Data Mining

Lesson 9

Support Vector Machines

MSc in Computer Science

University of New York Tirana

Assoc. Prof. Dr. Marenglen Biba

Data Mining: Content

- Introduction to data mining and machine learning
- Inductive learning
- Decision trees
- Rule induction
- Instance-based learning
- Bayesian learning
- Neural networks
- Support vector machines
- Other data mining models
- Engineering data mining tasks

Outline

- SVMs
- Association rule mining

- Lab session
  - SVM and Ass. Rule mining in Weka
  - SVM and Ass. Rule mining in Oracle

Numeric prediction: Linear regression

- The methods we have been looking at for decision trees and rules work most naturally with nominal attributes.

- They can be extended to numeric attributes either by incorporating numeric-value tests directly into the decision tree or rule induction scheme, or by prediscretizing numeric attributes into nominal ones.

- However, there are methods that work most naturally with numeric attributes

Linear regression

- When the outcome, or class, is numeric, and all the attributes are numeric, linear regression is a natural technique to consider.

- The idea is to express the class as a linear combination of the attributes, with predetermined weights:

$$X = w_0 + w_1 a_1 + w_2 a_2 + \ldots + w_k a_k$$

- where x is the class; a_1, a_2, . . ., a_k are the attribute values and w_0, w_1, . . ., w_k are weights.

- The weights are calculated from the training data.

Linear regression

- The predicted value for the first instance's class can be written as:

$$w_0 a_0^{(1)} + w_1 a_1^{(1)} + w_2 a_2^{(1)} + \ldots + w_k a_k^{(1)} = \sum_{j=0}^{k} w_j a_j^{(1)}.$$

- This is the predicted, not the actual, value for the first instance's class.
- Of interest is the difference between the predicted and the actual values.
- The method of linear regression is to choose the coefficients w_j — there are $k + 1$ of them — to minimize the sum of the squares of these differences over all the training instances.

Linear regression

- Suppose there are *n* training instances; denote the *i*th one with a superscript (*i*). Then the sum of the squares of the differences is:

$$\sum_{i=1}^{n}\left(x^{(i)}-\sum_{j=0}^{k}w_j a_j^{(i)}\right)^2$$

- where the expression inside the parentheses is the difference between the *i*th instance's actual class and its predicted class.

- This sum of squares is what we have to minimize by choosing the coefficients appropriately.

SVMs

- Simple linear models can be used for classification in situations where all attributes are numeric.

- Their biggest disadvantage is that they can only represent linear boundaries between classes, which makes them too simple for many practical applications.

- *Support vector machines use linear models to implement nonlinear class boundaries.*

Who invented SVMs?

- The original SVM algorithm was invented by Vladimir N. Vapnik.
- In 2002 he joined NEC Laboratories in Princeton, New Jersey, where he currently works in the Machine Learning group.
- Long career, check this:
  - http://en.wikipedia.org/wiki/Vladimir_N._Vapnik

SVMs

- Although it is a widely used term, *support vector machines* is something of a misnomer: these are algorithms, not machines.
- How do SVMs work: *transform the input using a nonlinear mapping; in other words, transform the instance space into a new space.*
- With a nonlinear mapping, a straight line in the new space doesn't look straight in the original instance space.
- A linear model constructed in the new space can represent a nonlinear decision boundary in the original space.

Problems for SVMs: computational complexity and overfitting

- Problems arise with this procedure because of the large number of coefficients introduced by the transformation in any realistic setting.
- The first problem is computational complexity.

- The second problem is one of principle: overfitting.
- If the number of coefficients is large relative to the number of training instances, the resulting model will be "too nonlinear" — it will overfit the training data.
- There are just too many parameters in the model!!!

The maximum margin hyperplane

- SVMs solve both problems of computational complexity and overfitting.
- They are based on an algorithm that finds a special kind of linear model: the *maximum margin hyperplane.*
- We already know what a hyperplane is — It's just another word for a linear model.
- To visualize a maximum margin hyperplane, imagine a two-class dataset whose classes are linearly separable; that is, there is a hyperplane in instance space that classifies all training instances correctly.
- *The maximum margin hyperplane is the one that gives the greatest separation between the classes — it comes no closer to either than it has to.*

