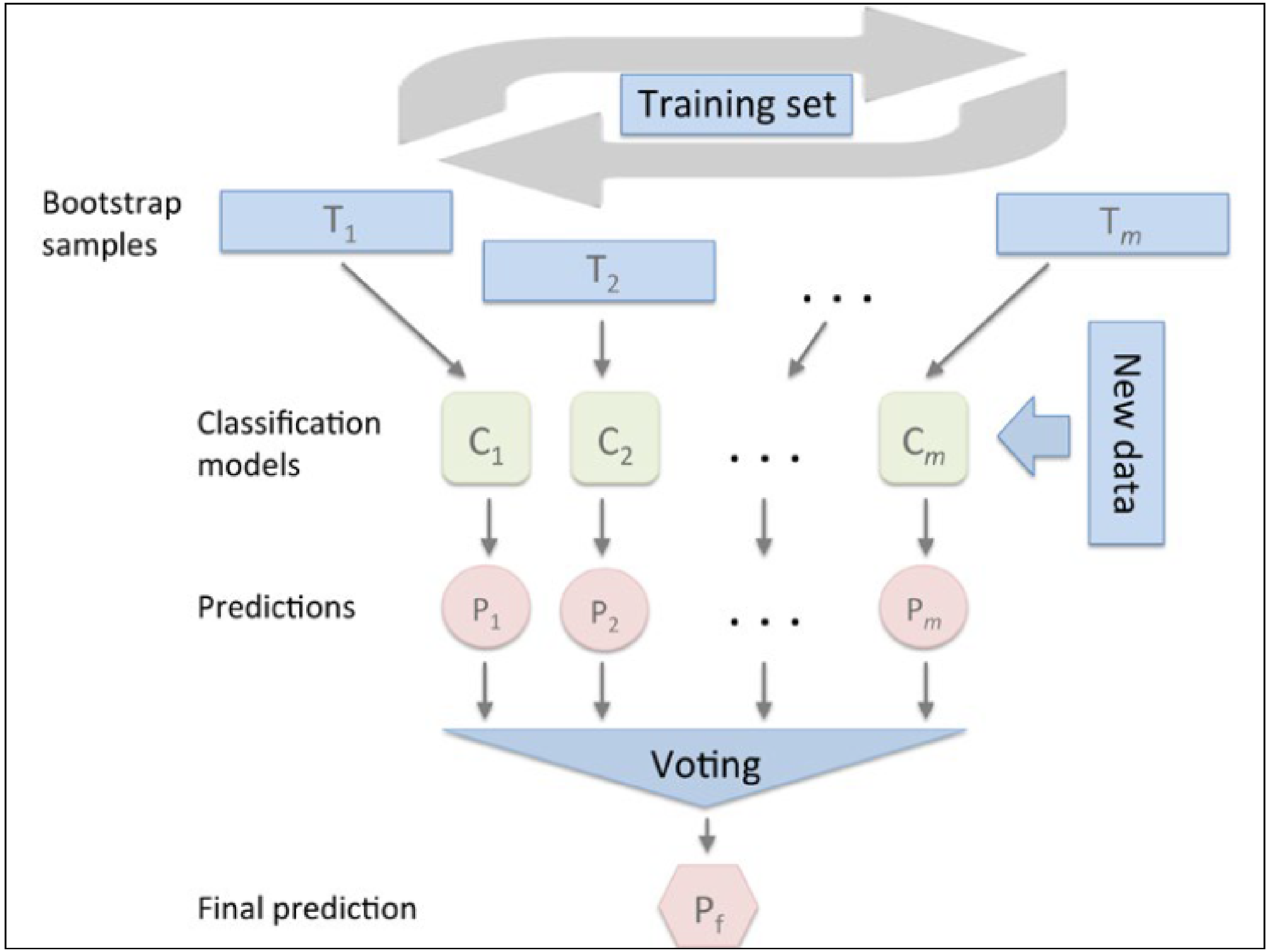
- Majority Voting
  - Return most popular prediction from multiple prediction algorithms

- Bootstrap Aggregation, aka Bagging
  - Train algorithm repeatedly on different random subsets of the training set

- Boosting
  - Train algorithms that each specialize on different “hard” training examples

- Stacking
  - Train a model that learns how to aggregate classifiers’ predictions

