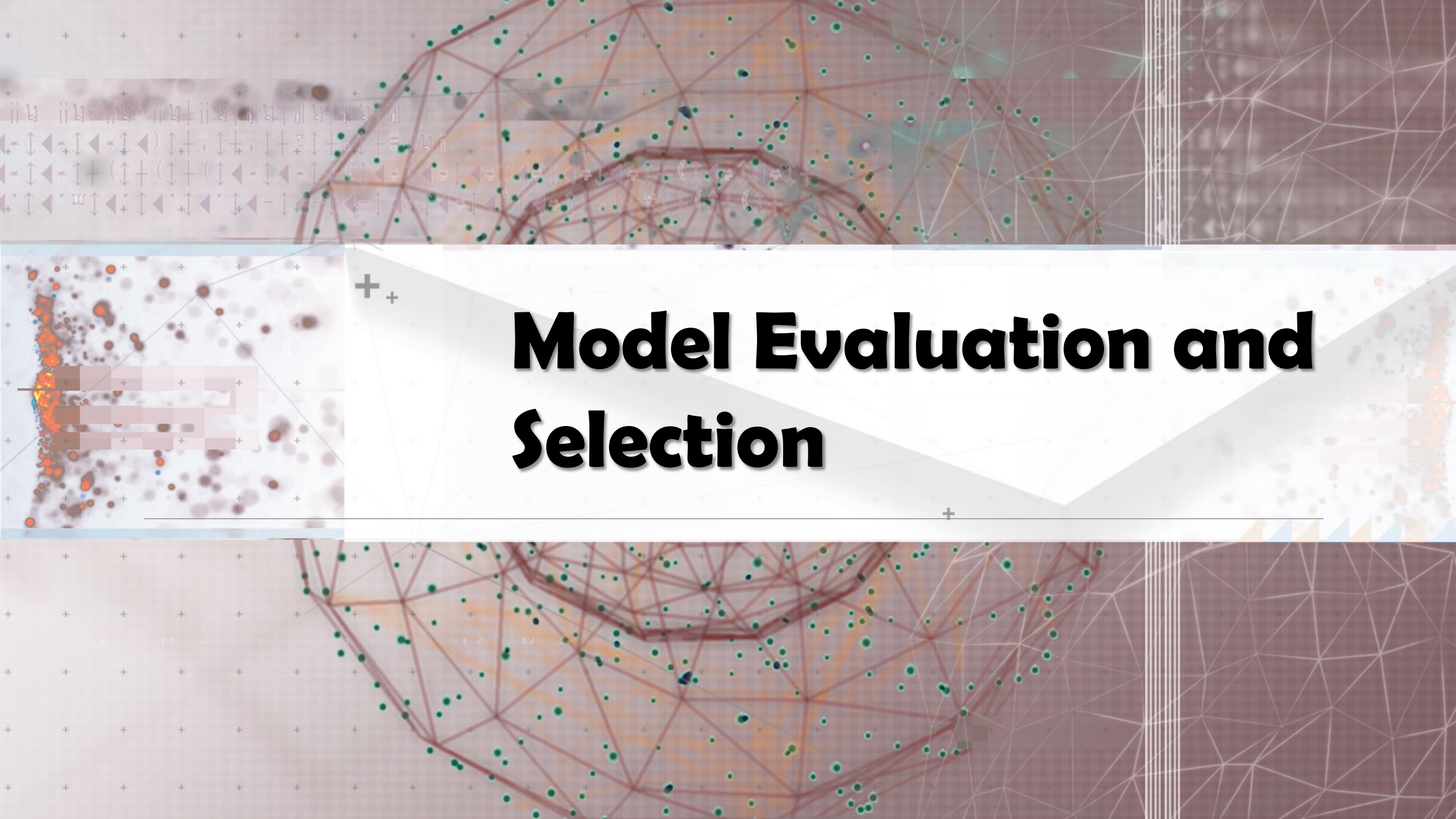
Evaluation and Improvement of Classification Quality

Outline

- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Multi-Class Classification



Model Evaluation and Selection

Model Evaluation and Selection

- How to evaluate the quality of a classifier?
  - A typical measure: Accuracy
  - Other metrics to consider?
- How to assess the classification quality?
  - Use (independent) **validation test set** instead of training set when assessing accuracy
- Methods for estimating a classifier's accuracy
  - Holdout method
  - Cross-validation
  - Bootstrap
- Comparing classifiers using ROC Curves

Classifier Evaluation Metrics: Confusion Matrix

- Confusion Matrix:

Actual class\Predicted class	C_1	¬ C_1
C_1	**True Positives (TP)**	**False Negatives (FN)**
¬ C_1	**False Positives (FP)**	**True Negatives (TN)**

- In a confusion matrix with *m* classes, $CM_{i,j}$ indicates # of tuples in class *i* that were labeled by the classifier as class *j*
  - May have extra rows/columns to provide totals
- An example of Confusion Matrix:

Actual class\Predicted class	buy_computer = yes	buy_computer = no	Total
buy_computer = yes	**6954**	**46**	7000
buy_computer = no	**412**	**2588**	3000
Total	7366	2634	10000

Classifier Evaluation Metrics: Accuracy, Error Rate, Sensitivity and Specificity

A confusion matrix

A\P	C	¬C	
C	**TP**	**FN**	**P**
¬C	**FP**	**TN**	**N**
	P'	**N'**	**All**

- **Classifier accuracy,** or recognition rate
  - Percentage of test set tuples that are correctly classified

    Accuracy = (TP + TN)/All
- **Error rate:** *1 – accuracy*, or

  Error rate = (FP + FN)/All

- **Class imbalance problem**
  - One class may be *rare*
    - E.g., fraud, or HIV-positive
  - Significant *majority of the negative class* and minority of the positive class
- Measures handle the class imbalance problem
  - **Sensitivity** (recall): True positive recognition rate
    - **Sensitivity = TP/P**
  - **Specificity**: True negative recognition rate
    - **Specificity = TN/N**

Classifier Evaluation Metrics: Precision and Recall, and F-measures

- **Precision** (Exactness): What percentage of tuples labeled as positive is actually positive?

$$\mathrm{P} = \mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

- **Recall** (Completeness): What percentage of positive tuples are labeled as positive?

$$\mathrm{R} = \mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

  - Range: [0, 1]
  - The "inverse" relationship between precision & recall
- ***F* measure (**or ***F*-score):** Harmonic mean of precision and recall
  - In general, it is the weighted measure of precision & recall

$$\mathrm{F}_\beta = \frac{1}{\alpha \cdot \frac{1}{\mathrm{P}} + (1 - \alpha) \cdot \frac{1}{\mathrm{R}}} = \frac{(\beta^2 + 1)\mathrm{PR}}{\beta^2 \mathrm{P} + \mathrm{R}}$$

Assigning β times as much weight to recall as to precision)

  - ***F1-measure (balanced F-measure)***
    - That is, when β = 1,

$$\mathrm{F}_1 = \frac{2\mathrm{PR}}{\mathrm{P} + \mathrm{R}}$$

Classifier Evaluation Metrics: Example

- Use the same confusion matrix, calculate the measure just introduced

Actual Class\Predicted class	cancer = yes	cancer = no	Total	Recognition(%)
cancer = yes	**90**	**210**	300	30.00 (*sensitivity*)
cancer = no	**140**	**9560**	9700	98.56 (*specificity*)
Total	230	9770	10000	96.50 (*accuracy*)

