CS 412 Intro. to Data Mining

Chapter 8. Classification: Advanced Methods

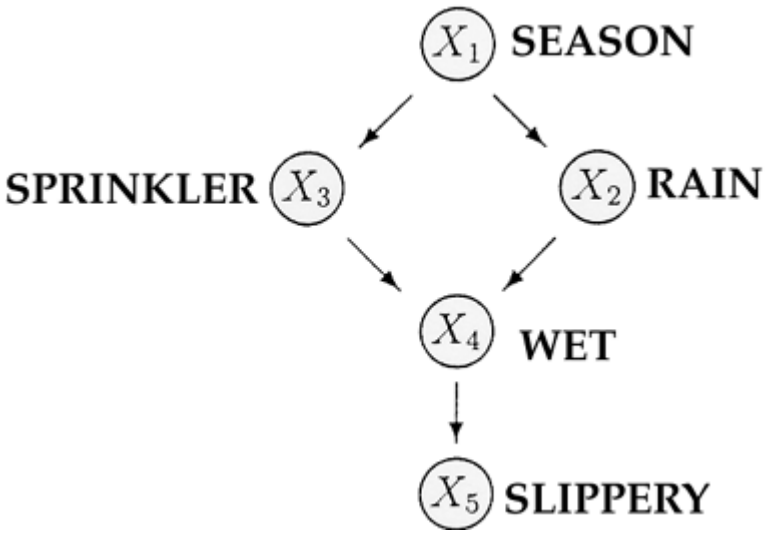
Arindam Banerjee, Computer Science, UIUC, Fall 2022

Chapter 8. Classification: Advanced Methods

- Bayesian Belief Networks
- Support Vector Machines
- Rule-Based and Pattern-Based Classification
- Classification with Weak Supervision
- Classification with Rich Data Type
- Summary

From Naïve Bayes to Bayesian Networks

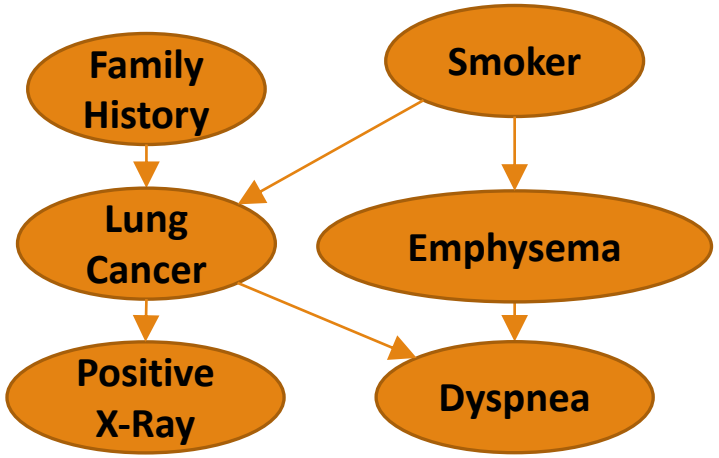
- Naïve Bayes classifiers assume that the value of a particular feature is **independent** of the value of any other feature, given the class variable
  - This assumption is often too simple to model the real world well
- Bayesian network (or Bayes network, belief network, Bayesian model or probabilistic directed acyclic graphical model) is a **probabilistic graphical model**
  - Represented by a set of *random variables* and *their conditional dependencies* via a *directed acyclic graph* (DAG)
  - E.g. Given symptoms, the network can be used to compute the probabilities of the presence of various diseases

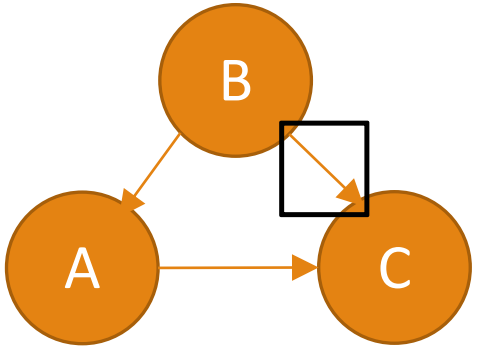


Bayesian Belief Networks

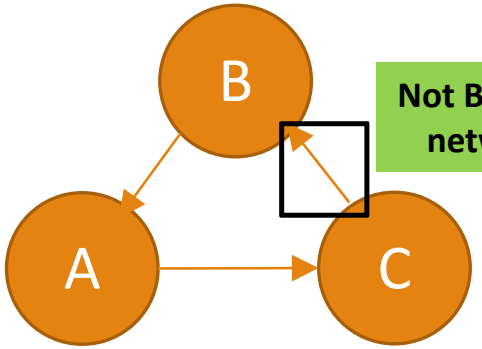
- **Bayesian belief network** (or **Bayesian network, probabilistic network**):
  - Allows *class conditional independencies* between *subsets* of variables
- Two components:
  - A *directed acyclic graph* (called a structure)
  - A set of *conditional probability tables* (CPTs)

Nodes: random variables Links: dependency





directed **acyclic** graphical model



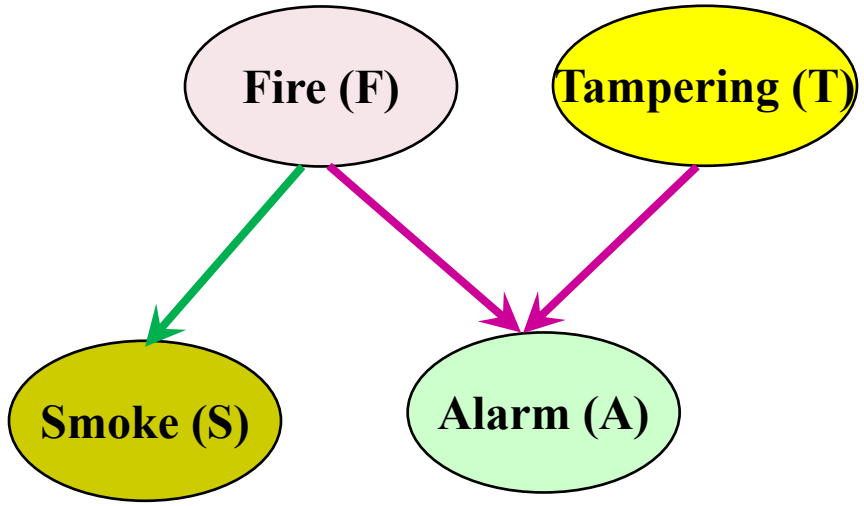
Not Bayesian networks

directed **cyclic** graphical model

$$p(A, B, C) = p(B) \cdot p(A|B) \cdot p(C|A, B)$$

Family History	Smoker	Lung Cancer = Y	Lung Cancer = N
Y	Y	0.8	0.2
Y	N	0.5	0.5
N	Y	0.7	0.3
N	N	0.1	0.9

A Bayesian Network and Its CPTs



Conditional **P**robability **T**ables (CPT)

Fire	Smoke = T
T	0.9
F	0.01

Fire	Tampering	Alarm = T
T	T	0.5
T	F	0.99
F	T	0.85
F	F	0.0001

CPT shows the conditional probability for each possible combination of its parents:

$$p(X) = \prod_k p(x_k | \text{Parents}(x_k))$$

$$p(F, S, A, T) = p(F) \cdot p(T) \cdot p(S|F) \cdot p(A|F, T)$$

Training Bayesian Networks: Several Scenarios

- Scenario 1: Given both the network structure and all variables observable: *compute only the CPT entries*
- Scenario 2: Network structure known, some variables hidden: *gradient descent* (greedy hill-climbing) method, i.e., search for a solution along the steepest descent of a criterion function
  - Weights are initialized to random probability values
  - At each iteration, it moves towards what appears to be the best solution at the moment, without backtracking
  - Weights are updated at each iteration & converge to local optimum
- Use Expectation Maximization for working with hidden variables

Example: Learning by Gradient Descent

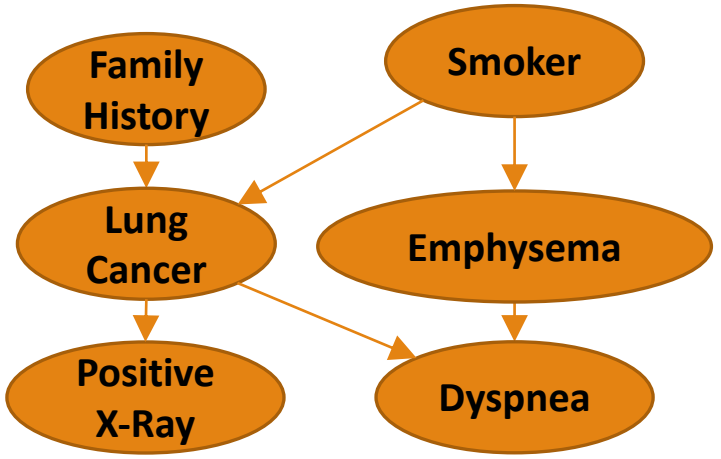
- Parameter wijk = $P(Y_i = y_{ij} \mid U_i = u_{ik})$ where
  - Y_i is "Lung Cancer", y_{ij} = "yes"
  - U_i is {"Family History", "Smoker"}, u_{ik} = {"yes","yes"}
- With D training examples, $P_w(D) = \prod_{d=1}^{|D|} P_w(\boldsymbol{X}_d)$
- Learning by gradient descent
  - Compute the gradients:

$$\frac{\partial \ln P_w(D)}{\partial w_{ijk}} = \sum_{d=1}^{|D|} \frac{\partial \ln(P(Y_i = y_{ij}, U_i = u_{ik}|\boldsymbol{X}_d))}{\partial w_{ijk}}.$$

  - Do gradient descent

$$w_{ijk} \leftarrow w_{ijk} + \eta \frac{\partial \ln P_w(D)}{\partial w_{ijk}},$$

