Modern Information Retrieval

Vector space classification[1]

Hamid Beigy

Sharif university of technology

November 25, 2022

[1] Some slides have been adapted from slides of Manning, Yannakoudakis, and Schütze.

Table of contents

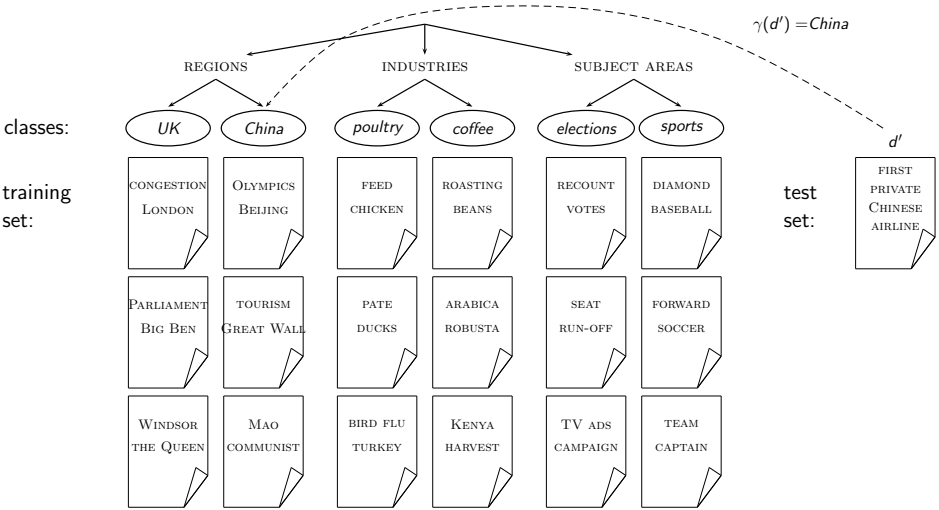
Introduction

Vector space representation

1. Each document is a vector, one component for each term.
2. Terms are axes.
3. High dimensionality: 100,000s of dimensions
4. Normalize vectors (documents) to unit length
5. How can we do classification in this space?

Classification terminology

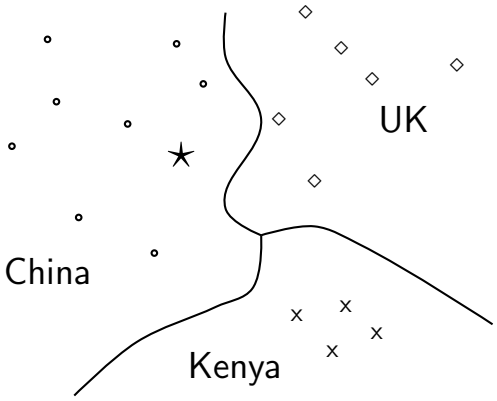
1. Consider a text classification with six classes {UK, China, poultry, coffee, elections, sports}



1. As before, the training set is a set of documents, each labeled with its class.
2. In vector space classification, this set corresponds to a labeled set of points or vectors in the vector space.
3. Assumption 1: Documents in the same class form a contiguous region.
4. Assumption 2: Documents from different classes don't overlap.
5. We define lines, surfaces, hypersurfaces to divide regions.

Classes in the vector space

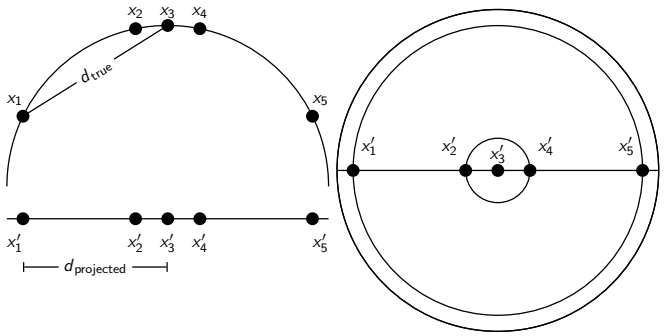
1. Consider the following regions.



2. Should the document $\star$ be assigned to *China*, *UK* or *Kenya*?
3. Find separators between the classes
4. Based on these separators: $\star$ should be assigned to *China*
5. How do we find separators that do a good job at classifying new documents like $\star$?

Aside: 2D/3D graphs can be misleading

1. Consider the following points.



2. Left: A projection of the 2D semicircle to 1D.
   For the points x_1, x_2, x_3, x_4, x_5 at x coordinates $-0.9, -0.2, 0, 0.2, 0.9$ the distance $|x_2x_3| \approx 0.201$ only differs by 0.5% from $|x'_2x'_3| = 0.2$; but $|x_1x_3|/|x'_1x'_3| = d_{\text{true}}/d_{\text{projected}} \approx 1.06/0.9 \approx 1.18$ is an example of a large distortion (18%) when projecting a large area.
3. Right: The corresponding projection of the 3D hemisphere to 2D.

Rocchio classifier

Relevance feedback

1. In relevance feedback, the user marks documents as relevant/nonrelevant.
2. Relevant/nonrelevant can be viewed as classes or categories.
3. For each document, the user decides which of these two classes is correct.
4. The IR system then uses these class assignments to build a better query ("model") of the information need and returns better documents.
5. Relevance feedback is a form of text classification.