- Sensitivity = TP/P = 90/300 = 30%
- Specificity = TN/N = 9560/9700 = 98.56%
- Accuracy = (TP + TN)/All = (90+9560)/10000 = 96.50%
- Error rate = (FP + FN)/All = (140 + 210)/10000 = 3.50%
- Precision = TP/(TP + FP) = 90/(90 + 140) = 90/230 = 39.13%
- Recall = TP/ (TP + FN) = 90/(90 + 210) = 90/300 = 30.00%
- F1 = 2 P × R /(P + R) = 2 × 39.13% × 30.00%/(39.13% + 30%) = 33.96%

Classifier Evaluation: Holdout & Cross-Validation

- **Holdout method**
  - The given data set is randomly partitioned into two independent sets
    - Training set (e.g., 2/3) for model construction
    - Test set (e.g., 1/3) for accuracy estimation
  - Repeated random sub-sampling validation: A variation of holdout
    - Repeat holdout k times, accuracy = average of the accuracies obtained
- **Cross-validation** (*k*-fold, where k = 10 is most popular)
  - Randomly partition the data into *k mutually exclusive* subsets, each approximately equal size
  - At *i*-th iteration, use D_i as test set and others as training set
  - Leave-one-out: *k* folds where *k* = # of tuples, for small sized data
  - ***Stratified cross-validation***: Folds are stratified so that class distribution in each fold is approximately the same as that in the initial data

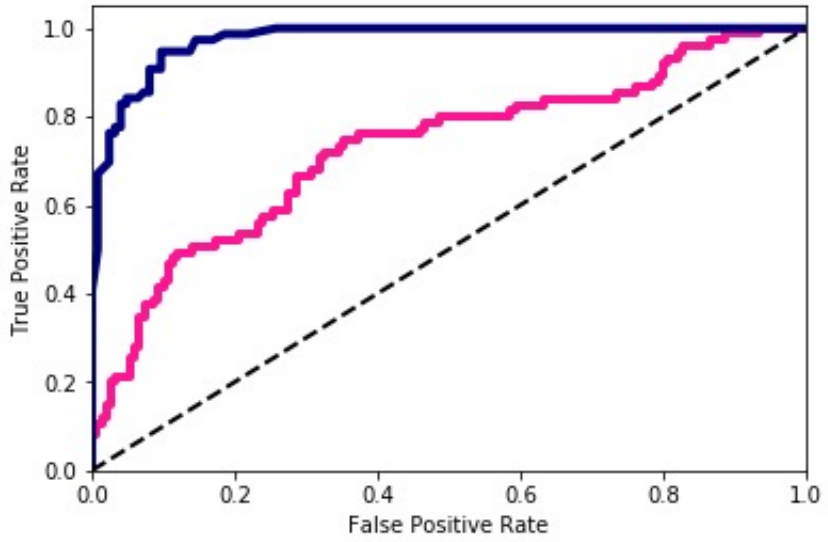
Classifier Evaluation: Bootstrap

- **Bootstrap**
  - Works well with small data sets
  - Samples the given training tuples uniformly *with replacement*
    - Each time a tuple is selected, it is equally likely to be selected again and re-added to the training set
- Several bootstrap methods, and a common one is **.632 bootstrap**
  - A data set with *d* tuples is sampled *d* times, with replacement, resulting in a training set of *d* samples. The data tuples that did not make it into the training set end up forming the test set. About 63.2% of the original data end up in the bootstrap, and the remaining 36.8% form the test set (since $(1 - 1/d)^d \approx e^{-1} = 0.368$)
  - Repeating the sampling procedure *k* times, the overall accuracy of the model:

$$Acc(M) = \frac{1}{k}\sum_{i=1}^{k}(0.632 \times Acc(M_i)_{test\_set} + 0.368 \times Acc(M_i)_{train\_set})$$

Model Selection: ROC Curves

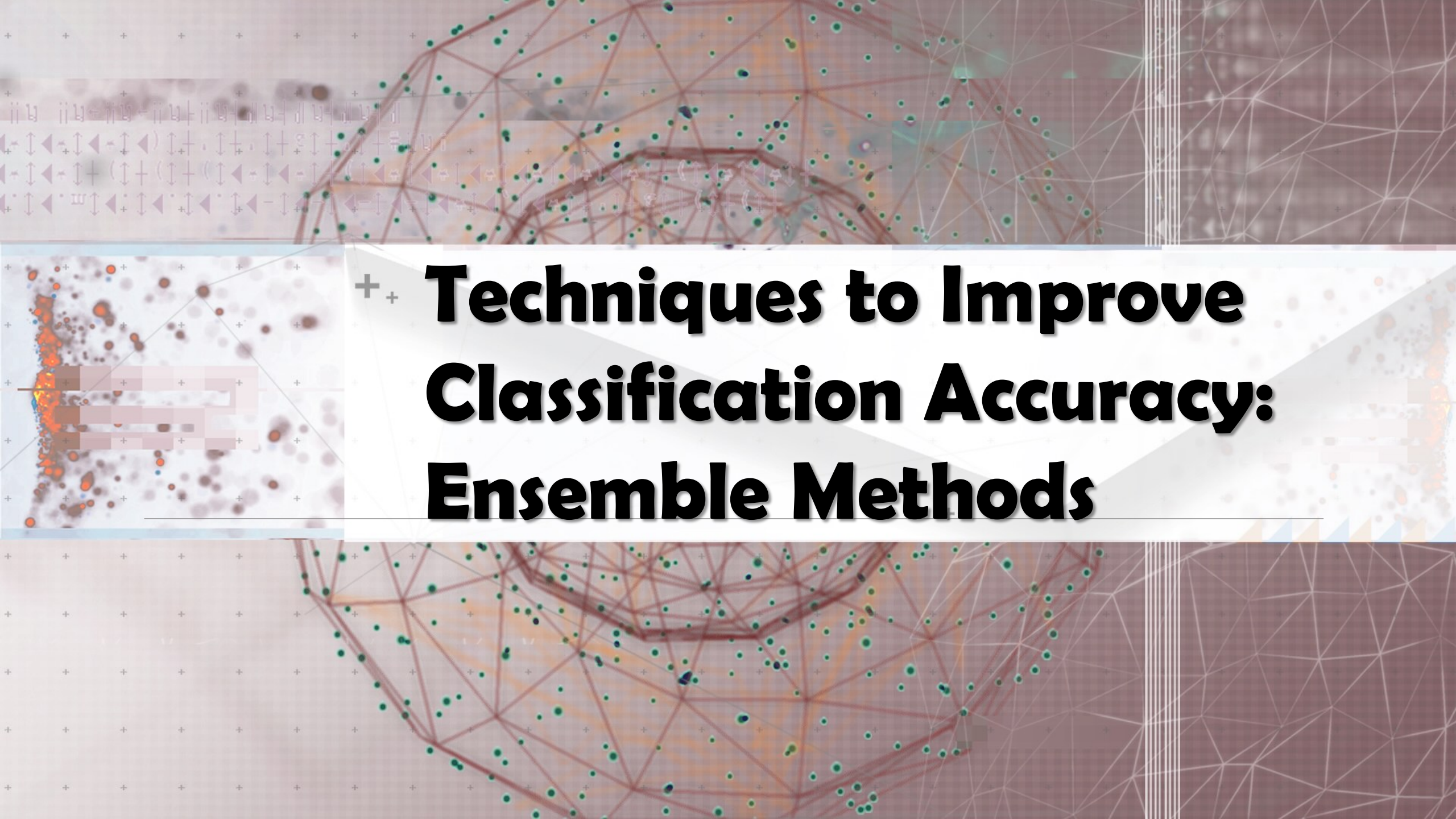
- **ROC** (Receiver Operating Characteristics) curve: for visual comparison of classification models
- Originated from signal detection theory
- Shows the trade-off between the true positive rate and the false positive rate
- The area under the ROC curve (**AUC**: Area Under Curve) is a measure of the accuracy of the model
- Rank the test tuples in decreasing order: The one that is most likely to belong to the positive class appears at the top of the list
- The closer to the diagonal line (i.e., the closer the area is to 0.5), the less accurate is the model