  - Renormalize the weights, since w_{ijk} (over j) is a distribution

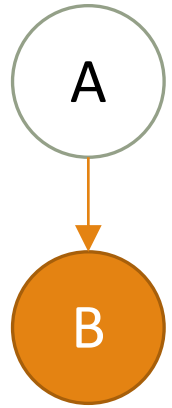


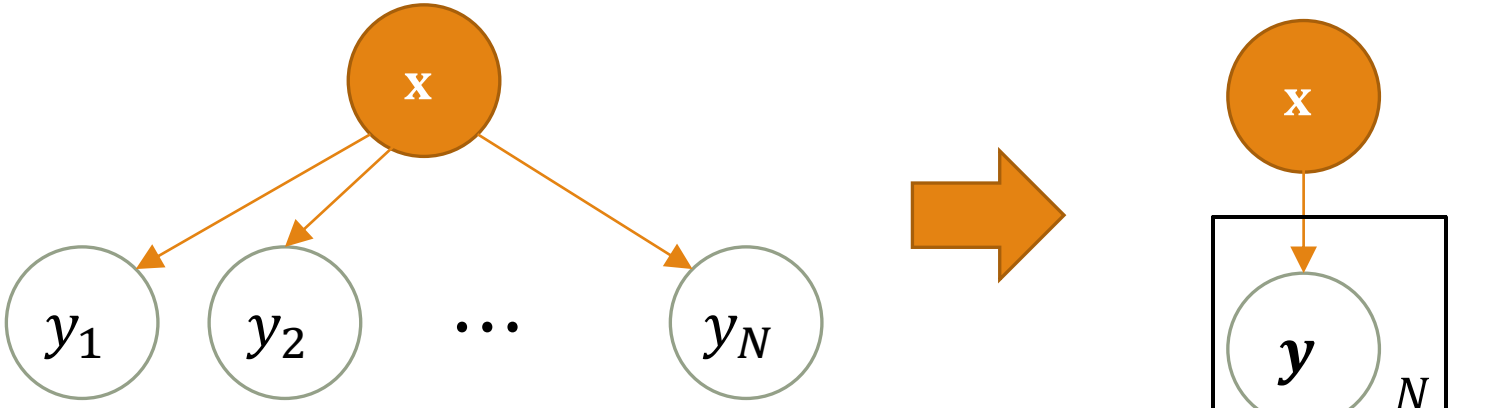
Training Bayesian Networks: Several Scenarios

- Scenario 3: Network structure unknown, all variables observable: search through the model space to *reconstruct network topology*
- Scenario 4: Unknown structure, all hidden variables: No good algorithms known for this purpose
- D. Heckerman. A Tutorial on Learning with Bayesian Networks. In *Learning in Graphical Models,* M. Jordan, ed. MIT Press, 1999
- D. Koller, N. Friedman, Probabilistic Graphical Models: Principles and Techniques, MIT Press, 2009

Probabilistic Graphic Model: Plate Notations

- Represent variables that repeat in a graphical model
- Variables
  - A solid (or shaded) circle means the corresponding variable is *observed*; otherwise it is *hidden*
- Dependency among variables:
  - A Directed Acyclic Graphical (DAG) model
- Using plate notation instead of flat notation

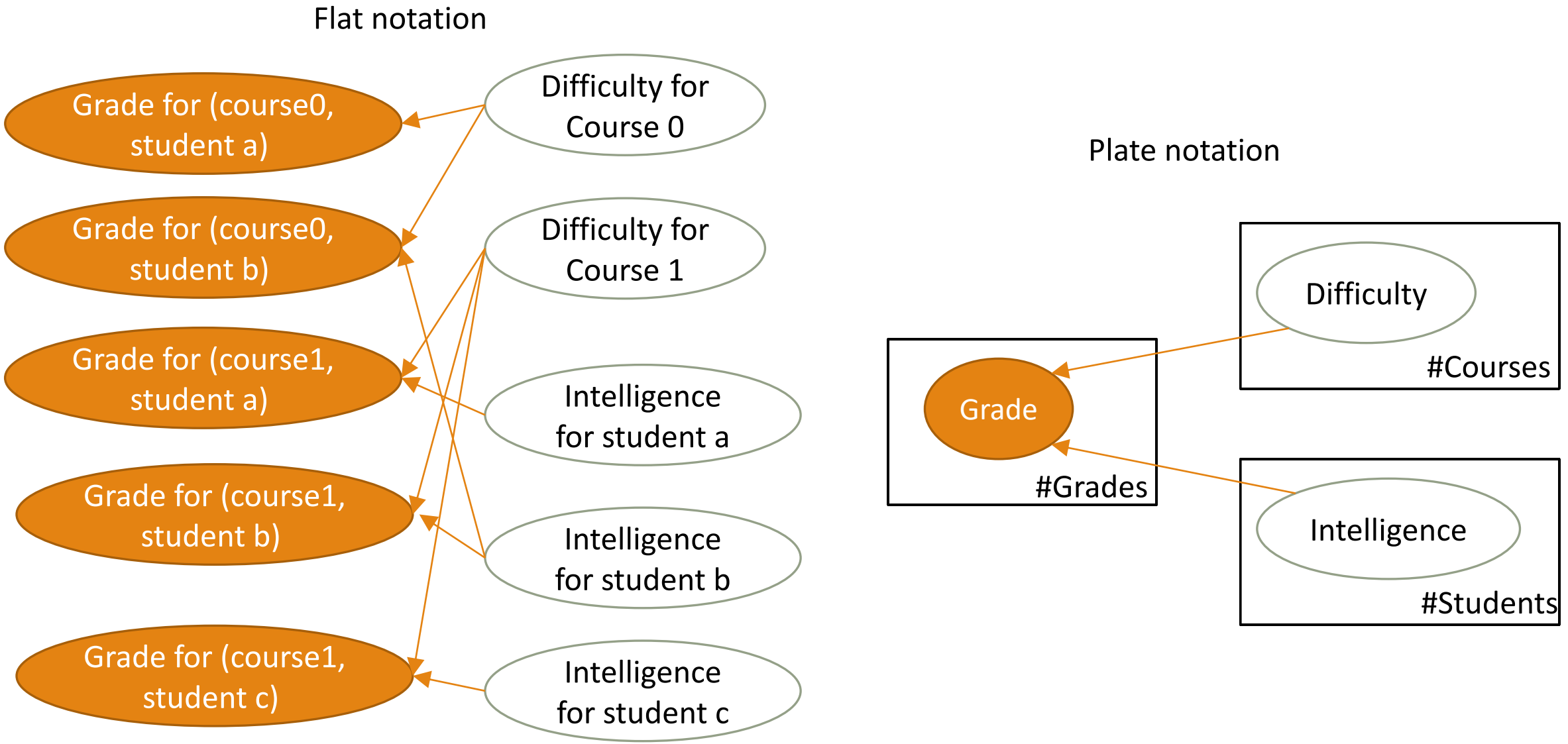




Flat notation

Plate notation

An Example of Plate Notation



Chapter 8. Classification: Advanced Methods

- Bayesian Belief Networks
- Support Vector Machines 
- Rule-Based and Pattern-Based Classification
- Classification with Weak Supervision
- Classification with Rich Data Type
- Potpourri: Other Related Techniques
- Summary

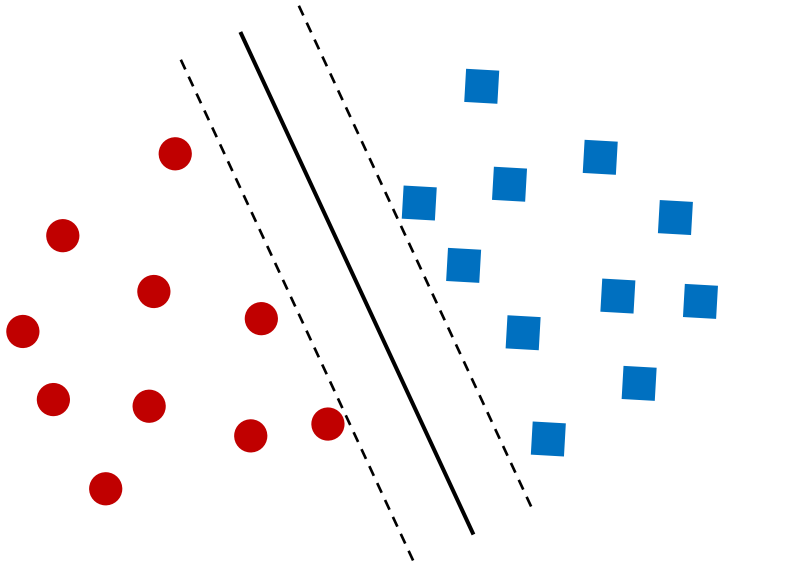
Classification: A Functional Mapping

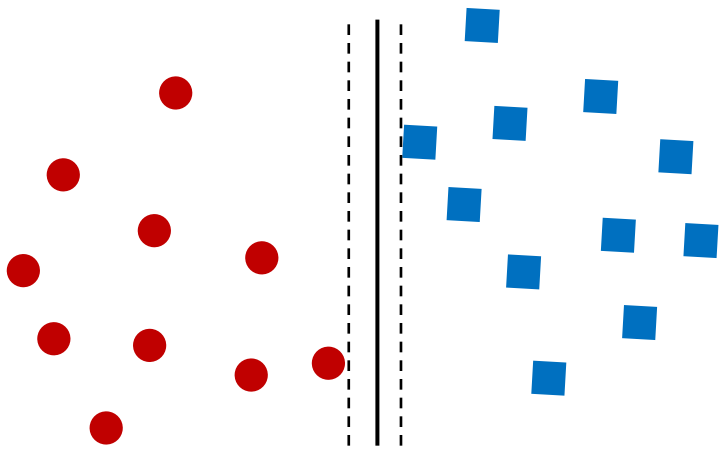
- **The binary classification problem:**
  - E.g., Movie review classification
    - $x_i = (x_1, x_2, x_3, \ldots)$, $y_i = +1$ or -1 (positive, negative)
    - x_1 : # of word "awesome"
    - x_2 : # of word "disappointing"
- Mathematically, $\boldsymbol{x} \in X = \Re^n, y \in Y = \{+1, -1\}$
  - We want to derive a function $f: X \rightarrow Y$
  - which maps input examples to their correct labels

SVM—Support Vector Machines

- linear and nonlinear
  - Vapnik and colleagues (1992)—groundwork from Vapnik & Chervonenkis' statistical learning theory in 1960s
- It uses a nonlinear mapping to transform the original training data into a higher dimension
- With the new dimension, it searches for the linear optimal separating **hyperplane** (i.e., "decision boundary")
- With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane
- SVM finds this hyperplane using **support vectors** ("essential" training tuples) and **margins** (defined by the support vectors)