Bagging


Training set
Bootstrap samples
T_1
T_2
T_m
Classification models
C_1
C_2
C_m
New data
Predictions
P_1
P_2
P_m
Voting
Final prediction
P_f

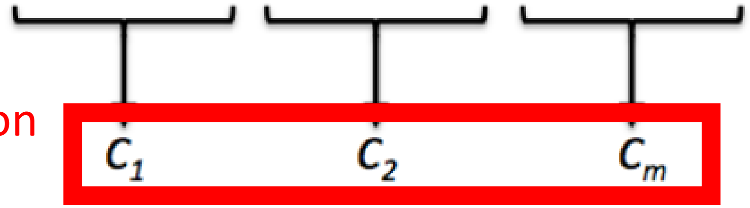
Bagging: Training

- Build ensemble from "bootstrap samples" drawn with
- replaceme

Duplicate data can occur for training

Some examples missing from training data; e.g., round 1

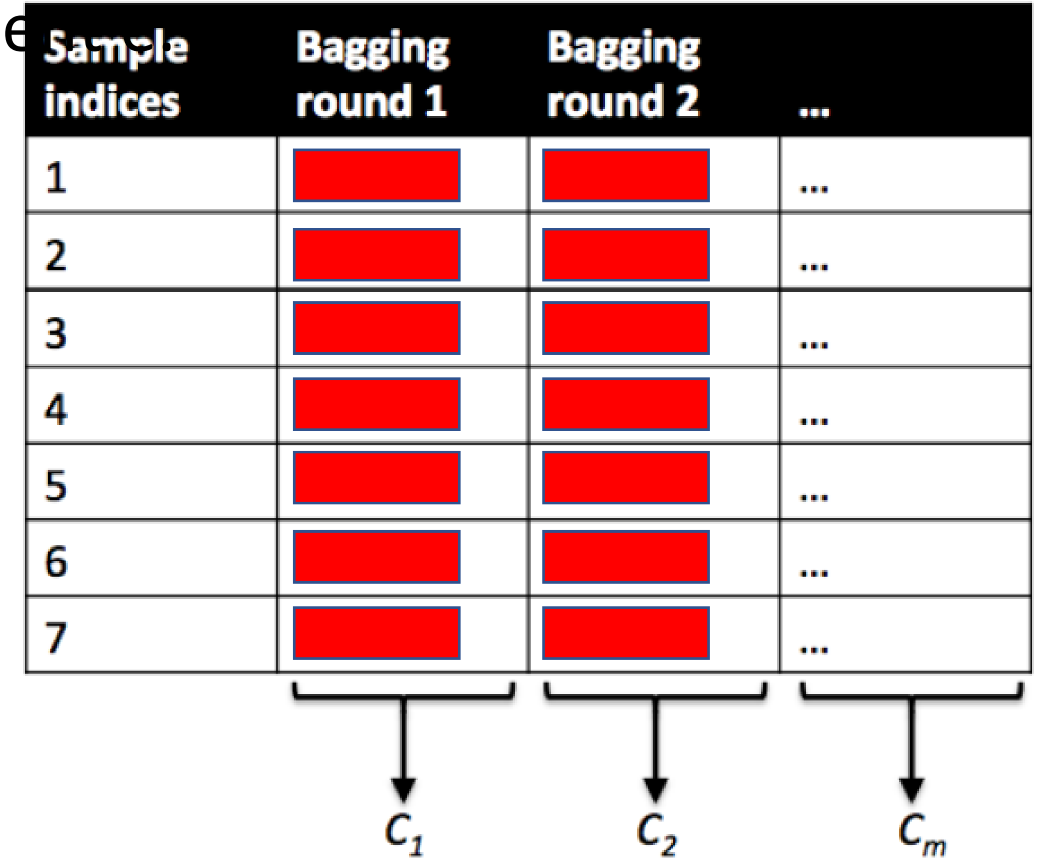
Sample indices	Bagging round 1	Bagging round 2	...
1	2	7	...
2	2	3	...
3	1	2	...
4	3	1	...
5	7	1	...
6	2	7	...
7	4	7	...



Each classifier trained on different subset of data

Bagging: Training

- Build ensemble from “bootstrap samples” drawn with
- replaceme



Class Demo:
-Pick a number
from the bag

Bagging: Predicting

Prediction Model Prediction Model Prediction Model Prediction Model

- Predict as done for “majority voting”
  - e.g., “hard” voting
  - e.g., “soft” voting
  - e.g., averaging values for regression

Bagging: Random Forest

- Build ensemble from “bootstrap samples” drawn with
- replaceme

Sample indices	Bagging round 1	Bagging round 2	...
1	2	7	...
2	2	3	...
3	1	2	...
4	3	1	...
5	7	1	...
6	2	7	...
7	4	7	...