- The vertical axis represents the true positive rate
- The horizontal axis represents the false positive rate
- The plot also shows a diagonal line
- A model with perfect accuracy will have an area of 1.0

Issues Affecting Model Selection

- **Accuracy**
  - Classifier accuracy: predicting class label
- **Speed**
  - Time to construct the model (training time)
  - Time to use the model (classification/prediction time)
- **Robustness**: Handling noise and missing values
- **Scalability**: Efficiency in disk-resident databases
- **Interpretability**
  - Understanding and insight provided by the model
- Other measures
  - E.g., goodness of rules, such as decision tree size or compactness of classification rules



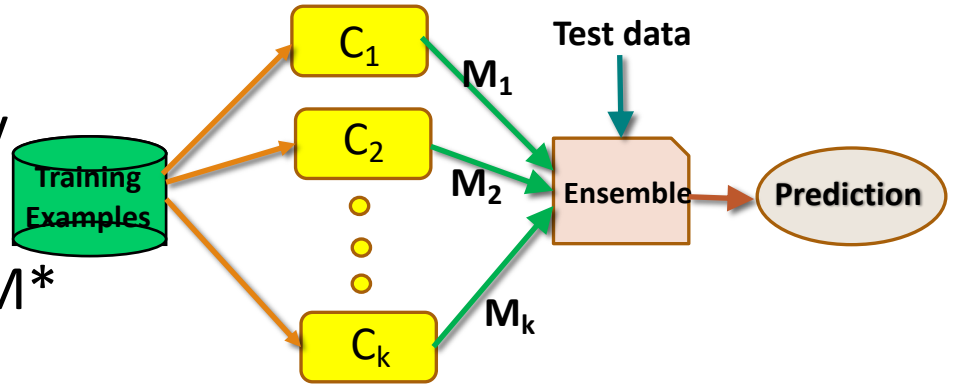
Techniques to Improve Classification Accuracy: Ensemble Methods

Outline

- Techniques to Improve Classification Accuracy: Ensemble Methods
- Bagging: Bootstrap Aggregation
- Random Forest: Basic Concepts and Methods
- Boosting and AdaBoost
- Classification of Class-Imbalanced Data Sets
- Classifying Data Streams with Skewed Distribution

Ensemble Methods: Increasing the Accuracy

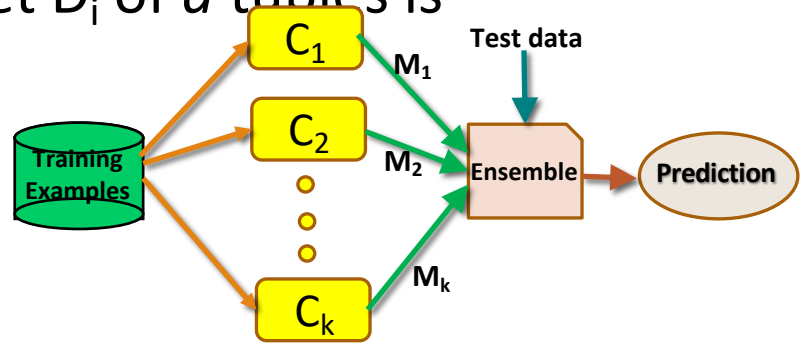
- Ensemble methods
  - Use a combination of models to increase accuracy
  - Combine a series of *k* learned models, M_1, M_2, …, M_k, with the aim of creating an improved model M*



- Popular ensemble methods
  - Bagging: Trains each model using a subset of the training set,
    - Models learned independently, in parallel
    - Simple voting: Outcome is determined by the majority of the parallel models
  - Boosting: Trains each new model instance to emphasize the training instances that previous models misclassified (correcting the "errors" of previous model)
    - Models learned in order (a sequential ensemble)
    - Weighted voting: Outcome is determined by the majority—but the sequential models were built by assigning greater weights to misclassified instances of the previous models

Bagging: Bootstrap Aggregation

- Analogy: Diagnosis based on multiple doctors' majority vote
- Training
  - Given a set D of tuples, at each iteration *i*, a training set D_i of *d* tuples is sampled with replacement from D (i.e., bootstrap)
  - A classifier model M_i is learned for each training set D_i

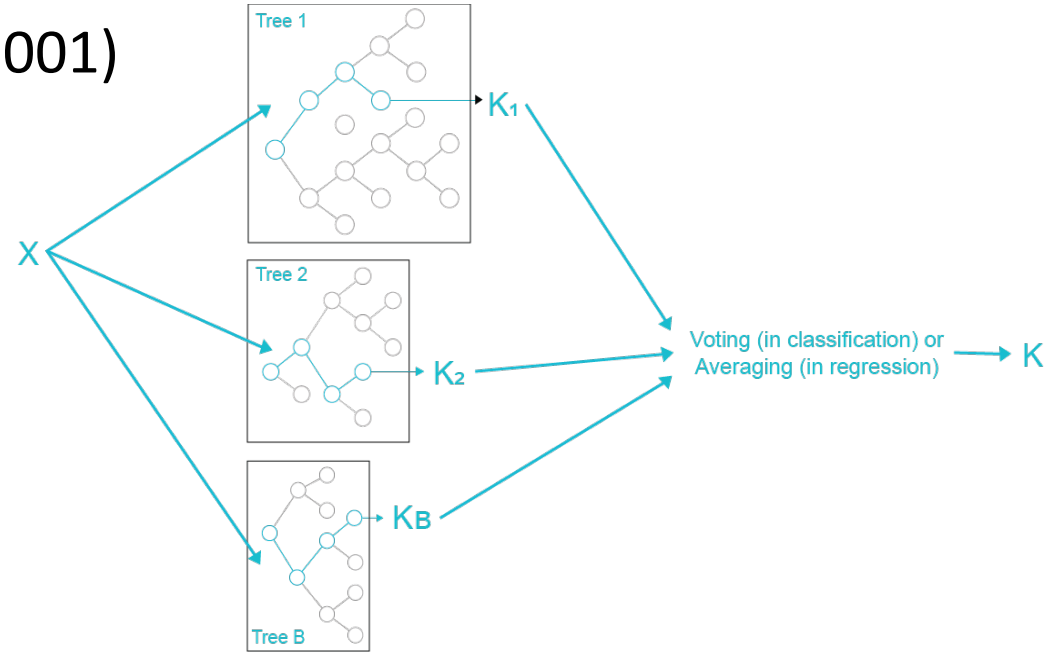


- Classification: Classify an unknown sample **X**
  - Each classifier M_i returns its class prediction
  - The bagged classifier M* counts the votes and assigns the class with the most votes to **X**
- Prediction: It can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple
- Accuracy: Improved accuracy in prediction
  - Often significantly better than a single classifier derived from D
  - For noise data: Not considerably worse, more robust