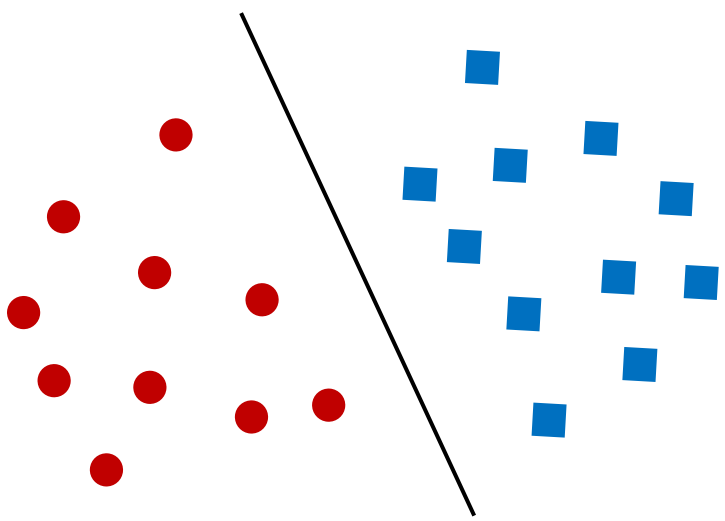
SVM—General Philosophy



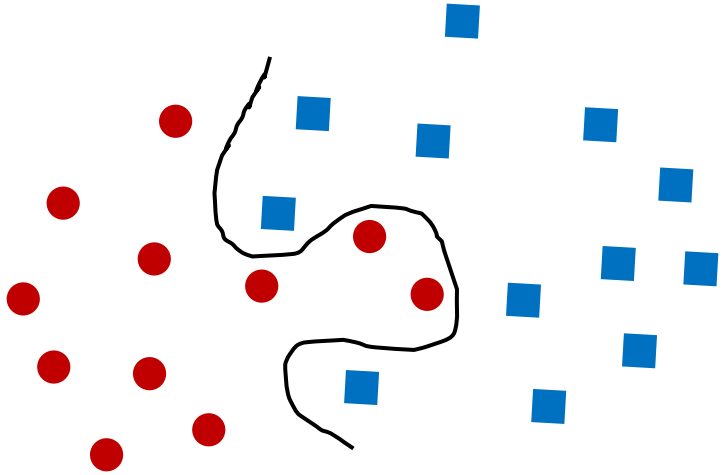


- Learning a max-margin classifier
  - From the infinite set of lines (hyperplanes) separating two classes
  - Find the one which separates two classes with the **largest margin**
  - i.e. a **maximum marginal hyperplane (MMH)**

SVM—When Data Is Linearly Separable



Linearly Separable



Linearly Inseparable

- The simplest case: When data is **linearly separable**
  - Data sets whose classes can be separated exactly by linear decision surfaces are said to be linearly separable

Linear SVM for Linearly Separable Data

- A separating hyperplane can be written as

$$\boldsymbol{w}^T \boldsymbol{x} + b = 0$$

Model parameters to learn

where $w = (w_1, w_2, \ldots, w_n)^T$ is a weight vector and b a scalar (bias)

  - For 2-D, it can be written as: $w_1 x_1 + w_2 x_2 + b = 0$

- The hyperplane defining the sides of the margin:

H_1: $w_0 + w_1 x_1 + w_2 x_2 \geq 1$ for $y_i = +1$, and

H_2: $w_0 + w_1 x_1 + w_2 x_2 \leq -1$ for $y_i = -1$

- Any training tuples that fall on hyperplanes H_1 or H_2 (i.e., the sides defining the margin) are **support vectors**

Linear SVM for Linearly Separable Data

- The distance from any data point $\boldsymbol{x}$ to the separating hyperplane is

$$\text{distance}(ax + by + c = 0, (x_0, y_0)) = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}} \qquad r = \frac{|f(\boldsymbol{x})|}{\|\boldsymbol{w}\|} = \frac{y_i(\boldsymbol{w}^T \boldsymbol{x}_i + w_0)}{\|\boldsymbol{w}\|}$$

- Our objective is to maximize the distance of the closest data point to the hyperplane

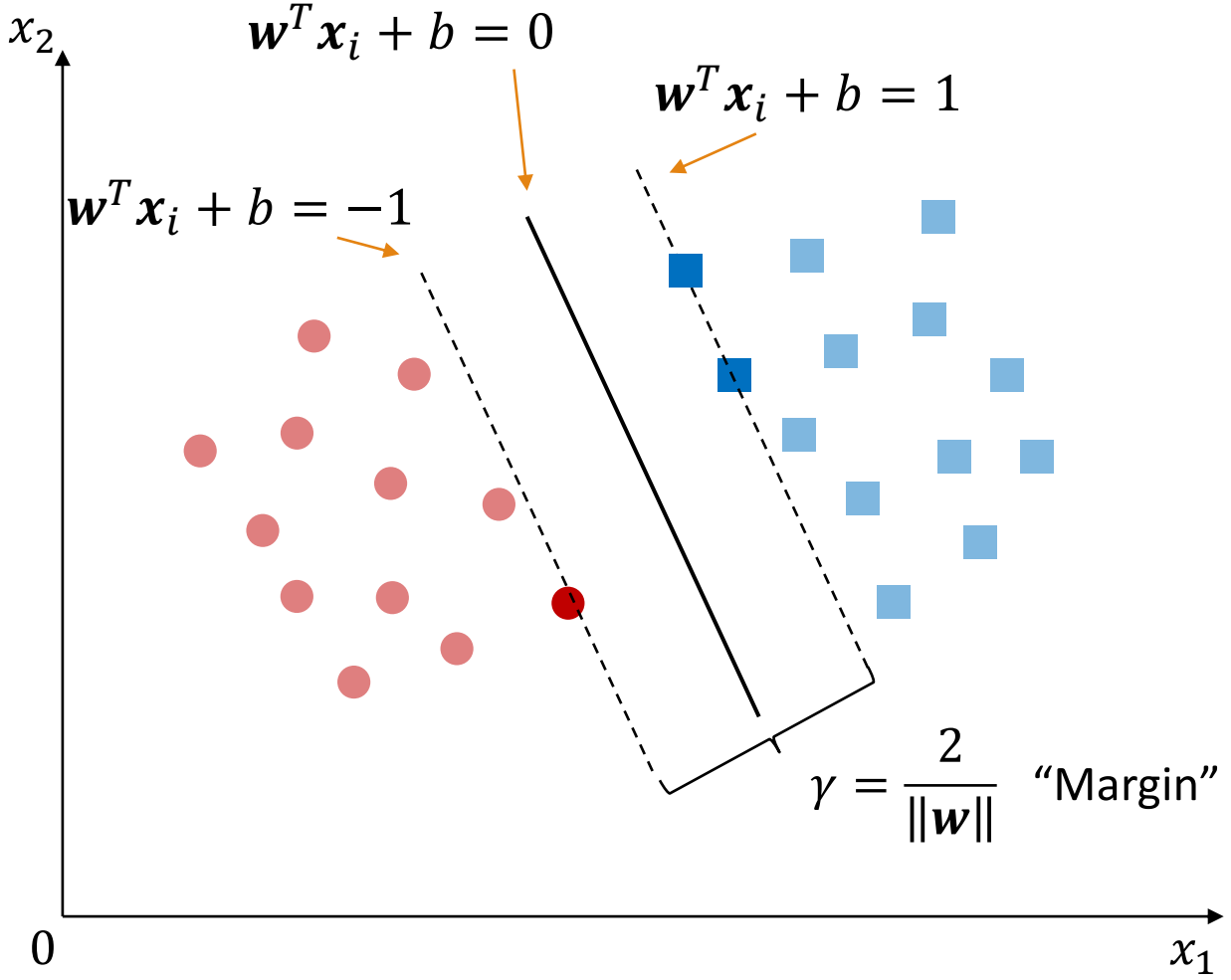
$$\arg\max_{w,b} \left\{ \frac{1}{\|\boldsymbol{w}\|} \min[y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b)] \right\}$$

- This is hard to solve, we shall convert it to an easier problem

$$\arg\min_{w,b} \quad \|\boldsymbol{w}\|^2$$
$$\text{s.t.} \quad y_i(\boldsymbol{w}^T \boldsymbol{x}_i + w_0) \geq 1, \qquad i = 1,2,\dots,n$$

- This is the basic form of SVM, and it can be solved by using *quadratic programming*

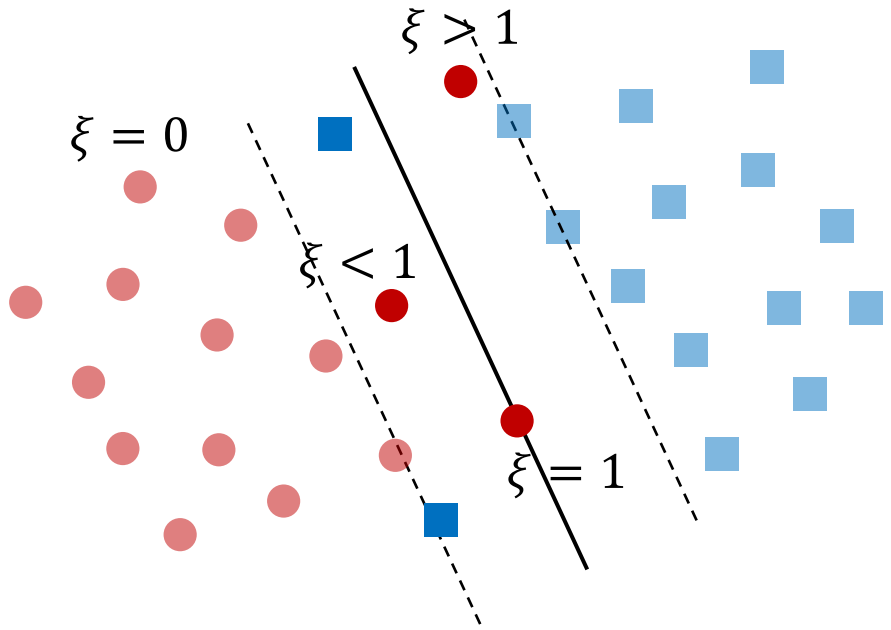
Linear SVM for Linearly Separable Data



- The data points closest to the separating hyperplane are called **support vectors**