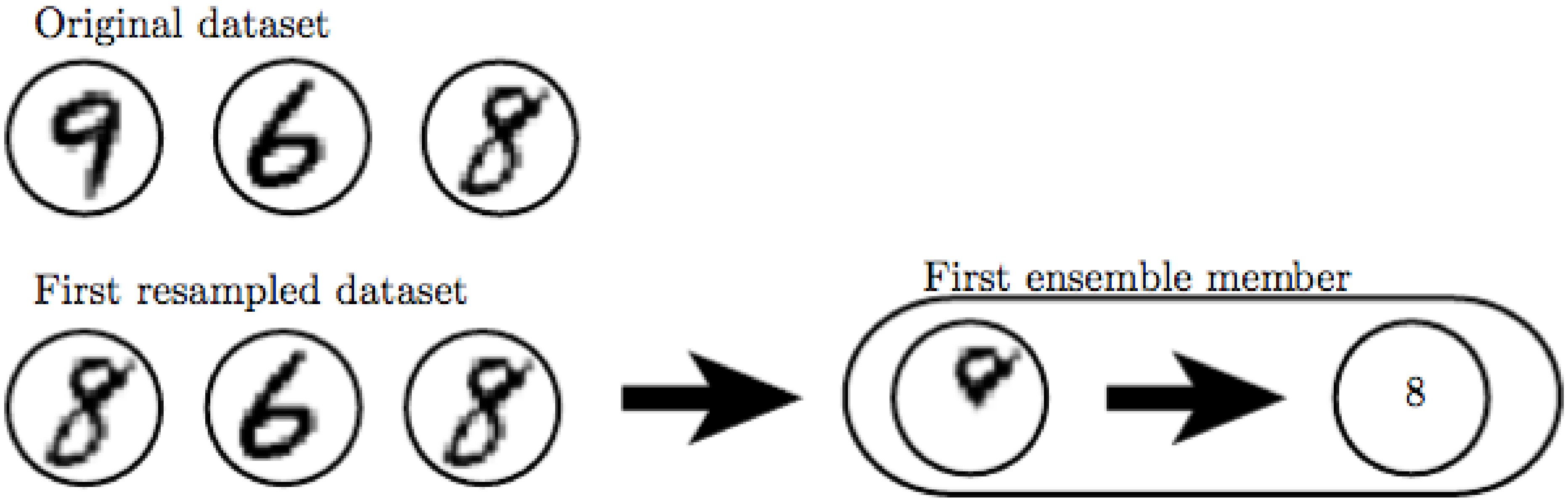
C_1 C_2 C_m

Fit decision trees by also selecting random feature subsets

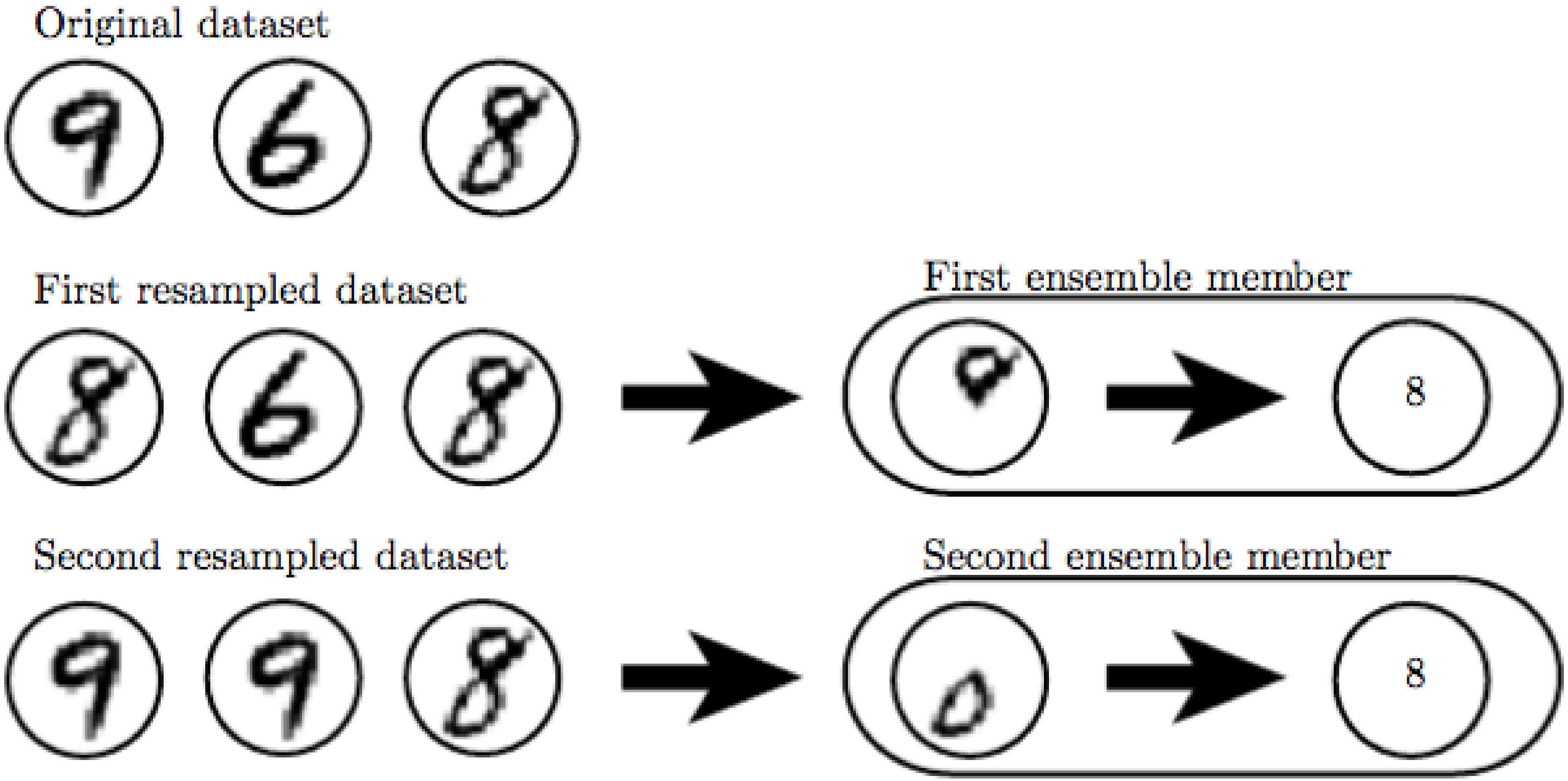
Bagging: Intuition (Train an 8 detector)

Original dataset

Bagging: Intuition (Train an 8 detector)



Bagging: Intuition (Train an 8 detector)



How to Predict with an Ensemble of Algorithms?

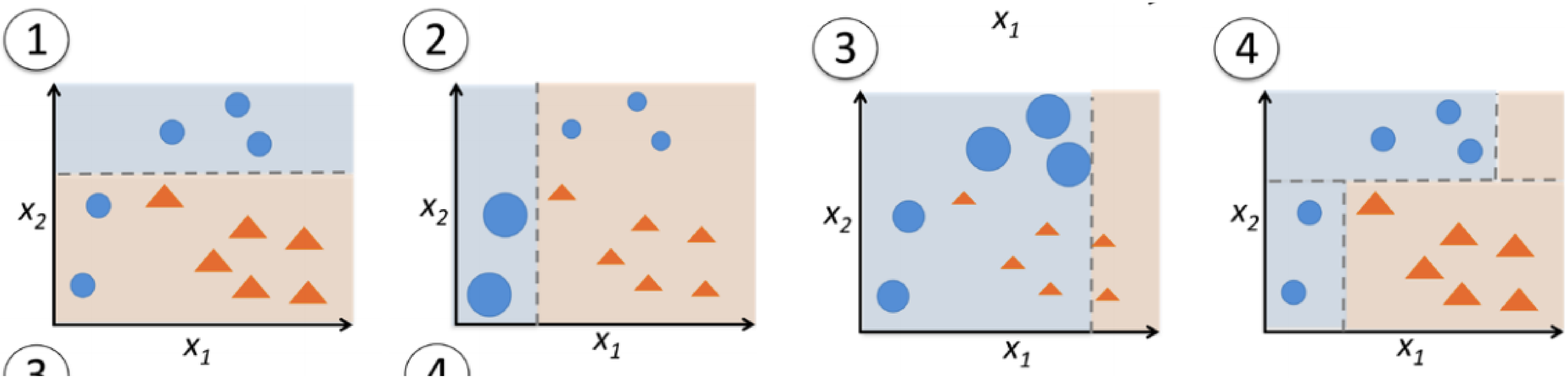
- Majority Voting
  - Return most popular prediction from multiple prediction algorithms
- Bootstrap Aggregation, aka Bagging
  - Train algorithm repeatedly on different random subsets of the training set
- Boosting
  - Train algorithms that each specialize on different “hard” training examples
- Stacking
  - Train a model that learns how to aggregate classifiers’ predictions