Random Forest: Basic Concepts

- Random Forest (first proposed by L. Breiman in 2001)
  - A variation of bagging for *decision trees*
  - *Data bagging*
    - Use a subset of training data

      by sampling with replacement for each tree
  - *Feature bagging*
    - At each node use a random selection of attributes as candidates and split by the best attribute among them
  - Comparing with original bagging, the *random forests* method increases the diversity among generated trees
  - During classification, each tree votes and the most popular class is returned



Methods for Constructing Random Forest

- Two Methods to construct Random Forest:
  - Forest-RI (*random input selection*)
    - Randomly select, at each node, *F* attributes as candidates for the split at the node
      - The CART methodology is used to grow the trees to maximum size
  - Forest-RC (*random linear combinations*)
    - Creates new attributes (or features) that are a linear combination of the existing attributes (reduces the correlation between individual classifiers)
- Comparable in accuracy to Adaboost, but more robust to errors and outliers
- Insensitive to the number of attributes selected for consideration at each split, and faster than typical bagging or boosting

Boosting

- Analogy: Consult several doctors, based on a combination of weighted diagnoses—weight assigned based on the previous diagnosis accuracy
- How boosting works?
  - **Weights** are assigned to each training tuple
  - A series of *k* classifiers is iteratively learned
  - After a classifier M_i is learned, the weights are updated to allow the subsequent classifier, M_{i+1}, to **pay more attention to the training tuples that were misclassified** by M_i
  - The final **M* combines the votes** of each individual classifier, where the weight of each classifier's vote is a function of its accuracy
- Boosting algorithm can be extended for numeric prediction
- Comparing with bagging: Boosting tends to have greater accuracy, but it also risks overfitting the model to misclassified data

Adaboost (Freund and Schapire, 1997)

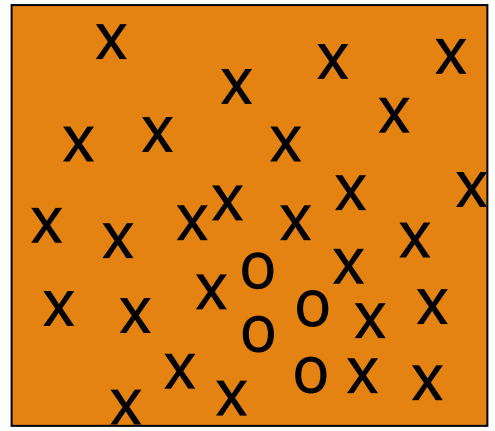
- Given a set of *d* class-labeled tuples, ($\mathbf{X_1}$, y_1), …, ($\mathbf{X_d}$, y_d)
- Initially, all the weights of tuples are set the same (i.e., 1/d)
- Generate *k* classifiers in *k* rounds. At round i,
  - Tuples from D are sampled (with replacement) to form a training set D_i of the same size
    - Each tuple's chance of being selected is based on its weight
  - A classification model M_i is derived from D_i
    - Its error rate is calculated using D_i as a test set
  - **If a tuple is misclassified, its weight is increased; otherwise, it is decreased**
- Error rate: err($\mathbf{X_j}$) is the misclassification error of tuple $\mathbf{X_j}$
- Classifier M_i error rate is the sum of the weights of the misclassified tuples:

$$error(M_i) = \sum_j^d w_j \times err(\mathbf{X_j})$$

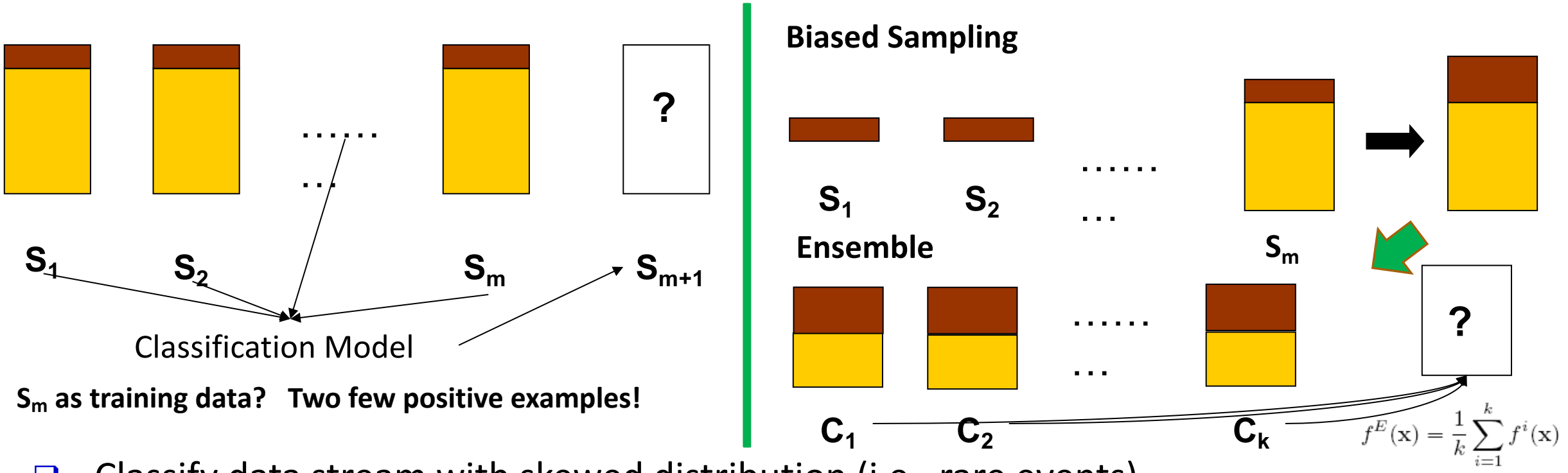
- The weight of classifier M_i's vote is $\log \frac{1 - error(M_i)}{error(M_i)}$

Classification of Class-Imbalanced Data Sets

- Class-imbalance problem: Rare positive examples but numerous negative ones
  - E.g., medical diagnosis, fraud transaction, accident (oil-spill), and product fault
- Traditional methods assume a balanced distribution of classes and equal error costs: not suitable for class-imbalanced data
- Typical methods on imbalanced data in two-class classification
  - **Oversampling**: Re-sampling of data from positive class
  - **Under-sampling**: Randomly eliminate tuples from negative class
  - **Threshold-moving**: Move the decision threshold, t, so that the rare class tuples are easier to classify, and hence, less chance of costly false negative errors
  - **Ensemble techniques**: Ensemble multiple classifiers introduced above
- Still difficult for class imbalance problem on multiclass tasks