The maximum margin hyperplane

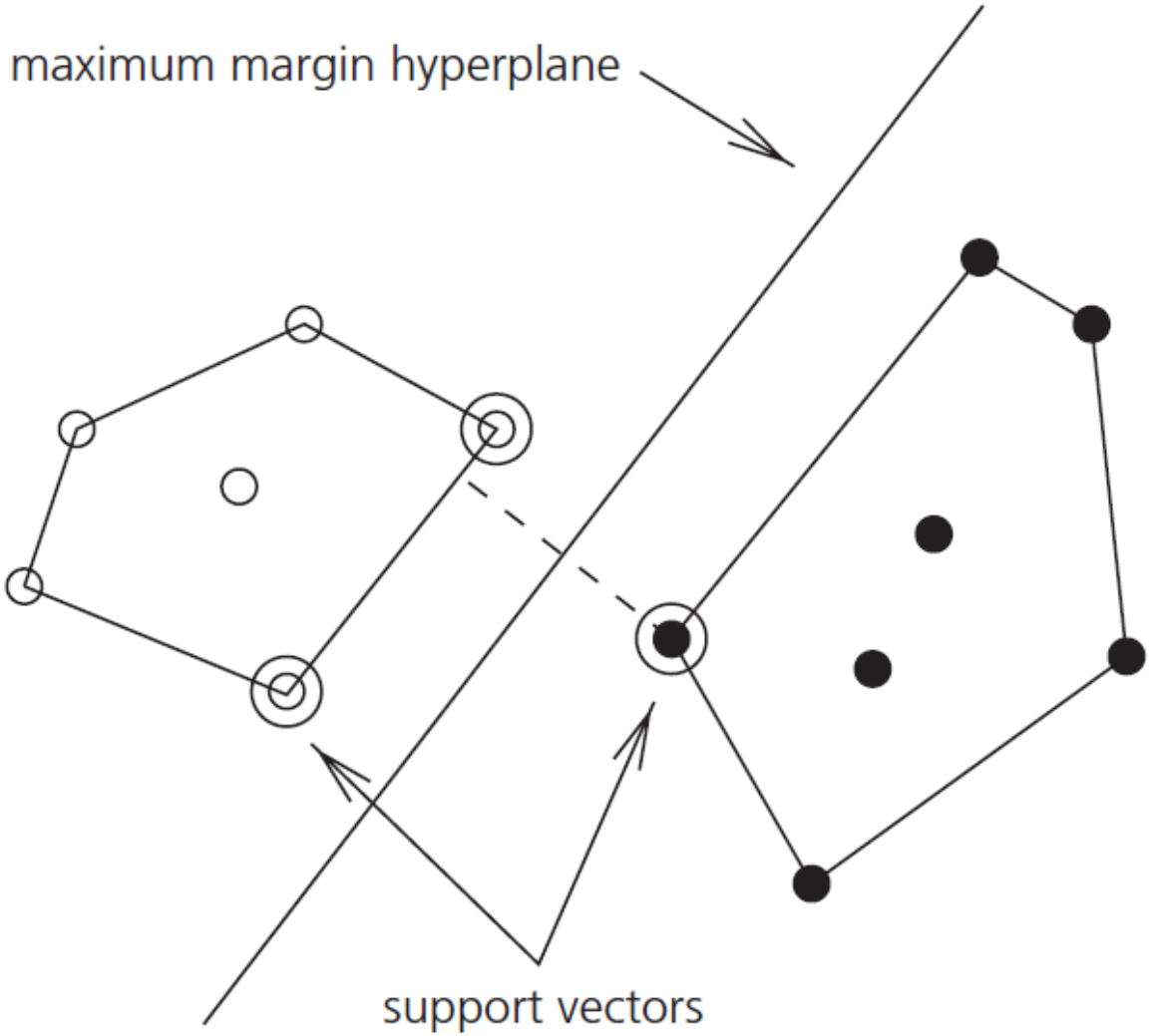

Figure 6.8 A maximum margin hyperplane.

Support vectors

- The instances that are closest to the maximum margin hyperplane — the ones with minimum distance to it— are called *support vectors.*
- There is always at least one support vector for each class, and often there are more.
- The important thing is that the set of support vectors uniquely defines the maximum margin hyperplane for the learning problem.
- Given the support vectors for the two classes, we can easily construct the maximum margin hyperplane.
- All other training instances are irrelevant — they can be deleted without changing the position and orientation of the hyperplane.

Finding the support vectors

- It turns out that finding the support vectors belongs to a standard class of optimization problems known as *constrained quadratic optimization.*

- There are off-the-shelf software packages for solving these problems (see Fletcher 1987 for a comprehensive and practical account of solution methods).

- However, the computational complexity can be reduced, and learning can be accelerated, if special-purpose algorithms for training support vector machines are applied — but the details of these algorithms lie beyond the scope of this course.