SVM for Linearly Inseparable Data

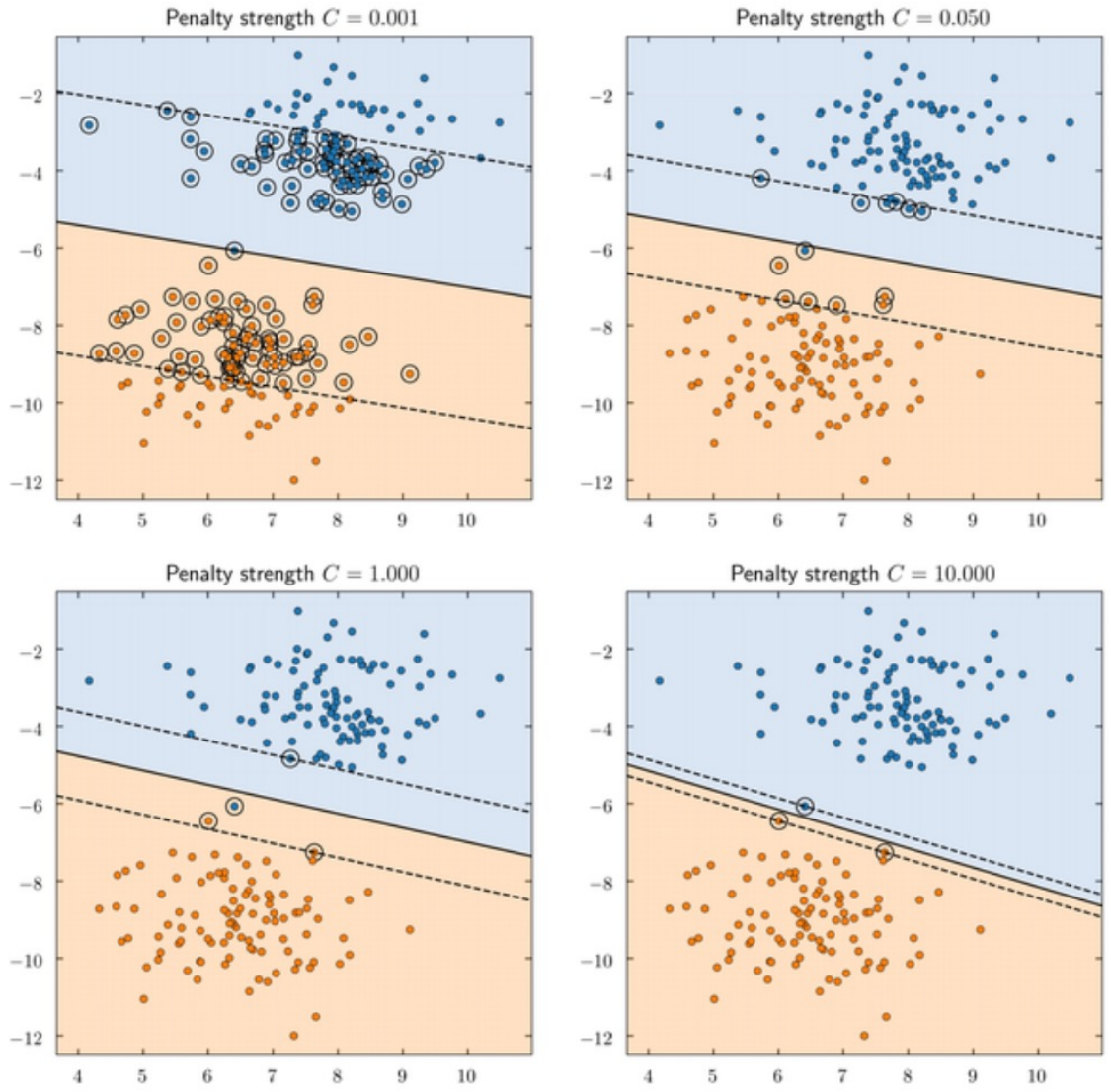
- We allow data points to be on the "wrong side" of the **margin boundary**
- Penalize points on the wrong side according to its distance to the margin boundary
- ξ : slack variable
- C (> 0): Controls the trade-off between the penalty and the margin
- Smaller C: allow more mistake
- Larger C: allow less mistake
- This is the widely used *soft-margin SVM*



$$\arg\min_{w,b,\xi} \quad \|\boldsymbol{w}\|^2 + C\Sigma_i \xi_i$$

$$\text{s.t.} \quad y_i(\boldsymbol{w}^T \boldsymbol{x}_i + w_0) \geq 1, \qquad i = 1,2,\dots,n$$

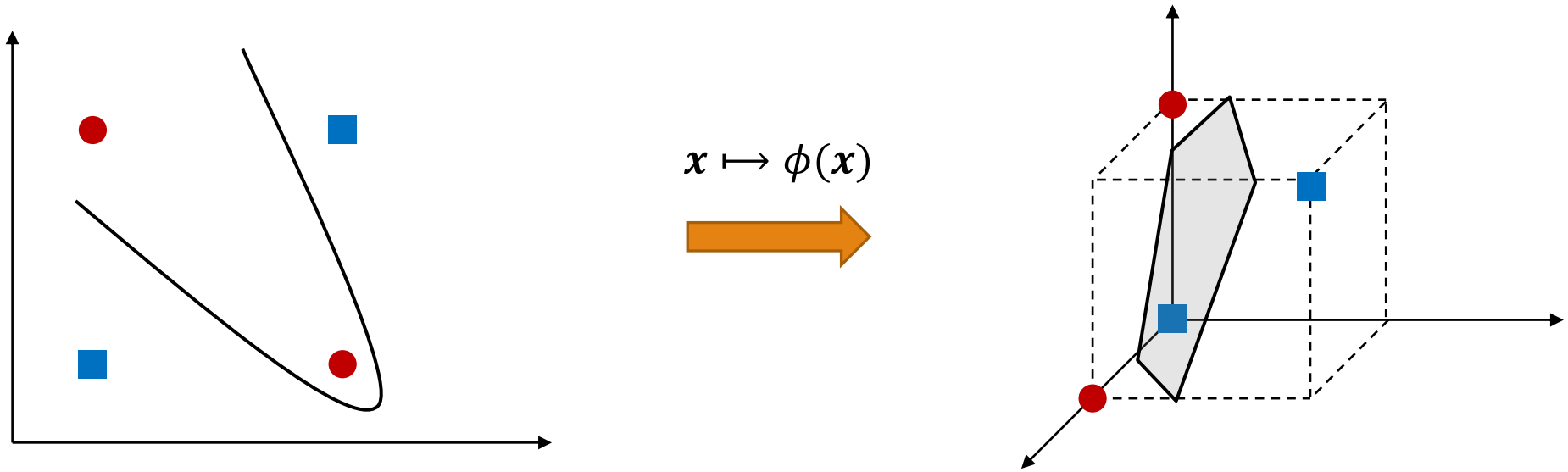
Effect of slack variable



https://www.quora.com/What-is-the-purpose-for-using-slack-variable-in-SVM

SVM for Linearly Inseparable Data

- Alternatively, for linearly inseparable data, we can map them to a higher dimensional space
- We search for a linear separating hyperplane in the new space
  - Example: The XOR problem



Kernel Functions for Nonlinear Classification

- Instead of computing the dot product on the transformed data, it is mathematically equivalent to applying a kernel function $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ to the original data, i.e.,
  - $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)\, \phi(\boldsymbol{x}_j)$
- Typical Kernel Functions

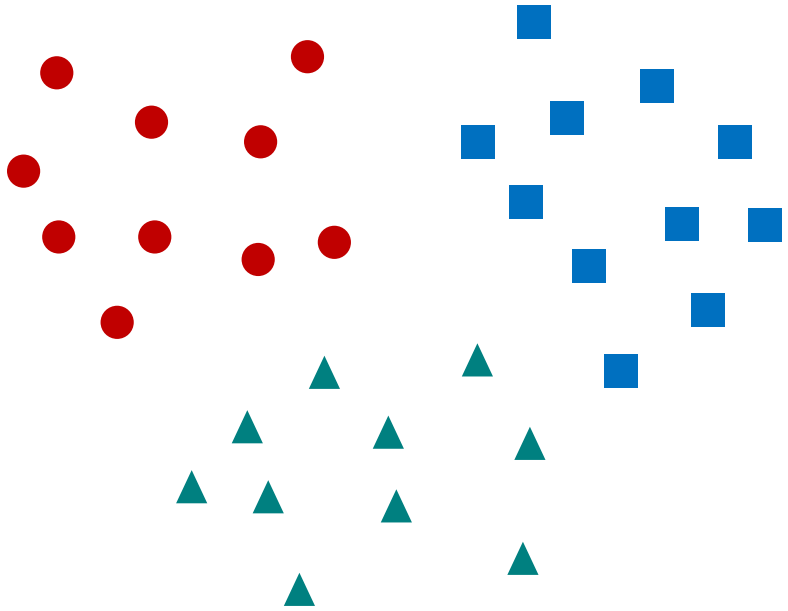
$$\textbf{Polynomial kernel of degree } h: \quad K(\boldsymbol{X_i}, \boldsymbol{X_j}) = (\boldsymbol{X_i} \cdot \boldsymbol{X_j} + 1)^h$$

$$\textbf{Gaussian radial basis function kernel}: \quad K(\boldsymbol{X_i}, \boldsymbol{X_j}) = e^{-\|\boldsymbol{X_i} - \boldsymbol{X_j}\|^2 / 2\sigma^2}$$

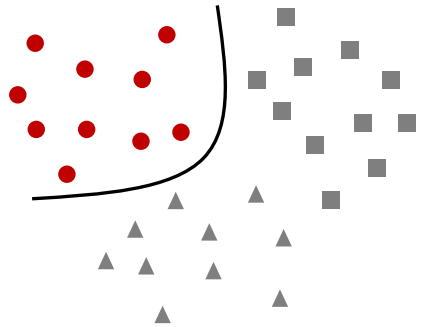
$$\textbf{Sigmoid kernel}: \quad K(\boldsymbol{X_i}, \boldsymbol{X_j}) = \tanh(\kappa \boldsymbol{X_i} \cdot \boldsymbol{X_j} - \delta)$$

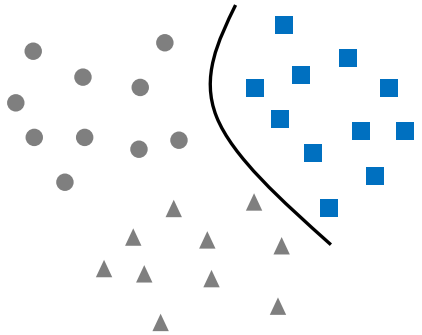
- SVMs can efficiently perform a non-linear classification using kernel functions, implicitly mapping their inputs into high-dimensional feature spaces

Multi-class Classification with SVM

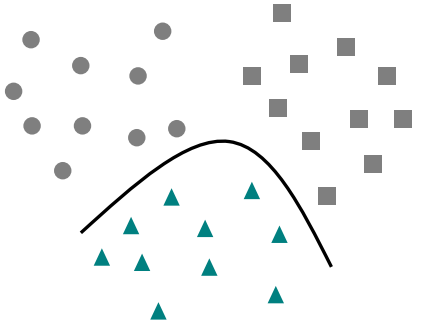


N classes

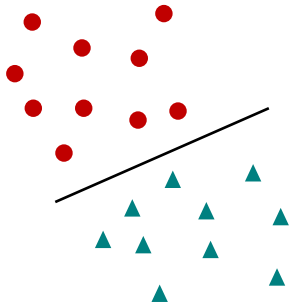


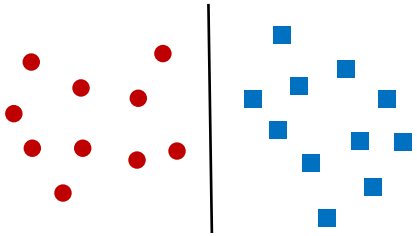


One-vs-Rest

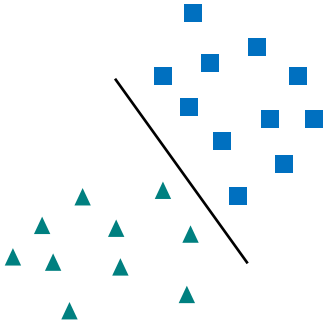


Requires N classifiers





One-vs-One



Requires N(N - 1)/2 classifiers

Is SVM Scalable to Massive Data?