Classifying Data Streams with Skewed Distribution

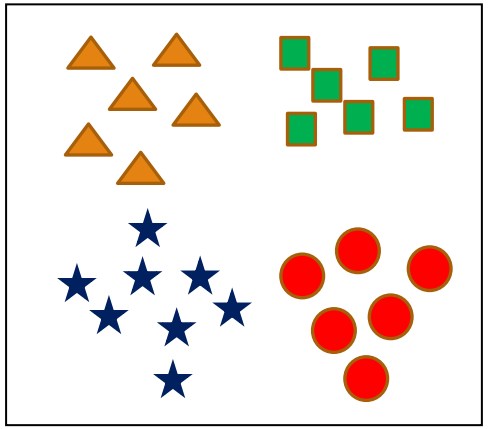


- Classify data stream with skewed distribution (i.e., rare events)
  - **Biased sampling:** Save only the positive examples in the streams
  - **Ensemble:** Partition negative examples of S_m into k portions to build k classifiers
  - Effectively reduce classification errors on the minority class

J. Gao, et al., "A General Framework for Mining Concept-Drifting Data Streams with Skewed Distributions", SDM'07

Multi-Class Classification

Multi-Class Classification



- Classification involving more than two classes (i.e., > 2 Classes)
- Methodology
  - Reducing the multi-class problem into multiple binary problems
- Method 1. **One-vs.-rest** (or **one-vs.-all**)
  - Given *m* classes, train *m* classifiers: one for each class
  - Classifier j: treat tuples in class j as *positive* & **all the rest** as *negative*
  - To classify a tuple **X**, the set of classifiers vote as an ensemble
- Method 2. **one-vs.-one** (or **all-vs.-all**): Learn a classifier for each pair of classes
  - Given *m* classes, construct $m(m-1)/2$ binary classifiers
  - A classifier is trained using tuples of the two classes
  - To classify **X**, each classifier votes: **X** is assigned to the class with maximal vote
- Comparison: One-vs.-one tends to perform better than one-vs.-rest
- Many new algorithms have been developed to go beyond binary classifier method

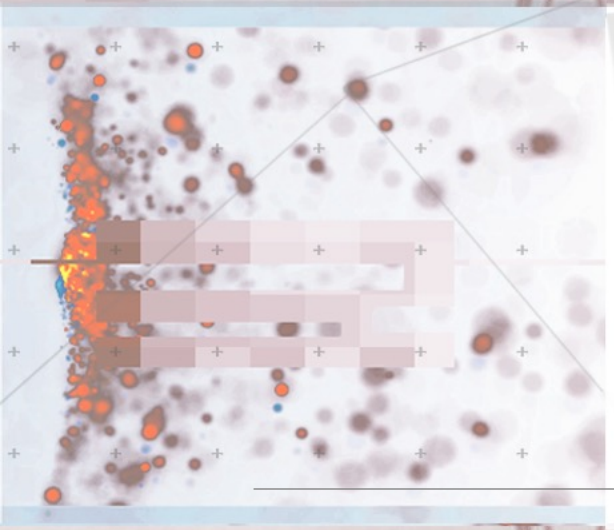
Summary

Summary

- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Multi-Class Classification

Recommended Readings

- L. Breiman, “Bagging predictors”, Machine Learning, 24 (2): 123–140, 1996
- B. Efron and R. Tibshirani. An Introduction to the Bootstrap. Chapman & Hall/CRC, 1993
- Y. Freund and R. E. Schapire, “A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting”, J. of Computer and System Sciences (JCSS), 55(1):119-139, 1997
- J. Gao, W. Fan and J. Han, “A General Framework for Mining Concept-Drifting Data Streams with Skewed Distributions”, Proc. of SDM, 2007
- R. Grossman, G. Seni, J. Elder, N. Agarwal, H. Liu, Ensemble Methods in Data Mining: Improving Accuracy Through Combining Predictions. Morgan & Claypool, 2010
- R. Kohavi, “A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection”, IJCAI 1995
- Y. Sun, A. K. C. Wong, M. S. Kamel, “Classification of Imbalanced Data: A Review”, Int. Journal of Pattern Recognition and Artificial Intelligence, 2009
- Z.-H. Zhou, Ensemble Methods: Foundations and Algorithms, Chapman & Hall/Crc, 2012



Classification with Weak Supervision

Outline

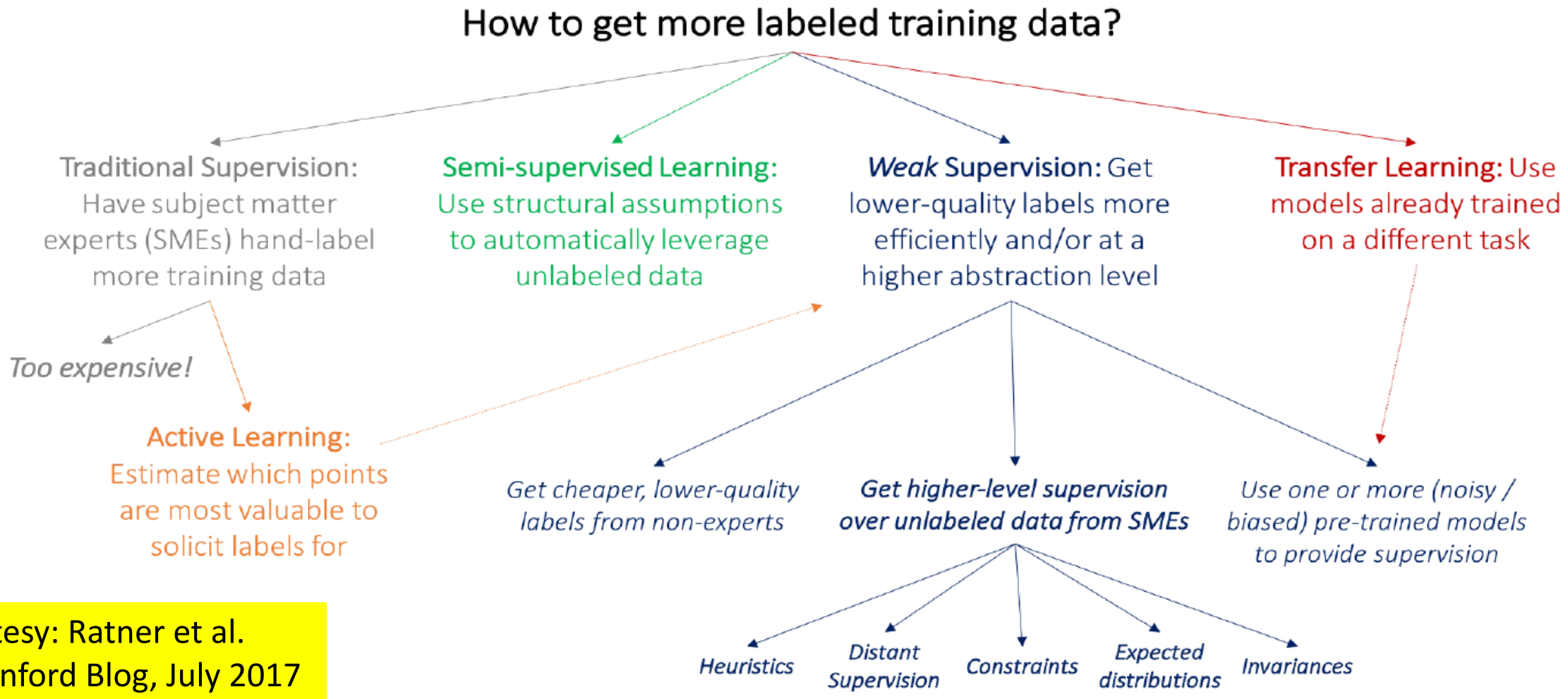
- Weak Supervision: An Important Paradigm in Machine Learning
- Semi-Supervised Classification
- Classification with Distant Supervision
- Active Learning
- Transfer Learning
- Zero-Shot Learning