Boosting

- Key idea: sequentially train predictors that each try to correctly predict examples that were hard for previous predictors

- Original Algorithm:
  - Train classifier 1: use random subset of examples without
  - replacement Train classifier 2: use a second random subset of examples without
  - replacement and add 50% of examples misclassified by classifier 1
  - Train classifier 3: use examples that classifiers 1 and 2 disagree on Predict using majority vote from 3 classifiers

Boosting –Adaboost(Adaptive Boosting)



Assign equal weights to all examples

- Assign larger weights to previous misclassifications
- Assign smaller weights to previous correct classifications

- Assign larger weights to training samples C1and C2 disagree on
- Assign smaller weights to previous correct classifications

Predict with weighted majority vote

Freund and Schapire, Experiments with a New Boosting Algorithm, 1996.

Raschkaand Mirjalili; Python Machine Learning

Boosting –Adaboost(Adaptive Boosting)

e.g., 1d dataset

Sample indices	x	y	Weights	$\hat{y}(x <= 3.0)$?	Correct?	Updated weights
1	1.0	1	0.1	1	Yes	0.072
2	2.0	1	0.1	1	Yes	0.072
3	3.0	1	0.1	1	Yes	0.072
4	4.0	-1	0.1	-1	Yes	0.072
5	5.0	-1	0.1	-1	Yes	0.072
6	6.0	-1	0.1	-1	Yes	0.072
7	7.0	1	0.1	-1	No	0.167
8	8.0	1	0.1	-1	No	0.167
9	9.0	1	0.1	-1	No	0.167
10	10.0	-1	0.1	-1	Yes	0.072

Round 2:
update weights

Round 1: training data, weights, predictions

Boosting –Adaboost(Adaptive Boosting)

e.g., 1d dataset

1. Compute error rate (sum misclassified examples' weights):

$$\varepsilon = 0.1\times 0+0.1\times 0+0.1\times 0+0.1\times 0+0.1\times 0+0.1\times 0+0.1\times 1+0.1\times 1$$
$$+0.1\times 1+0.1\times 0=\frac{3}{10}=0.3$$

2 Compute coefficient used to update weights and make majority vote prediction:

$$\alpha_j = 0.5\log\left(\frac{1-\varepsilon}{\varepsilon}\right)\approx 0.424$$

- Update weight vector:

3 $$w := w\times \exp\left(-\alpha_j \times \hat{y}\times y\right)$$

- Correct predictions will decrease weight and vice versa

$$0.1\times \exp\left(-0.424\times 1\times 1\right)\approx 0.065 \quad 0.1\times \exp\left(-0.424\times(-1)\times(1)\right)\approx 0.153$$