- SVM is effective for high dimensional data
  - The **complexity** of trained classifier is characterized by the # of support vectors rather than the dimensionality of the data
  - The **support vectors** are the essential or critical training examples—they lie closest to the decision boundary
  - Thus, an SVM with a small number of support vectors can have good generalization, even when the dimensionality of the data is high
- Scaling SVMs: # of data objects, training time, and memory usage
  - Linear SVMs are scalable, using SGD
  - Kernel SVMs need to solve quadratic program, can be slow

SVM: Applications

- Features: training can be slow but accuracy is high owing to their ability to model complex nonlinear decision boundaries (margin maximization)
- Used for: classification and numeric prediction
  - SVM can also be used for classifying multiple (> 2) classes and for regression analysis (with additional parameters)
- Applications:
  - handwritten digit recognition, object recognition, speaker identification, benchmarking time-series prediction tests

SVM Recap

- Pros
  - Elegant mathematical formulation, guaranteed global optimal with optimization
  - Trains well on small data sets
  - Flexibility through kernel functions
  - Conformity with semi-supervised training
- Cons
  - Kernel SVMs for large data sets can be slow
  - Representation restricted to (mapped) linear models

Chapter 8. Classification: Advanced Methods

- Bayesian Belief Networks
- Support Vector Machines
- Rule-Based and Pattern-Based Classification 
- Classification with Weak Supervision
- Classification with Rich Data Type
- Summary

Using IF-THEN Rules for Classification

- Represent the knowledge in the form of **IF-THEN** rules

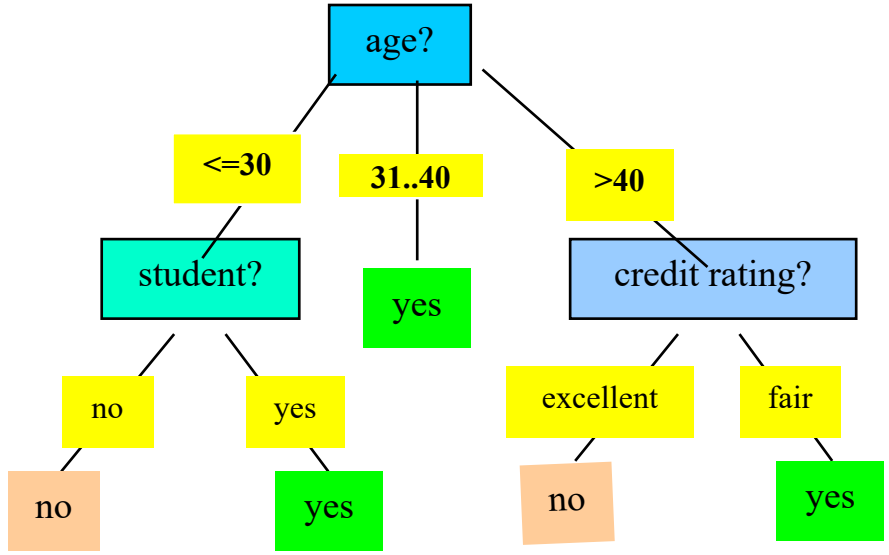
  R_1: IF *age* = youth AND *student* = yes THEN *buys_computer* = yes

- Assessment of a rule: *coverage* and *accuracy*
  - coverage(R_1) = ratio of tuples covered by **the condition of** R_1 (THEN-part is not important for this)
  - accuracy(R_1) = ratio of tuples correctly classified by R_1 in the covered ones (both IF-part and THEN-part counts)

- If more than one rule are triggered, need **conflict resolution**
  - **Size ordering**: assign the highest priority to the triggering rules that has the "toughest" requirement (i.e., with the *most attribute tests*)
  - **Class-based ordering**: decreasing order of *prevalence or misclassification cost per class*
  - **Rule-based ordering** (**decision list**): rules are organized into one long priority list, according to some measure of rule quality or by experts

Rule Extraction from a Decision Tree

- Rules are *easier to understand* than large trees
- One rule is created *for each path* from the root to a leaf
- Each attribute-value pair along a path forms a conjunction: the leaf holds the class prediction
- Rules are mutually exclusive and exhaustive



- Example: Rule extraction from our *buys_computer* decision-tree

IF *age* = young AND *student* = *no* THEN *buys_computer* = *no*

IF *age* = young AND *student* = *yes* THEN *buys_computer* = *yes*

IF *age* = mid-age THEN *buys_computer* = *yes*

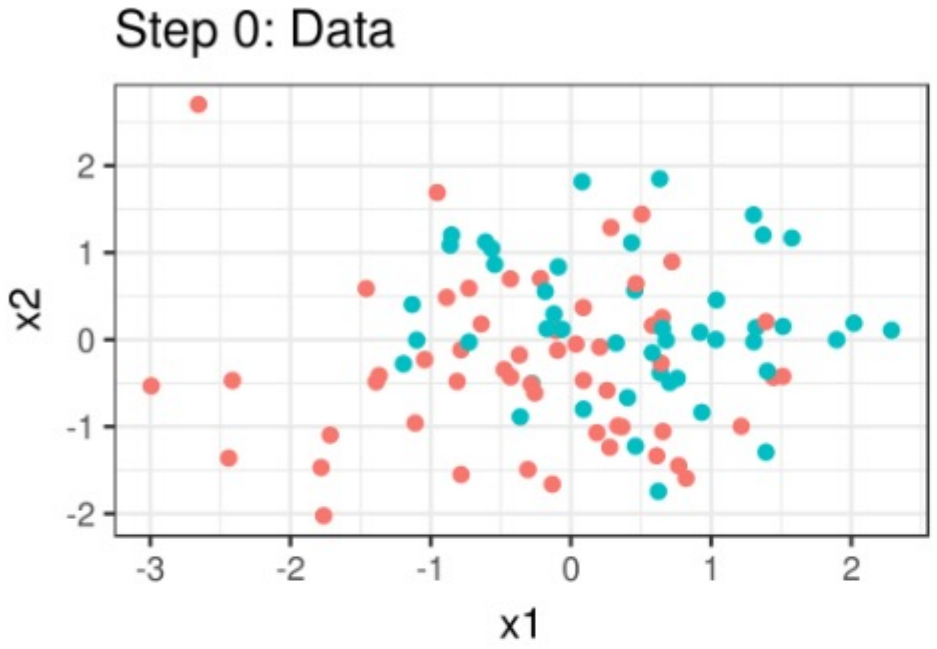
IF *age* = old AND *credit_rating* = *excellent* THEN *buys_computer* = *no*

IF *age* = old AND *credit_rating* = *fair* THEN *buys_computer* = *yes*

Rule Induction: Sequential Covering Method

- Sequential covering algorithm: Extracts rules directly from training data
- Rules are learned *sequentially*, each for a given class C_i will cover many tuples of C_i but none (or few) of the tuples of other classes
- Comp. w. decision-tree induction: learning a set of rules *simultaneously*

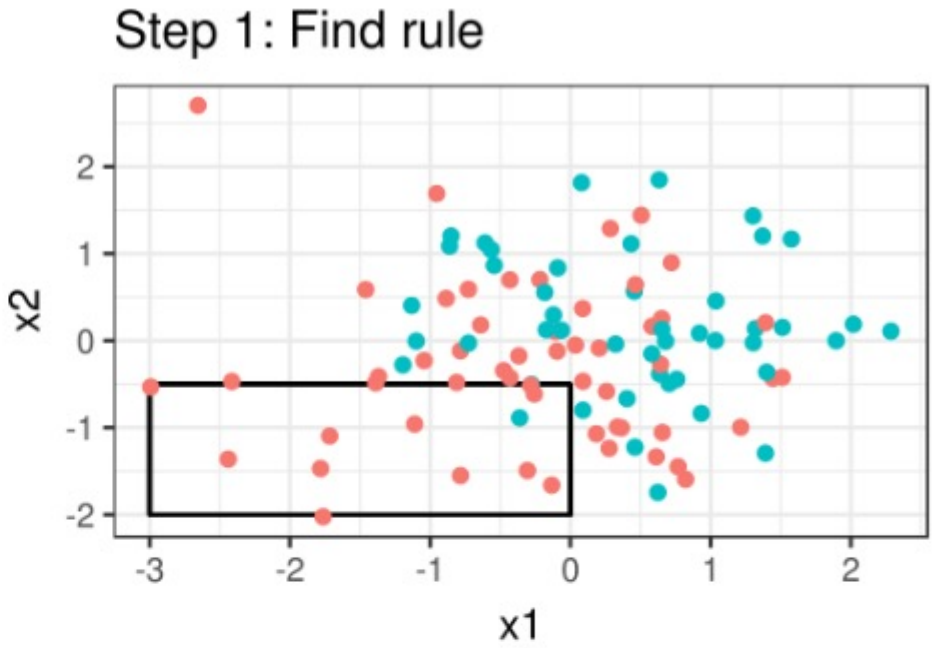
- *Step 0: Start with an empty list of rules.*



Rule Induction: Sequential Covering Method

- Sequential covering algorithm: Extracts rules directly from training data
- Rules are learned *sequentially*, each for a given class C_i will cover many tuples of C_i but none (or few) of the tuples of other classes
- Comp. w. decision-tree induction: learning a set of rules *simultaneously*