Classification with Weak Supervision: An Introduction

Weak Supervision: An Important Paradigm in Machine Learning

- Overcome the training data bottleneck
  - Leverage higher-level and/or noisier input from experts
- Exploring *weak label distributions* provided more cheaply and efficiently by
  - Higher-level, less precise supervision (e.g., heuristic rules, expected label distributions)
  - Cheaper, lower-quality supervision (e.g., crowdsourcing)
  - Existing resources (e.g., knowledge bases, pre-trained models)
- These weak label distributions could take many forms
  - Weak Labels from different sources, such as crowd workers, output of heuristic rules, result of distant supervision (from KBs), or output of other classifiers
  - Constraints and invariances (e.g., from physics, logic, or other experts)
  - Probability distributions (e.g., from weak or biased classifiers or user-provided label or feature expectations or measurements)

Relationships Among Different Kinds of Supervisions



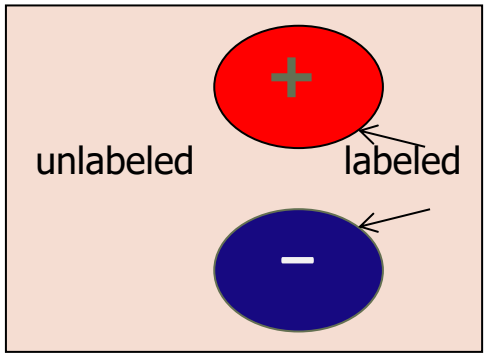
Courtesy: Ratner et al.
@Stanford Blog, July 2017

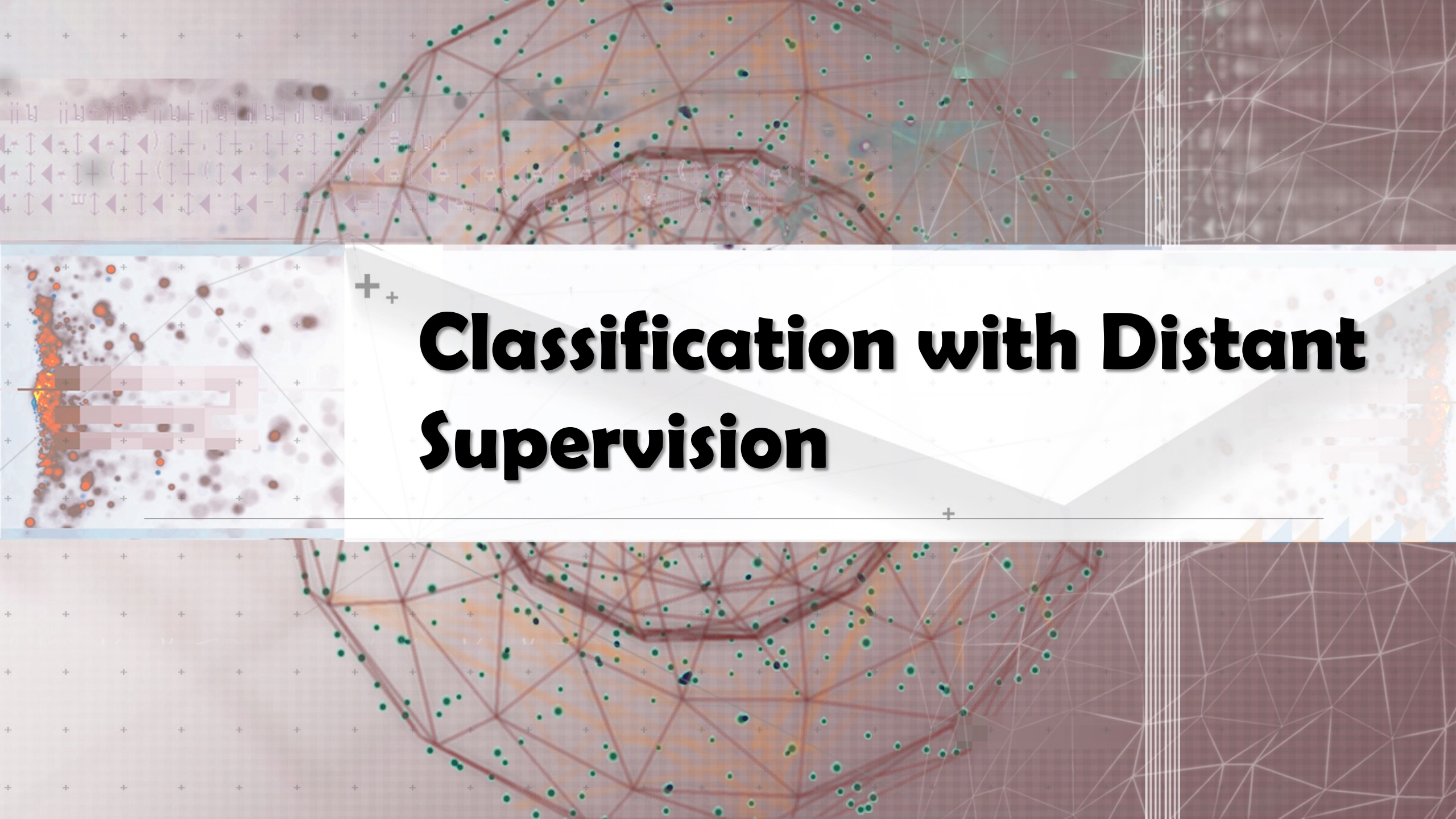
Many areas of machine learning are motivated by the bottleneck of labeled training data, but are divided at a high-level by what information they leverage instead.

Semi-Supervised Classification

Semi-Supervised Classification

- Semi-supervised: Uses both labeled and unlabeled data to build a classifier
- Self-training
  - Build a classifier using the labeled data
  - Use it to label the unlabeled data, and those with the most confident label prediction are added to the set of labeled data
  - Repeat the above process
  - Adv.: easy to understand; Disadv.: may reinforce errors
- Co-training: Use two or more classifiers to teach each other
  - Each learner uses a mutually independent set of features of each tuple to train a good classifier, say f_1 and f_2
  - Then f_1 and f_2 are used to predict the class label for unlabeled data X
  - Teach each other: The tuple having the most confident prediction from f_1 is added to the set of labeled data for f_2 & vice versa
- Other methods include joint probability distribution of features and labels

