Introduction to

Information Retrieval

Hinrich Schütze and Christina Lioma

Lecture 15-1: Support Vector Machines

Overview

1. Support Vector Machines
2. Issues in the classification of text documents

Outline

1. Support Vector Machines
2. Issues in the classification of text documents

Today’s class

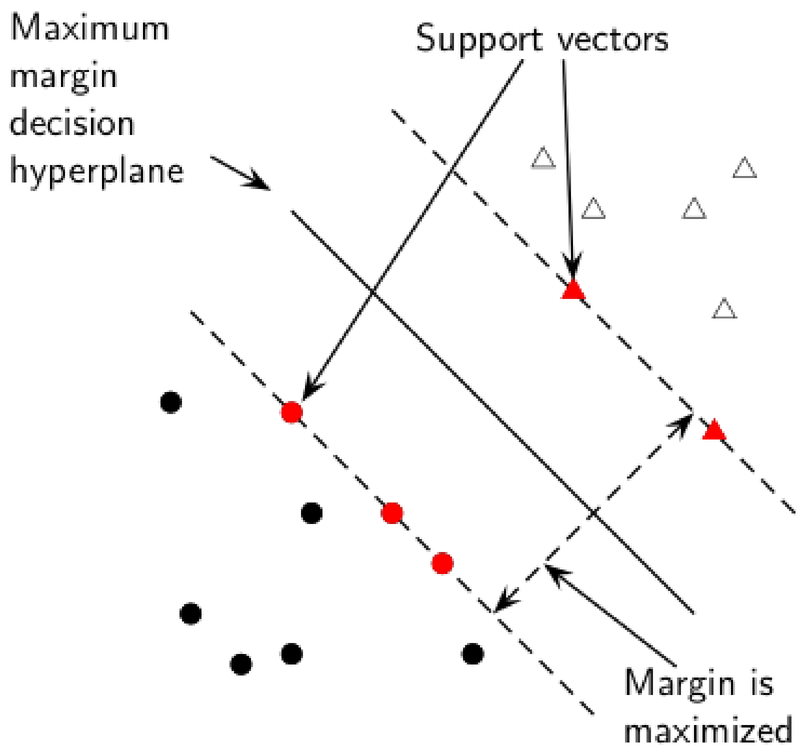
- Intensive machine-learning research in the last two decades to improve classifier effectiveness
  - New generation of state-of-the-art classifiers: support vector machines (SVMs), boosted decision trees, regularized logistic regression, neural networks, and random forests
  - Applications to IR problems, particularly text classification

SVMs: A kind of large-margin classifier

Vector space based machine-learning method aiming to find a decision boundary between two classes that is maximally far from any point in the training data (possibly discounting some points as outliers or noise)

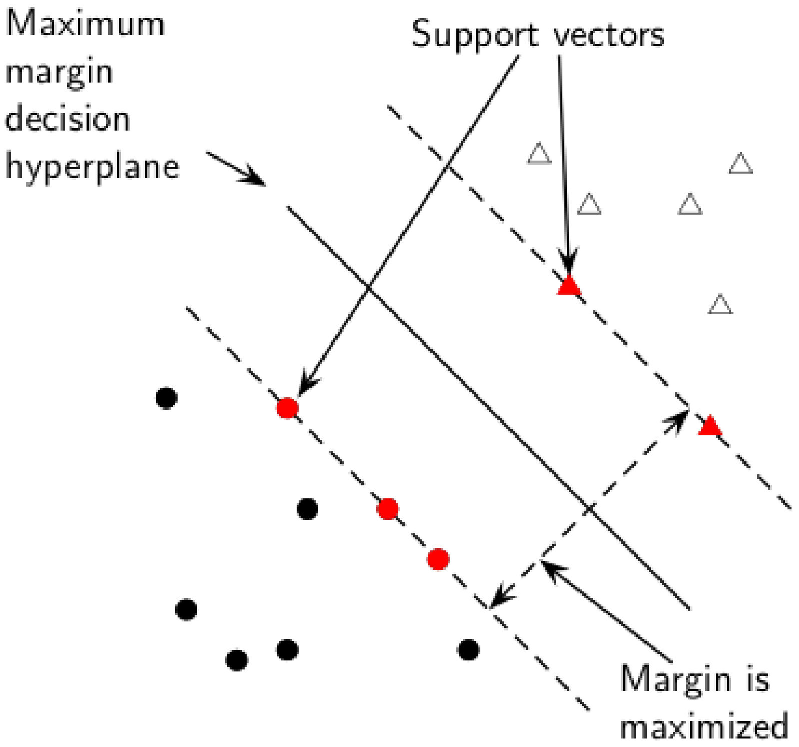
Support Vector Machines

- 2-class training data
- decision boundary
  → **linear separator**
- criterion: being maximally far away from any data point
  → determines classifier **margin**
- linear separator position defined by **support vectors**



Why maximise the margin?