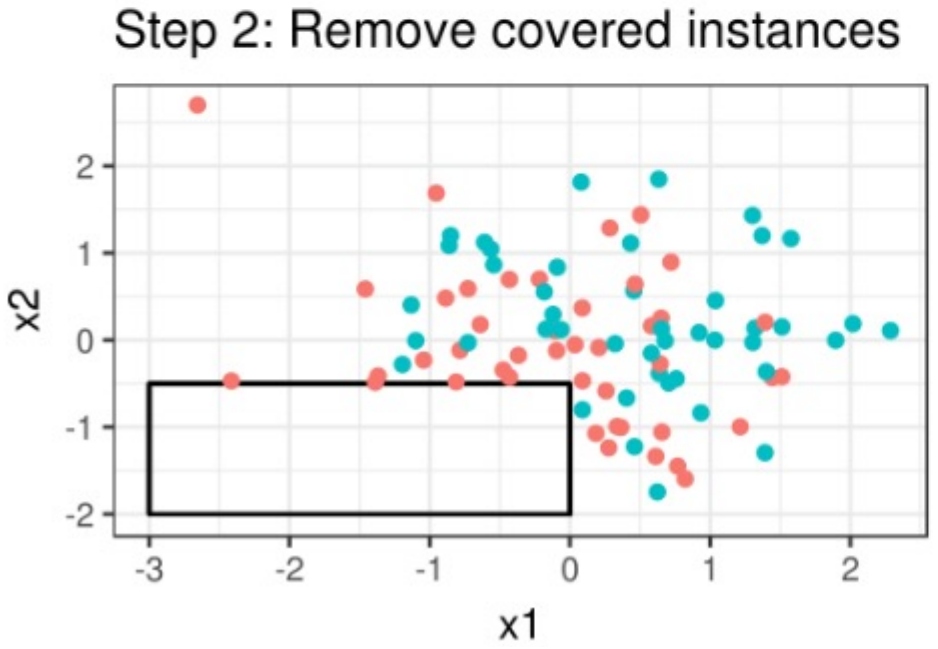
- *Step 1: Learn a rule r.*



Rule Induction: Sequential Covering Method

- Sequential covering algorithm: Extracts rules directly from training data
- Rules are learned *sequentially*, each for a given class C_i will cover many tuples of C_i but none (or few) of the tuples of other classes
- Comp. w. decision-tree induction: learning a set of rules *simultaneously*

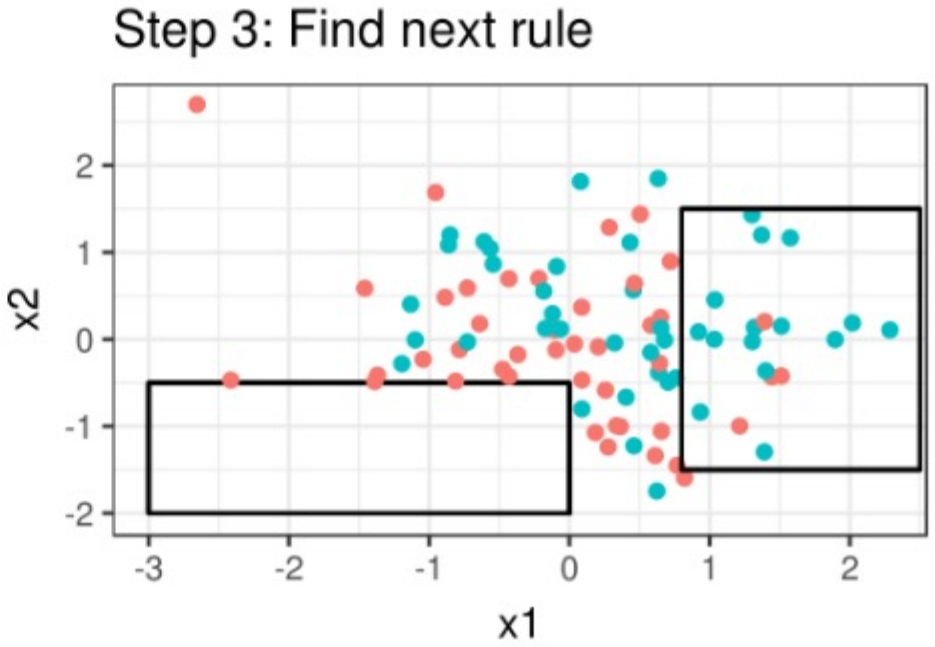
- *Step 2: The tuples covered by the rules are removed.*



Rule Induction: Sequential Covering Method

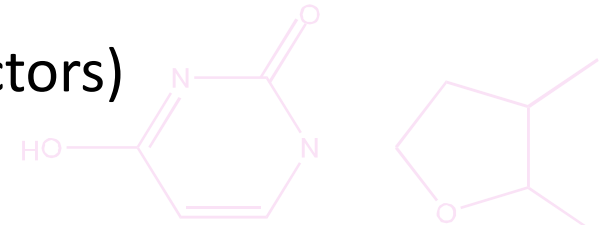
- Sequential covering algorithm: Extracts rules directly from training data
- Rules are learned *sequentially*, each for a given class C_i will cover many tuples of C_i but none (or few) of the tuples of other classes
- Comp. w. decision-tree induction: learning a set of rules *simultaneously*

- *Step 3: Repeat the process* on the remaining tuples until *termination condition*, e.g., when no more training examples or when the quality of a rule returned is below a threshold.



Pattern-Based Classification, Why?

- **Pattern-based classification:** An integration of both themes
- **Why pattern-based classification?**
  - **Feature construction**
    - Higher order; compact; discriminative
    - E.g., single word → phrase (Apple pie, Apple i-pad)
  - **Complex data modeling**
    - Graphs (no predefined feature vectors)
    - Sequences
    - Semi-structured/unstructured Data



Discriminative Frequent Pattern-based Classification

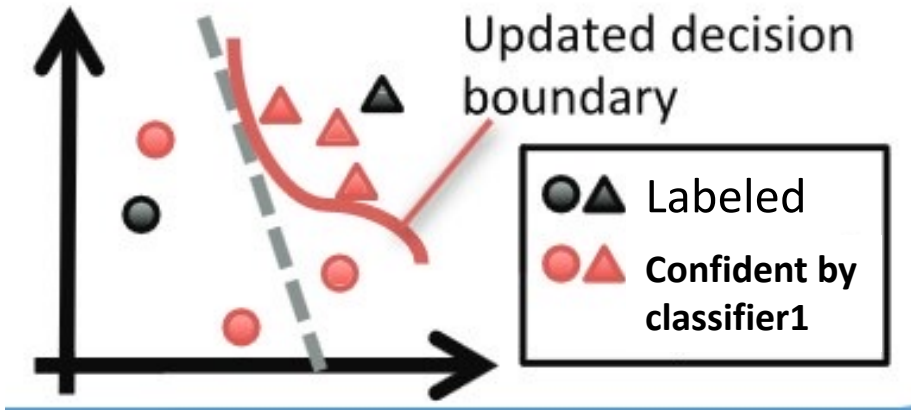
- Consider frequent patterns as combined/complex features
  - Basis for frequent-pattern based classification
- How discriminative are frequent patterns?
  - Not necessarily discriminative, derived based on support, not predictive power
  - Too many patterns, slows down learning
- Framework for discriminative frequent pattern mining
  - Feature generation: Frequent patterns for each class
  - Feature selection: Select set of discriminative patterns as features
    - Based on information gain, scoring methods; remove redundant patterns
  - Learn classifier based on single features and selected frequent patterns
- Direct Discriminative Pattern Mining (DDPMine), extension DPClass
  - Combine feature generation and selection: using FP-tree, information gain

Chapter 8. Classification: Advanced Methods

- Bayesian Belief Networks
- Support Vector Machines
- Rule-Based and Pattern-Based Classification
- Classification with Weak Supervision 
- Classification with Rich Data Types
- Summary

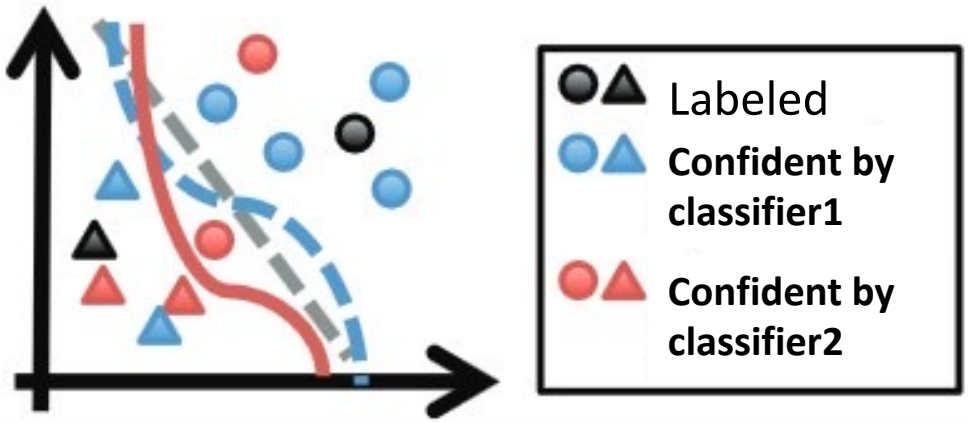
Semi-Supervised Classification

- Semi-supervised: Uses labeled and unlabeled data to build a classifier
- Self-training
  - 1. Build a classifier using the labeled data
  - 2. Use it to label the unlabeled data, and those with the most confident label prediction are added to the set of labeled data
  - 3. Repeat the step 1 and 2
  - Adv.: easy to understand; Disadv.: may reinforce errors