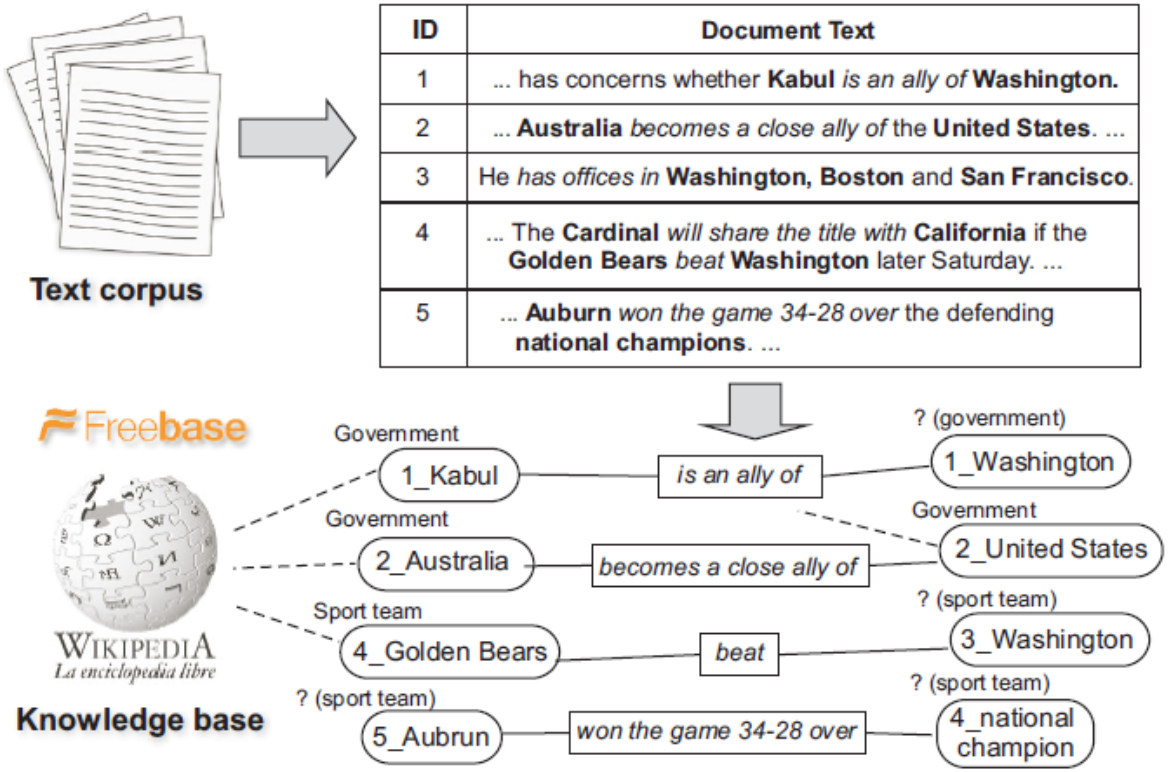




Classification with Distant Supervision

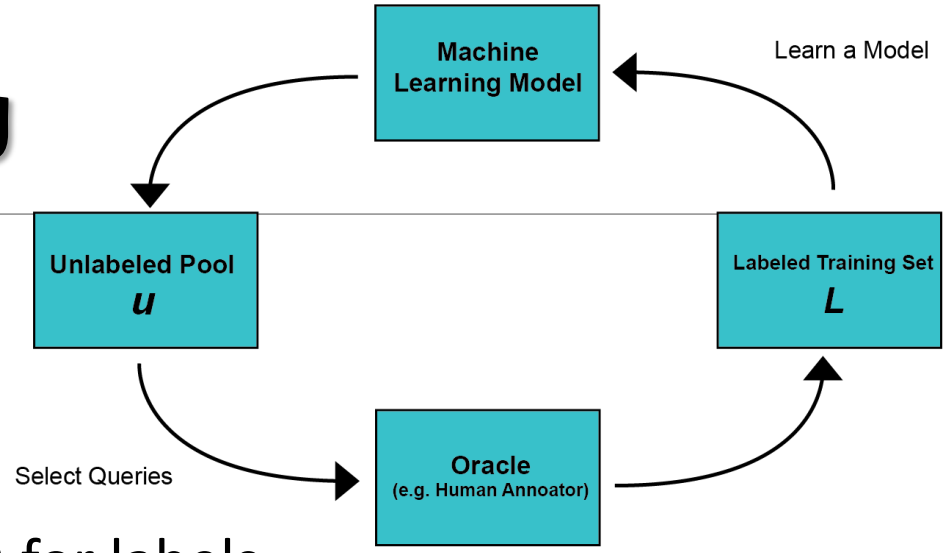
Classification with Distant Supervision

- **Distant supervision**: Training from distantly labeled data (e.g., typical, from a *knowledge base* (e.g., Freebase) or a *domain-specific database*)
  - Ex. Freebase: United States [country, government], Golden Bears [Sport team]
    - Take such labeled pairs as *positive examples* (a large number of training data)
- Distant supervision may use such training data to do "relation/type clustering"
  - [S1] "… has concerns whether Kabul is an allay of *Washington*" [government]
  - [S2] "… Australia becomes a close ally of the United States"
  - [S3] "He has offices in *Washington*, Boston, …"
- Other training sources, such as training tweets using their linked news, ODP (Open Directory Project), or YouTube videos



Active Learning

Active Learning

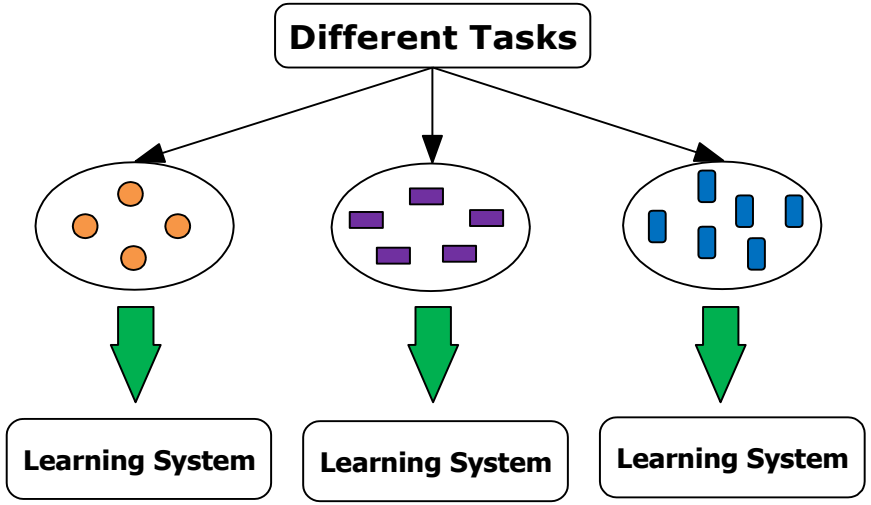


- A special case of semi-supervised learning
  - Unlabeled data: Abundant
  - Class labels are expensive to obtain
- Active learner: Interactively query teachers (oracle) for labels
- Pool-based approach: Uses a pool of unlabeled data
  - L: a small set of labeled data in D; U: a large pool of unlabeled data in D
  - Use a query function to carefully select one or more tuples from U and request labels from an oracle (a human annotator)
  - The newly labeled samples are added to L, and learn a model
  - Goal: **Achieve high accuracy using as few labeled data as possible**
- Evaluated using *learning curves*: Accuracy as a function of the number of instances queried (# of tuples to be queried should be small)
- A lot of algorithms have been developed for active learning