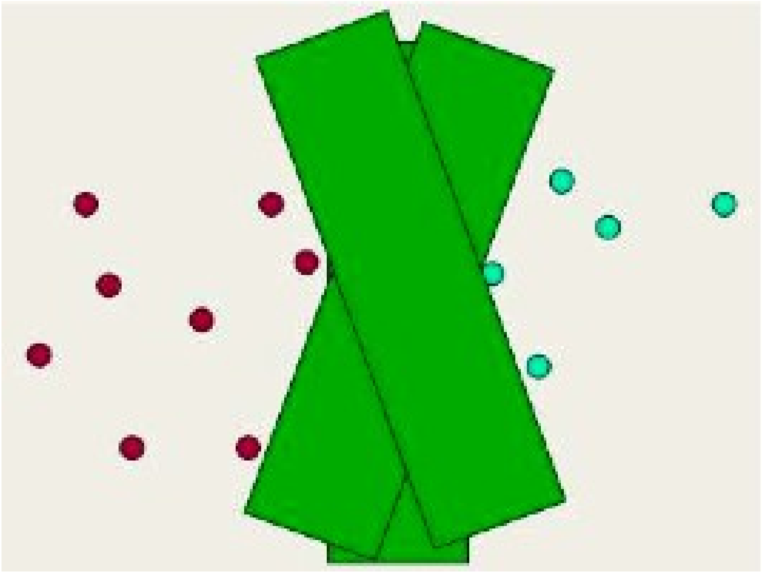
Points near decision surface → uncertain classification decisions (50% either way).
A classifier with a large margin makes no low certainty classification decisions.
Gives classification safety margin w.r.t slight errors in measurement or doc. variation



Why maximise the margin?

SVM classifier: large margin around decision boundary

- compare to decision hyperplane: place fat separator between classes
  - fewer choices of where it can be put
- decreased memory capacity
- increased ability to correctly generalize to test data



Let’s formalise an SVM with algebra

Hyperplane

An n-dimensional generalisation of a plane (point in 1-D space, line in 2-D space, ordinary plane in 3-D space).

Decision hyperplane (previously seen, page 278)

Can be defined by:

- intercept term *b*
- normal vector $\vec{w}$ (**weight vector**) which is perpendicular to the hyperplane

All points $\vec{x}$ on the hyperplane satisfy:

$$\vec{w}^{\mathrm{T}}\vec{x} = -b \tag{1}$$

Let’s formalise an SVM with algebra

Preliminaries

Consider a binary classification problem:

- $\vec{x}_i$ are the input vectors
- y_i are the labels

The $\vec{x}_i$ define a space of labelled points called input space.

For SVMs, the two data classes are always named +1 and −1, and the intercept term is always explicitly represented as b.

The linear classifier is then:

$$f(\vec{x}) = \mathrm{sign}(\vec{w}^{\mathrm{T}}\vec{x} + b) \qquad (2)$$

A value of −1 indicates one class, and a value of +1 the other class.

Functional Margin

We are confident in the classification of a point if it is far away from the decision boundary.

Functional margin

The functional margin of the i^{th} example $\vec{x}_i$ w.r.t the hyperplane

$$\langle \vec{w}, b \rangle \text{ is: } y_i(\vec{w}^{\mathsf{T}}\vec{x}_i + b)$$

The functional margin of a data set w.r.t a decision surface is twice the functional margin of any of the points in the data set with minimal functional margin

- factor 2 comes from measuring across the whole width of the margin

But we can increase functional margin by scaling $\vec{w}$ and b.
We need to place some constraint on the size of the $\vec{w}$ vector.

Geometric margin

Geometric margin of the classifier: maximum width of the band that can be drawn separating the support vectors of the two classes.

$$r = y\frac{\vec{w}^{\mathsf{T}}\vec{x} + b}{|\vec{w}|} \qquad (3)$$

The geometric margin is clearly invariant to scaling of parameters: if we replace $\vec{w}$ by 5 $\vec{w}$ and *b* by 5*b*, then the geometric margin is the same, because it is inherently normalized by the length of $\vec{w}$.