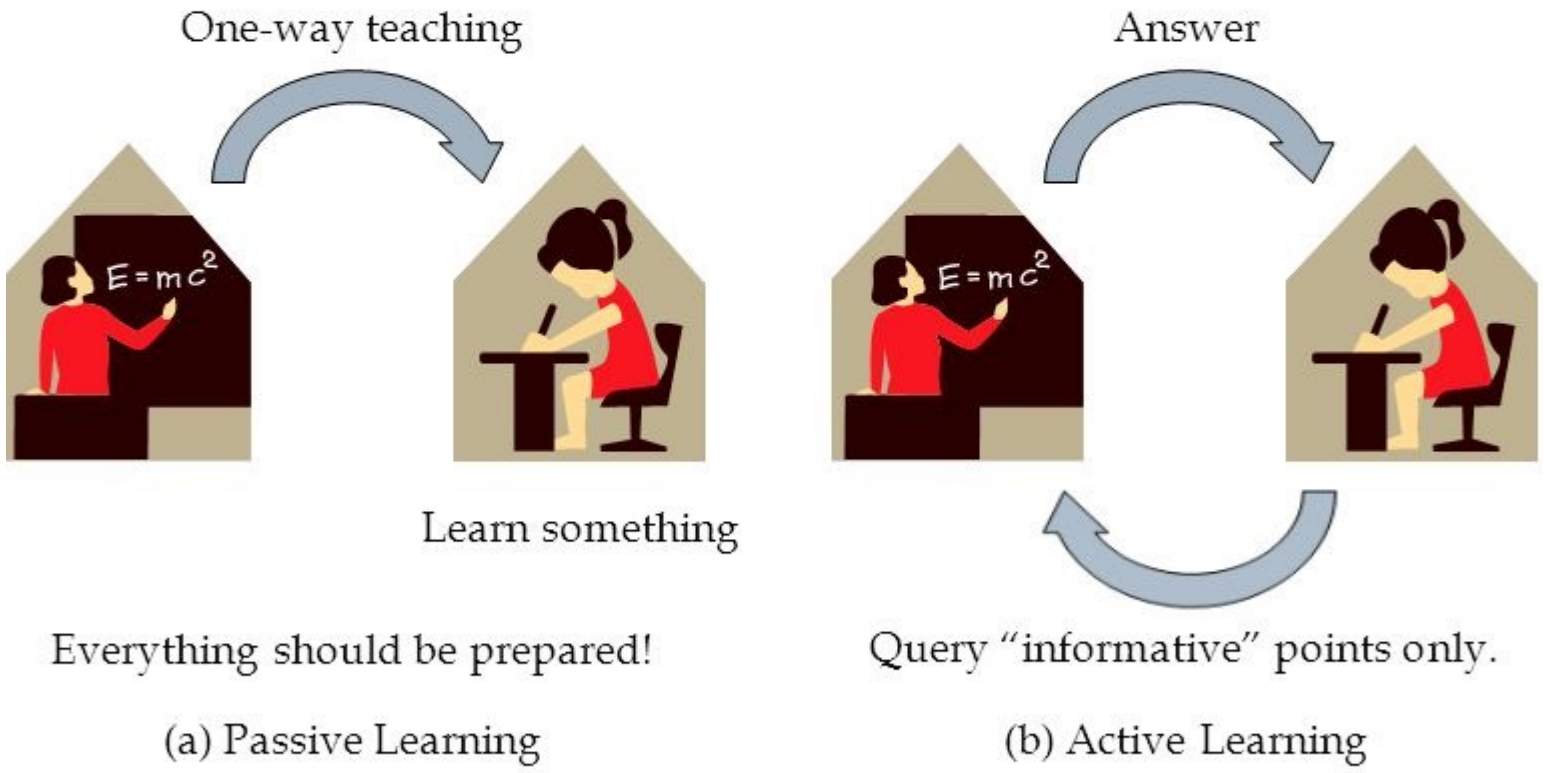
Semi-Supervised Classification

- Semi-supervised: Uses labeled and unlabeled data to build a classifier
- 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



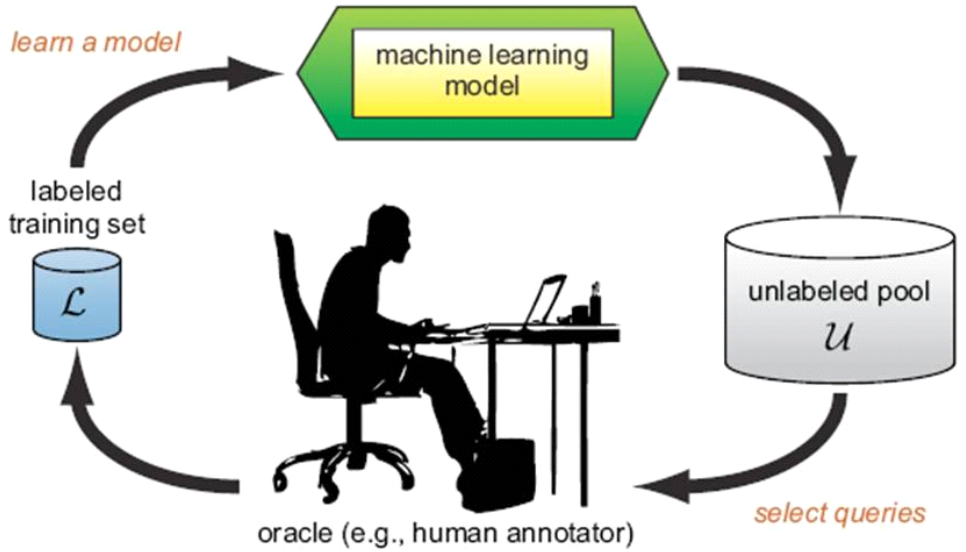
Active Learning

- A special case of semi-supervised learning
- Active learner: Interactively query teachers (oracle) for labels of "informative" data



Active Learning

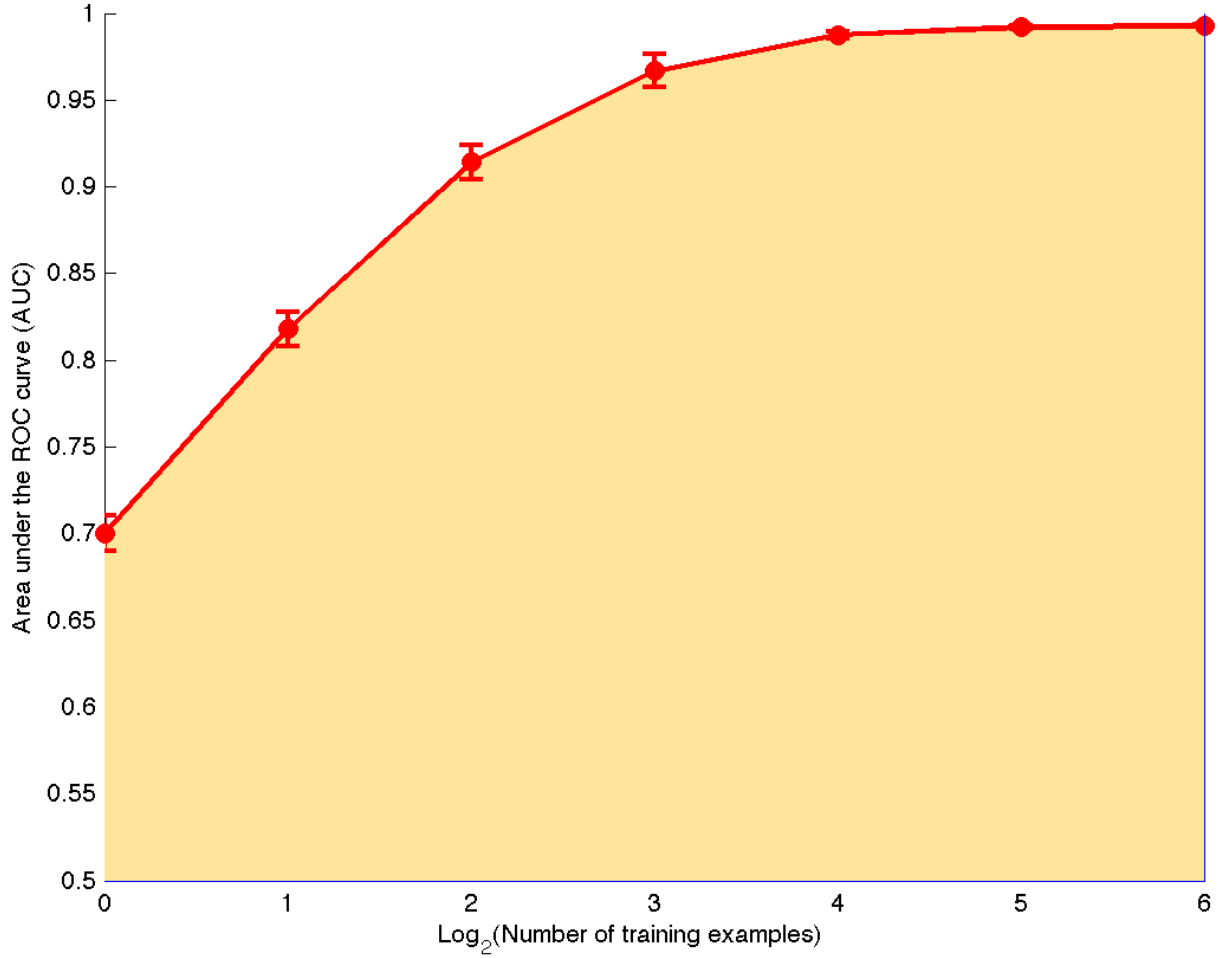
- Pool-based approach: Uses a pool of unlabeled data
  - L: a small subset of D is labeled, U: a pool of unlabeled data in D
  - Use a query function to carefully select one or more tuples from U and request labels from an oracle (a human annotator)
  - The newly labeled samples are added to L, and learn a model
  - Goal: **Achieve high accuracy using as few labeled data as possible**