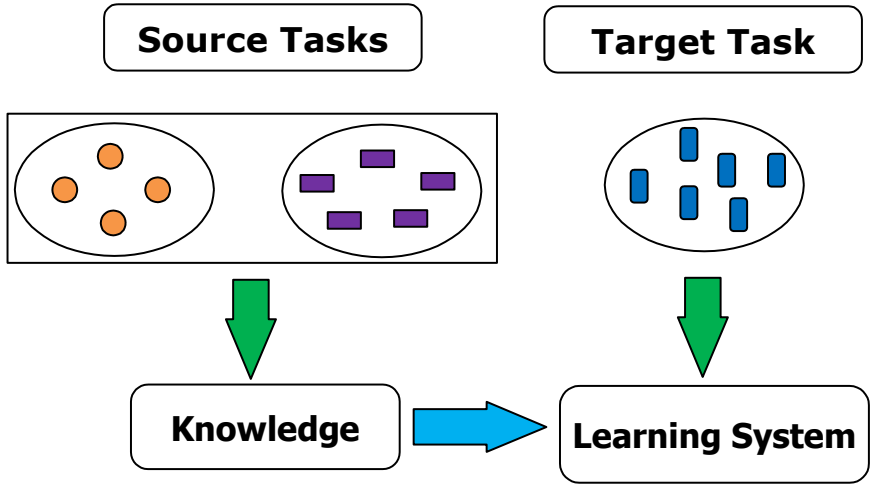
Transfer Learning

Transfer Learning

- Transfer learning: Extract knowledge from one or more source tasks (e.g., recognizing cars) and apply the knowledge to a target task (e.g., recognizing trucks)
- Traditional learning: Build a new classifier for each new task
- Transfer learning: Build new classifier by applying existing knowledge learned from source tasks
- Many algorithms are developed, applied to text classification, spam filtering, etc.



Traditional Learning Framework

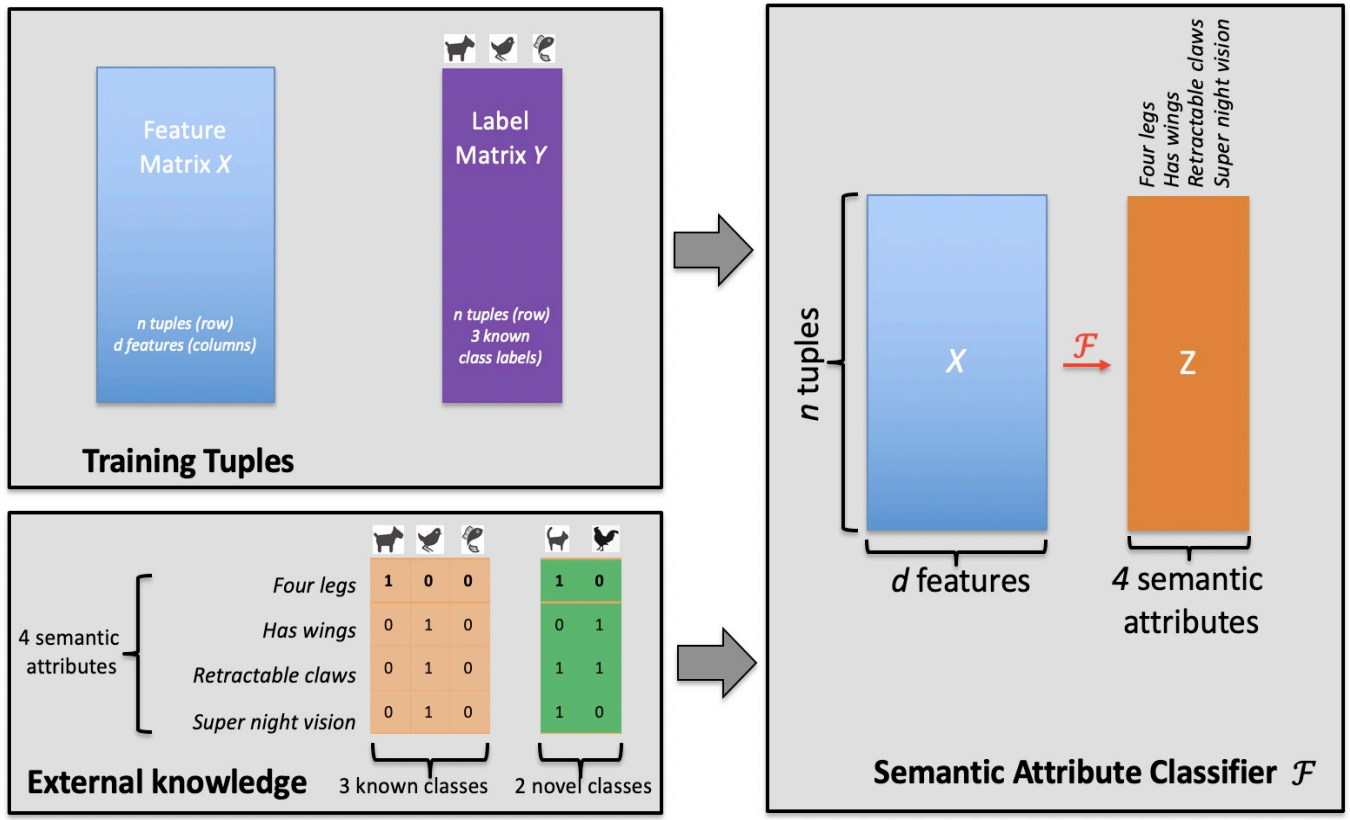


Transfer Learning Framework

Zero-Shot Learning

Zero-Shot Learning

- Build a classifier when there is zero training tuple for the novel class label
  - Classify instances whose classes have not been seen previously
  - Leverage external knowledge or side-information to build a classifier
- Ex. Classify "cat" based on 4 attributes of 3 known classes (dog, owl, fish): '4-legs', 'has wings', 'retractable claws', 'super night vision'
  - Such external knowledge is available for both known classes and novel classes (e.g., "cat" and "rooster")
- Train a semantic attribute classier F, which predicts a 4-dimen. semantic attribute vector for an input image represented by a d-dim. feature vector



Zero-Shot Learning: Forms of External Knowledge

- Zero-shot learning: can be viewed as a special case of transfer learning
  - Transfer the knowledge about the known class labels to novel classes
  - Leverage the semantic attributes as a bridge to transfer the semantic classifier trained on the known class labels to predict the novel class labels
- Forms of external knowledge
  - Attribute info: Classes are accompanied by pre-defined structured description
  - Textual description: class labels are augmented with definitions or natural-language description (e.g., a Wikipedia description of the class)
  - Class-class similarity: Classes are embedded in a continuous space, and the nearest embedded class is used as a predicted class
- Generalized zero-shot learning:
  - Samples to be classified can come from both new and known classes
    - Ex. A test sample can be in one of 5 classes: {cat, rooster, dog, owl, fish}

Summary

- Weak Supervision: An Important Paradigm in Machine Learning
- Semi-Supervised Classification
- Classification with Distant Supervision
- Active Learning
- Transfer Learning
- Zero-Shot Learning

Recommended Readings

- Han, J., Pei, J., & Tong, H. (2022). *Data mining: Concepts and Techniques (4th ed.)*. Morgan Kaufmann
- Zhi-Hua Zhou (2018). “A Brief Introduction to Weakly Supervised Learning”, National Science Review. 5: 44–53
- S. J. Pan and Q. Yang, “A Survey on Transfer Learning”, IEEE Trans. on Knowledge and Data Eng., 2010
- Mike Mintz, Steven Bills, Rion Snow, Dan Jurafsky, “Distant supervision for relation extraction without labeled data”, ACL 2009
- Xiang Ren, Ahmed El-Kishky, Chi Wang, Fangbo Tao, Clare R. Voss, Heng Ji, Jiawei Han, “ClusType: Effective Entity Recognition and Typing by Relation Phrase-Based Clustering”, in KDD 2015