Linear SVM Mathematically

Assume canonical distance

Assume that all data is at least distance 1 from the hyperplane, then:

$$y_i(\vec{w}^{\mathsf{T}}\vec{x}_i + b) \geq 1 \qquad (4)$$

Since each example's distance from the hyperplane is $r_i = y_i(\vec{w}^{\mathsf{T}}\vec{x}_i + b)/|\vec{w}|$, the geometric margin is $\rho = 2/|\vec{w}|$

We want to maximize this geometric margin.

That is, we want to find $\vec{w}$ and b such that:

- $\rho = 2/|\vec{w}|$ is maximized
- For all $(\vec{x}_i, y_i) \in \mathbb{D},\ y_i(\vec{w}^{\mathsf{T}}\vec{x}_i + b) \geq 1$

Linear SVM Mathematically (cont.)

Maximizing $2/|\vec{w}|$ is the same as minimizing $|\vec{w}|/2$. This gives the final standard formulation of an SVM as a minimization problem:

Example

Find *w* and *b* such that:
$\frac{1}{2}\vec{w}^{\mathrm{T}}\vec{w}$ is minimized (because $|\vec{w}| = \sqrt{\vec{w}^{\mathrm{T}}\vec{w}}$) and
for all $\{(\vec{x}_i, y_i)\}$, $y_i(\vec{w}^{\mathrm{T}}\vec{x}_i + b) \geq 1$

We are now optimizing a quadratic function subject to linear constraints. Quadratic optimization problems are standard mathematical optimization problems, and many algorithms exist for solving them (e.g. Quadratic Programming libraries).

Recapitulation

We start a training data set

- The data set defines the best separating hyperplane
- We feed the data through a quadratic optimization procedure to find this plane
- Given a new point $\vec{x}$ to classify, the classification function $f(\vec{x})$ computes the projection of the point onto the hyperplane normal.
- The sign of this function determines the class to assign to the point.
- If the point is within the margin of the classifier, the classifier can return "don't know" rather than one of the two classes.
- The value of $f(\vec{x})$ may also be transformed into a probability of classification

Soft margin classification

What happens if data is not linearly separable?

- Standard approach: allow the fat decision margin to make a few mistakes
  - some points, outliers, noisy examples are inside or on the wrong side of the margin
- Pay cost for each misclassified example, depending on how far it is from meeting the margin requirement

Slack variable ξ_i : A non-zero value for ξ_i allows $\vec{x}_i$ to not meet the margin requirement at a cost proportional to the value of ξ_i.

Optimisation problem: trading off how fat it can make the margin vs. how many points have to be moved around to allow this margin.

The sum of the ξ_i gives an upper bound on the number of training errors.

Soft-margin SVMs minimize training error traded off against margin.

Multiclass support vector machines

SVMs: inherently two-class classifiers.

- Most common technique in practice: build |C| one-versus-rest classifiers (commonly referred to as “one-versus-all” or OVA classification), and choose the class which classifies the test data with greatest margin
- Another strategy: build a set of one-versus-one classifiers, and choose the class that is selected by the most classifiers. While this involves building |C|(|C| − 1)/2 classifiers, the time for training classifiers may actually decrease, since the training data set for each classifier is much smaller.

Multiclass support vector machines

Better alternative: structural SVMs

- Generalization of classification where the classes are not just a set of independent, categorical labels, but may be arbitrary structured objects with relationships defined between them
- Will look at this more closely with respect to IR ranking next time.

Outline

1. Support Vector Machines
2. Issues in the classification of text documents

Text classification

Many commercial applications

- "There is no question concerning the commercial value of being able to classify documents automatically by content. There are myriad potential applications of such a capability for corporate Intranets, government departments, and Internet publishers."

Often greater performance gains from exploiting domain-specific

text features than from changing from one machine learning method to another.

- "Understanding the data is one of the keys to successful categorization, yet this is an area in which most Categorization tool vendors are extremely weak. Many of the 'one size fits all' tools on the market have not been tested on a wide range of content types."

Choosing what kind of classifier to use

When building a text classifier, first question: how much training data is there currently available?

Practical challenge: creating or obtaining enough training data

Hundreds or thousands of examples from each class are required to produce a high performance classifier and many real world contexts involve large sets of categories.

- None?
- Very little?
- Quite a lot?
- A huge amount, growing every day?