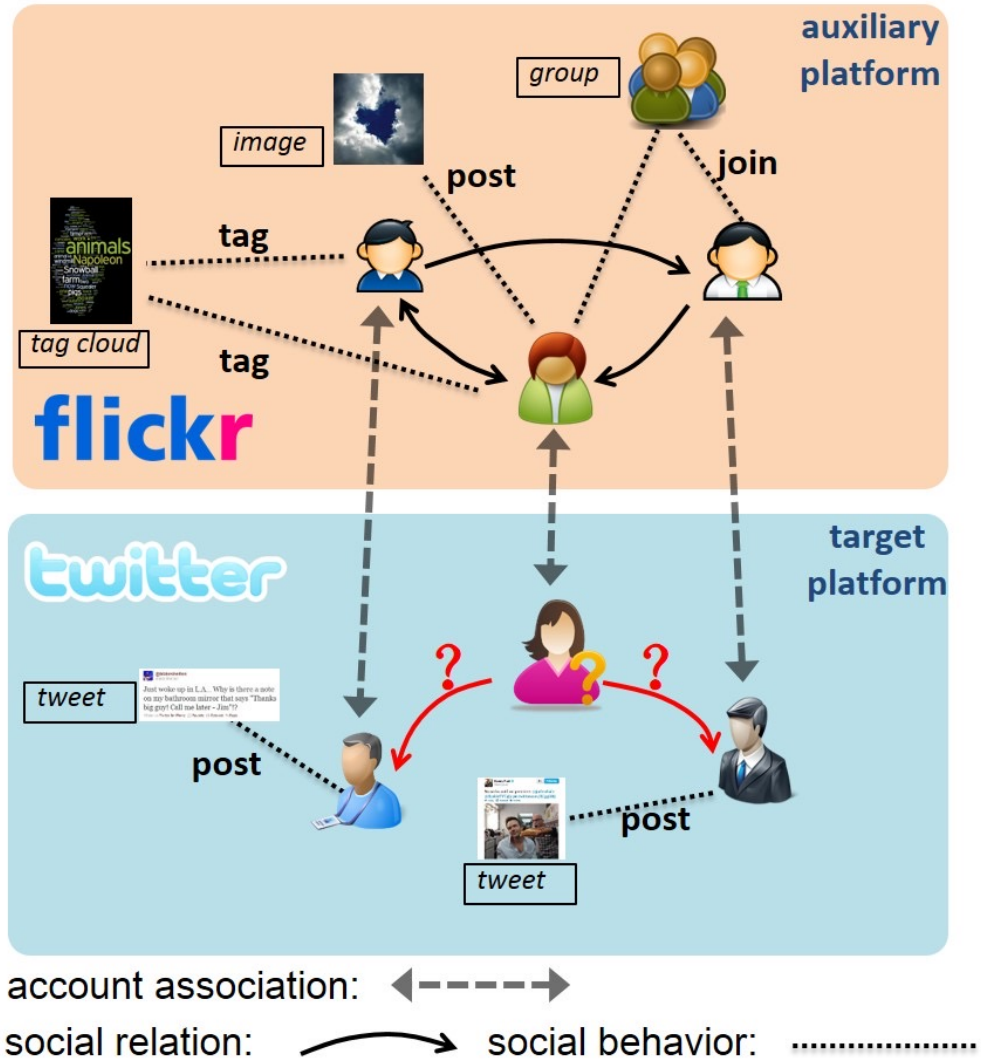
One good choice: unlabeled point closest to the current decision boundary

Active Learning

- Evaluated using *learning curves*: Accuracy as a function of the number of instances queried (# of tuples to be queried should be small)



Transfer Learning



- **Traditional learning**: Build a new classifier for each new task
- **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)
- **Example: Cross-platform friend recommendation** [1]
  - Users' social relation and behavior in one platform(flickr) offers important knowledge about social interest in another platform(Twitter)

[1]http://nlpr-web.ia.ac.cn/mmc/homepage/myan/Project_YanMing/ming_ICME2013/material/ICME2013Final.pdf

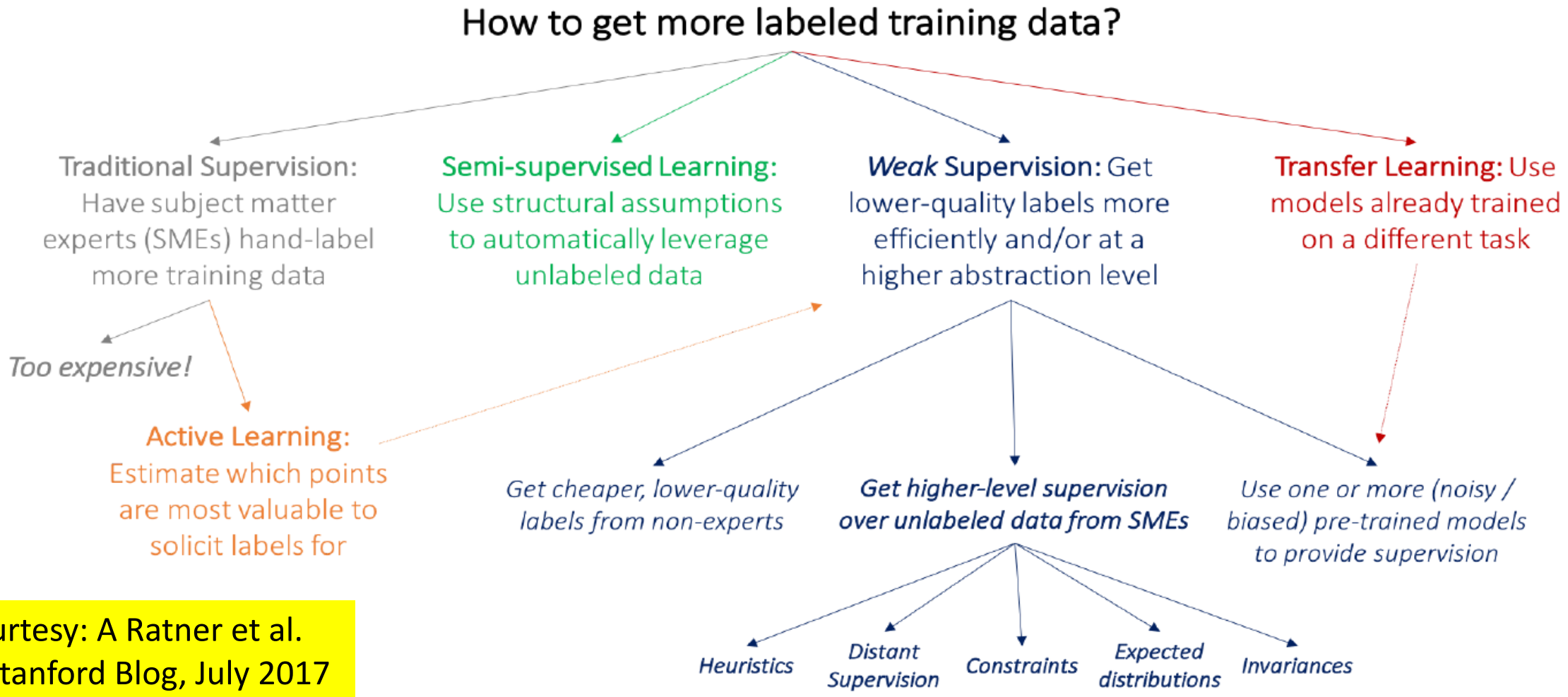
Weak Supervision

- Overcome the training data bottleneck
  - Leverage higher-level and/or noisier input from experts
- Sources of cheaply and efficiently provided weak labels:
  - 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)

Weak Supervision

- These weak label distributions could take **many forms**
  - Weak Labels from crowd workers, output of heuristic rules, knowledge base(e.g., wikipedia), or the output of other classifiers, etc.
  - 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: A 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.

Chapter 8. Classification: Advanced Methods

- Bayesian Belief Networks
- Support Vector Machines
- Rule-Based and Pattern-Based Classification
- Classification with Weak Supervision
- Classification with Rich Data Types 
- Summary

Classification with Rich Data Types

- Input need not be a feature/attribute vectors, Rich data types
  - Streams, time-series, multi-variate extensions
  - Sequences, e.g., amino-acids, product purchase
  - Graphs, e.g., social networks, power-grid, biological networks
- Approaches: Using suitable representations
  - Extract features, e.g., frequent patterns, embeddings, encoders, etc.
  - Implicit, based on Kernel methods, learning representations
- Direct mappings between rich data types
  - Sequence to sequence, graph to vectors, etc.

Chapter 8. Classification: Advanced Methods

- Feature Selection and Engineering
- Bayesian Belief Networks
- Support Vector Machines
- Rule-Based and Pattern-Based Classification
- Classification with Weak Supervision
- Classification with Rich Data Type
- Summary

Summary

- Deterministic vs. generative
- Parametric vs. non-parametric
- Lazy vs. eager
- Linear vs. non-linear
- Performance
  - Bias vs. balance
  - Robustness
  - Interpretability
  - Scalability
- Feature engineering

References (1)

- C. M. Bishop, Neural Networks for Pattern Recognition. Oxford University Press, 1995
- C. M. Bishop, Pattern Recognition and Machine Learning. Springer, 2006
- L. Breiman, J. Friedman, R. Olshen, and C. Stone. Classification and Regression Trees. Wadsworth International Group, 1984
- C. J. C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2): 121-168, 1998
- N. Cristianini and J. Shawe-Taylor, Introduction to Support Vector Machines and Other Kernel-Based Learning Methods, Cambridge University Press, 2000
- H. Yu, J. Yang, and J. Han. Classifying large data sets using SVM with hierarchical clusters. KDD'03
- A. J. Dobson. An Introduction to Generalized Linear Models. Chapman & Hall, 1990
- R. O. Duda, P. E. Hart, and D. G. Stork. Pattern Classification, 2ed. John Wiley, 2001
- T. Hastie, R. Tibshirani, and J. Friedman. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer-Verlag, 2001
- S. Haykin, Neural Networks and Learning Machines, Prentice Hall, 2008
- D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. Machine Learning, 1995

References (2): Rule and Pattern-Based Classification

- H. Cheng, X. Yan, J. Han & C.-W. Hsu, Discriminative Frequent Pattern Analysis for Effective Classification, ICDE'07
- H. Cheng, X. Yan, J. Han & P. S. Yu, Direct Discriminative Pattern Mining for Effective Classification, ICDE'08
- W. Cohen. Fast effective rule induction. ICML'95
- G. Cong, K. Tan, A. Tung & X. Xu. Mining Top-k Covering Rule Groups for Gene Expression Data, SIGMOD'05
- M. Deshpande, M. Kuramochi, N. Wale & G. Karypis. Frequent Substructure-based Approaches for Classifying Chemical Compounds, TKDE'05
- G. Dong & J. Li. Efficient Mining of Emerging Patterns: Discovering Trends and Differences, KDD'99
- W. Fan, K. Zhang, H. Cheng, J. Gao, X. Yan, J. Han, P. S. Yu & O. Verscheure. Direct Mining of Discriminative and Essential Graphical and Itemset Features via Model-based Search Tree, KDD'08
- W. Li, J. Han & J. Pei. CMAR: Accurate and Efficient Classification based on Multiple Class-association Rules, ICDM'01
- B. Liu, W. Hsu & Y. Ma. Integrating Classification and Association Rule Mining, KDD'98
- J. R. Quinlan and R. M. Cameron-Jones. FOIL: A midterm report. ECML'93
- Jingbo Shang, Wenzhu Tong, Jian Peng, and Jiawei Han, "DPClass: An Effective but Concise Discriminative Patterns-Based Classification Framework", SDM'16
- J. Wang and G. Karypis. HARMONY: Efficiently Mining the Best Rules for Classification, SDM'05
- X. Yin & J. Han. CPAR: Classification Based on Predictive Association Rules, SDM'03