Overfitting and SVMs

- With support vectors, overfitting is unlikely to occur.
- The reason is that it is inevitably associated with instability: changing one or two instance vectors will make significant changes to large sections of the decision boundary.
- But the maximum margin hyperplane is relatively stable: it only moves if training instances are added or deleted that are support vectors — and this is true even in the high-dimensional space spanned by the nonlinear transformation.
- *Overfitting is caused by too much flexibility in the decision boundary.*
- The support vectors are global representatives of the whole set of training points, and there are usually few of them, which gives little flexibility.
- Thus overfitting is unlikely to occur.

SVMs and computational complexity

- What about computational complexity? This is still a problem.
- Suppose that the transformed space is a high-dimensional one so that the transformed support vectors and test instance have many components.
- In SVM, every time an instance is classified its dot product with all support vectors must be calculated.
- *In the high-dimensional space produced by the nonlinear mapping this is rather expensive*.

SVMs and computational complexity

- Fortunately, it turns out that it is possible to calculate the dot product *before* the nonlinear mapping is performed, on the original attribute set.

- This is performed through a Kernel Function.

Kernels

- The function $(\mathbf{x}*\mathbf{y})^n$, which computes the dot product of two vectors *x* and *y* and raises the result to the power *n,* is called a *polynomial kernel.*

- Other kernel functions can be used instead to implement different nonlinear mappings.

- Two that are often suggested are the: *radial basis function (RBF) kernel* and the *sigmoid kernel.*

- Which one produces the best results depends on the application, although the differences are rarely large in practice.

Accuracy of SVMs

- We should mention that compared with other methods such as decision tree learners, even the fastest training algorithms for support vector machines are slow when applied in the nonlinear setting.

- On the other hand, they often produce very accurate classifiers because subtle and complex decision boundaries can be obtained.

Readings for this part

- Data Mining. Witten and Frank
  - Sections 4.6 and 6.3.

Data Mining: Content

- Introduction to data mining and machine learning
- Inductive learning
- Decision trees
- Rule induction
- Instance-based learning
- Bayesian learning
- Neural networks
- Support vector machines
- Other data mining models
- Engineering data mining tasks

Other machine learning models

- Association rules

Association rules: intro

- Association rules are like classification rules.

- But any attribute can occur on the right-hand side with any possible value.

- A single association rule often predicts the value of more than one attribute.

Association rules: intro

- To find such rules, you would have to execute the rule-induction procedure once for every possible *combination* of attributes, with every possible combination of values, on the right-hand side.
- That would result in an enormous number of association rules, which would then have to be pruned down on the basis of their *coverage* (the number of instances that they predict correctly) and their *accuracy* (the same number expressed as a proportion of the number of instances to which the rule applies).
- This approach is quite infeasible.
- Note that, what we are calling *coverage* is often called *support* and what we are calling *accuracy* is often called *confidence.*

Item sets

- We capitalize on the fact that we are only interested in association rules with high coverage.
- We ignore, for the moment, the distinction between the left- and right-hand sides of a rule and seek combinations of attribute–value pairs that have a prespecified minimum coverage.
- These are called *item sets:* an attribute–value pair is an *item.*
- The terminology derives from market basket analysis, in which the items are articles in your shopping cart and the supermarket manager is looking for associations among these purchases.

Table 4.10 Item sets for the weather data with coverage 2 or greater.

	One-item sets	Two-item sets	Three-item sets	Four-item sets
1	outlook = sunny (5)	outlook = sunny temperature = mild (2)	outlook = sunny temperature = hot humidity = high (2)	outlook = sunny temperature = hot humidity = high play = no (2)
2	outlook = overcast (4)	outlook = sunny temperature = hot (2)	outlook = sunny temperature = hot play = no (2)	outlook = sunny humidity = high windy = false play = no (2)
3	outlook = rainy (5)	outlook = sunny humidity = normal (2)	outlook = sunny humidity = normal play = yes (2)	outlook = overcast temperature = hot windy = false play = yes (2)
4	temperature = cool (4)	outlook = sunny humidity = high (3)	outlook = sunny humidity = high windy = false (2)	outlook = rainy temperature = mild windy = false play = yes (2)
5	temperature = mild (6)	outlook = sunny windy = true (2)	outlook = sunny humidity = high play = no (3)	outlook = rainy humidity = normal windy = false play = yes (2)
6	temperature = hot (4)	outlook = sunny windy = false (3)	outlook = sunny windy = false play = no (2)	temperature = cool humidity = normal windy = false play = yes (2)

Item sets

Table 4.10 (continued)