4. Normalize weights to sum to 1: $$w := \frac{w}{\sum_i w_i}$$

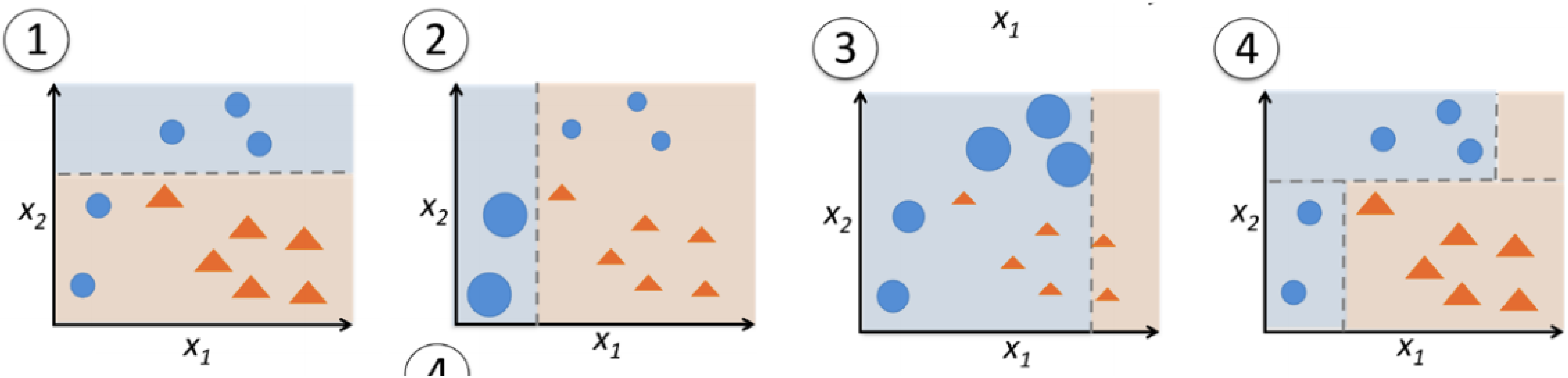
$$\sum_i w_i = 7\times 0.065+3\times 0.153=0.914$$

Correct?	Updated weights
Yes	0.072
Yes	0.072
Yes	0.072
Yes	0.072
Yes	0.072
Yes	0.072
No	0.167
No	0.167
No	0.167
Yes	0.072

$0.065 / 0.914$

$0.153 / 0.914$

Boosting –Adaboost(Adaptive Boosting)



To predict, use α calculated for each classifier as its weight when voting with all trained classifiers.

Idea: value the prediction of each classifier based on the accuracies they had on the training dataset.

How to Predict with an Ensemble of Algorithms?

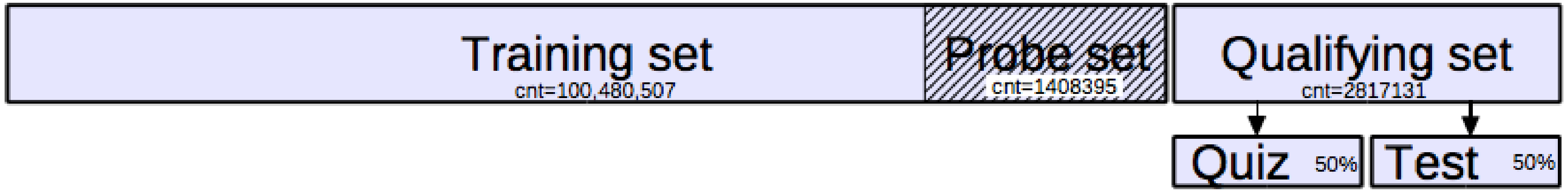
- Majority Voting
  - Return most popular prediction from multiple prediction algorithms
- Bootstrap Aggregation, aka Bagging
  - Train algorithm repeatedly on different random subsets of the training set
- Boosting
  - Train algorithms that each specialize on different “hard” training examples
- Stacking
  - Train a model that learns how to aggregate classifiers’ predictions

Stacked Generalization, aka Stacking

- Train meta-learner to learn the optimal weighting of each classifiers' predictions for making the final prediction
- Algorithm:
  1. Split dataset into three disjoint sets.
  2. Train several base learners on the first partition.
  3. Test the base learners on the second partition and third partition.
  4. Train meta-learner on second partition using classifiers' predictions as features
  5. Evaluate meta-learner on third prediction using classifiers' predictions as features

Ensemble Learner Won Netflix Prize “Challenge”

- In 2009 challenge, winning team won $1 million using ensemble approach:
  - https://www.netflixprize.com/assets/GrandPrize2009_BPC_BigChaos.pdf
  - Dataset: 5-star ratings on 17770 movies from 480189 “anonymous” users collected by Netflix over ~7 years. In total, the number of ratings is 100,480,507.



- Netflix did not use ensemble recommendation system. Why?
- “We evaluated some of the new methods offline but the additional accuracy gains that we measured did not seem to justify the engineering effort needed to bring them into a production environment” -https://medium.com/netflix-techblog/netflix-recommendations-beyond-the-5-stars-part-1-55838468f429
  - Computationally slow and complex from using “sequential” training of learners

Today's Topics

- One-vs-all multiclass classification
- Classifier confidence
- Evaluation: ROC and PR-curves
- Ensemble learning