If you have no labeled training data

Use hand-written rules

Example

IF (wheat OR grain) AND NOT (whole OR bread) THEN
c = grain

In practice, rules get a lot bigger than this, and can be phrased using more sophisticated query languages than just Boolean expressions, including the use of numeric scores. With careful crafting, the accuracy of such rules can become very high (high 90% precision, high 80% recall). Nevertheless the amount of work to create such well-tuned rules is very large. A reasonable estimate is 2 days per class, and extra time has to go into maintenance of rules, as the content of documents in classes drifts over time.

If you have fairly little data and you are going to train a supervised classifier

Work out how to get more labeled data as quickly as you can.

- Best way: insert yourself into a process where humans will be willing to label data for you as part of their natural tasks.

Example

Often humans will sort or route email for their own purposes, and these actions give information about classes.

Active Learning

A system is built which decides which documents a human should label.
Usually these are the ones on which a classifier is uncertain of the correct classification.

If you have labeled data

Reasonable amount of labeled data

Use everything that we have presented about text classification.
Preferably hybrid approach (overlay Boolean classifier)

Huge amount of labeled data

Choice of classifier probably has little effect on your results.
Choose classifier based on the scalability of training or runtime efficiency. Rule of thumb: each doubling of the training data size produces a linear increase in classifier performance, but with very large amounts of data, the improvement becomes sub-linear.

Large and difficult category taxonomies

If small number of well-separated categories, then many classification algorithms are likely to work well. But often: very large number of very similar categories.

Example

Web directories (e.g. the Yahoo! Directory consists of over 200,000 categories or the Open Directory Project), library classification schemes (Dewey Decimal or Library of Congress), the classification schemes used in legal or medical applications.

Accurate classification over large sets of closely related classes is **inherently difficult.**

Recap

- SVMs: main idea, maximum margin (soft margin briefly), binary classification (multiclass briefly)
- Issues in text classification: training data availability, taxonomies in practice

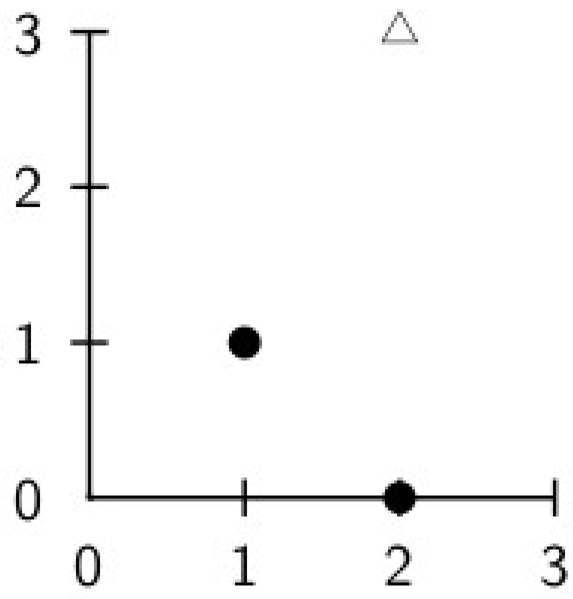
Resources

- Chapter 15 of IIR
- Resources at http://ifnlp.org/ir

Walkthrough example: building an SVM over the data set shown in the figure

Working geometrically:

- The maximum margin weight vector will be parallel to the shortest line connecting points of the two classes, that is, the line between (1, 1) and (2, 3), giving a weight vector of (1,2).
- The optimal decision surface is orthogonal to that line and intersects it at the halfway point. Therefore, it passes through (1.5, 2).
- So, the SVM decision boundary is:



$$y = x_1 + 2x_2 - 5.5$$

Walkthrough example: building an SVM over the data set shown in the figure

Working algebraically:

- With the constraint sign $(y_i(\vec{w}^\mathsf{T}\vec{x}_i + b)) \geq 1$, we seek to minimize $|\vec{w}|$.
- We know that the solution is $\vec{w} = (a, 2a)$ for some a. So: $a + 2a + b = -1$, $2a + 6a + b = 1$
- Hence, $a = 2/5$ and $b = -11/5$.

  So the optimal hyperplane is given by $\vec{w} = (2/5, 4/5)$ and $b = -11/5$.
- The margin ρ is

$$2/|\vec{w}| = 2/\sqrt{4/25 + 16/25} = 2/(2\sqrt{5}/5) = \sqrt{5}.$$