One-item sets	Two-item sets	Three-item sets	Four-item sets
. . .	. . .	. . .	
38	humidity = normal windy = false (4)	humidity = normal windy = false play = yes (4)	
39	humidity = normal play = yes (6)	humidity = high windy = false play = no (2)	
40	humidity = high windy = true (3)		
. . .	. . .		
47	windy = false play = no (2)		

Potential rules

- Once all item sets with the required coverage have been generated, the next step is to turn each into a rule, or set of rules, with at least the specified minimum accuracy.
- Some item sets will produce more than one rule; others will produce none. For example, there is one three-item set with a coverage of 4 (row 38 of Table 4.10): humidity = normal, windy = false, play = yes

```
If humidity = normal and windy = false then play = yes          4/4
If humidity = normal and play = yes then windy = false          4/6
If windy = false and play = yes then humidity = normal          4/6
If humidity = normal then windy = false and play = yes          4/7
If windy = false then humidity = normal and play = yes          4/8
If play = yes then humidity = normal and windy = false          4/9
If - then humidity = normal and windy = false and play = yes    4/12
```

- The figures at the right show the number of instances for which all three conditions are true—that is, the coverage — divided by the number of instances for which the conditions in the antecedent are true. Interpreted as a fraction, they represent the proportion of instances on which the rule is correct — that is, its accuracy.
- Assuming that the minimum specified accuracy is 100%, only the first of these rules will make it into the final rule set.

Rules with 100% accuracy

Table 4.11 Association rules for the weather data.

	Association rule			Coverage	Accuracy
1	humidity = normal windy = false	⇒	play = yes	4	100%
2	temperature = cool	⇒	humidity = normal	4	100%
3	outlook = overcast	⇒	play = yes	4	100%
4	temperature = cool play = yes	⇒	humidity = normal	3	100%
5	outlook = rainy windy = false	⇒	play = yes	3	100%
6	outlook = rainy play = yes	⇒	windy = false	3	100%
7	outlook = sunny humidity = high	⇒	play = no	3	100%
8	outlook = sunny play = no	⇒	humidity = high	3	100%
9	temperature = cool windy = false	⇒	humidity = normal play = yes	2	100%
10	temperature = cool humidity = normal windy = false	⇒	play = yes	2	100%
11	temperature = cool windy = false play = yes	⇒	humidity = normal	2	100%
12	outlook = rainy humidity = normal windy = false	⇒	play = yes	2	100%
13	outlook = rainy humidity = normal play = yes	⇒	windy = false	2	100%
14	outlook = rainy temperature = mild windy = false	⇒	play = yes	2	100%
15	outlook = rainy temperature = mild play = yes	⇒	windy = false	2	100%

Association rules in very large datasets

- The algorithm for producing association rules with specified minimum coverage and accuracy works in two stages: *generating item sets with the specified minimum coverage, and from each item set determining the rules that have the specified minimum accuracy.*

- Association rules are often sought for very large datasets, and efficient algorithms are highly valued.

- In practice, the amount of computation needed to generate association rules depends critically on the minimum coverage specified.

Apriori algorithm

- Agrawal, R.; Imieliński, T.; Swami, A. (1993). "Mining association rules between sets of items in large databases". *Proceedings of the 1993 ACM SIGMOD international conference on Management of data - SIGMOD '93*. p. 207. doi:10.1145/170035.170072. ISBN 0897915925.
  - http://www.almaden.ibm.com/cs/quest/papers/sigmod93.pdf

- Agrawal, Rakesh; and Srikant, Ramakrishnan; *Fast algorithms for mining association rules in large databases*, in Bocca, Jorge B.; Jarke, Matthias; and Zaniolo, Carlo; editors, *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB), Santiago, Chile, September 1994*, pages 487-499.

- Rakesh Agrawal
  - http://rakesh.agrawal-family.com/
  - http://research.microsoft.com/en-us/people/rakesha/

Readings for this part

- Data Mining. Witten and Frank
  - Sections 4.5

SVMs in Weka

- Parameters
  - Kernel
  - Parameters of the particular kernel

- Autoprice
  - PolyKernel
  - NormalizedPolyKernel
  - RBFKernel
  - Puk (Best Results)
- AutoMpg
- Housing

Association rules in Weka

- Association rules
  - Market Basket Analysis
    - Only a subset of the attributes (computationally expensive to use all the attributes)
  - Anduin dataset
    - Discretize sum with unsupervised discretization
    - Remove first two attributes
    - Use only some attributes for memory reasons!!!
    - Command: java –Xmx1000m –jar Weka.jar

Oracle

- SVMs in Oracle
- Association Rules in Oracle
  - Apriori algorithm