Using Rocchio for vector space classification

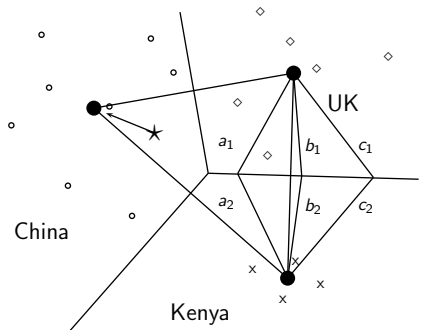
1. The principal difference between relevance feedback and text classification:
   - The training set is given as part of the input in text classification.
   - It is interactively created in relevance feedback.
2. Basic idea of Rocchio classification
   - Compute a centroid for each class
   - The centroid is the average of all documents in the class.
   - Assign each test document to the class of its closest centroid.

1. The definition of centroid is

$$\vec{\mu}(c) = \frac{1}{|D_c|} \sum_{d \in D_c} \vec{v}(d)$$

where D_c is the set of all documents that belong to class c and $\vec{v}(d)$ is the vector space representation of d.

2. An example of Rocchio classification ($a_1 = a_2, b_1 = b_2, c_1 = c_2$)

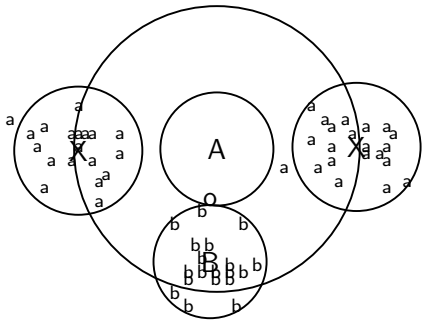


Rocchio properties

1. Rocchio forms a simple representation for each class: the centroid
   - We can interpret the centroid as the prototype of the class.
2. Classification is based on similarity to / distance from centroid/prototype.
3. Does not guarantee that classifications are consistent with the training data!

Rocchio vs. Naive Bayes

1. In many cases, Rocchio performs worse than Naive Bayes.
2. One reason: Rocchio does not handle nonconvex, multimodal classes correctly.



kNN classification

kNN classification

1. kNN classification is another vector space classification method.
2. It also is very simple and easy to implement.
3. kNN is more accurate (in most cases) than Naive Bayes and Rocchio.
4. If you need to get a pretty accurate classifier up and running in a short time and you don't care about efficiency that much use kNN.

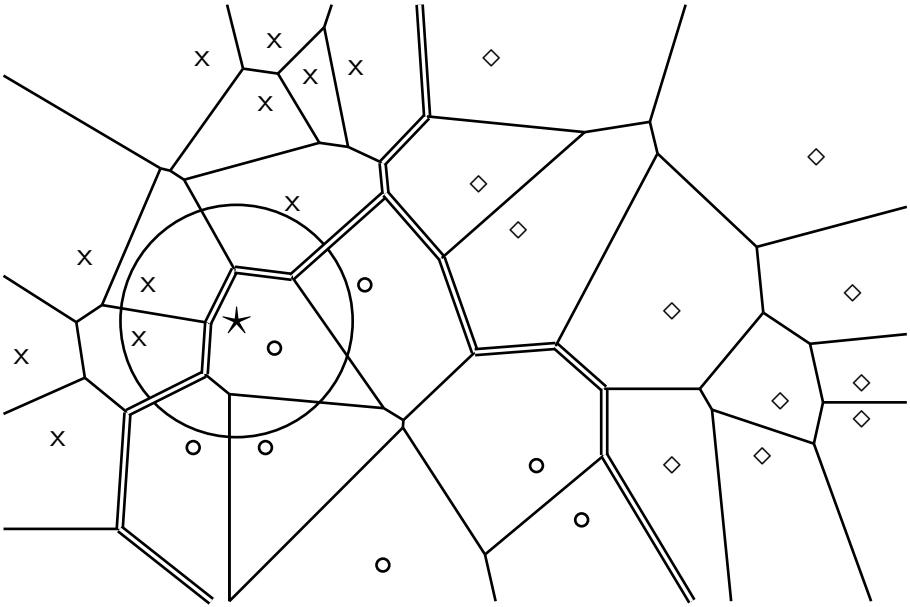
kNN classification

1. kNN = k nearest neighbors
2. kNN classification rule for $k = 1$ (1NN): Assign each test document to the class of its nearest neighbor in the training set.
3. 1NN is not very robust – one document can be mislabeled or atypical.
4. kNN classification rule for $k > 1$ (kNN): Assign each test document to the majority class of its k nearest neighbors in the training set.
5. Rationale of kNN: contiguity hypothesis
6. We expect a test document d to have the same label as the training documents located in the local region surrounding d.

Probabilistic kNN

1. Probabilistic version of kNN: $P(c|d)$ = fraction of k neighbors of d that are in c
2. kNN classification rule for probabilistic kNN: Assign d to class c with highest $P(c|d)$

kNN is based on Voronoi tessellation



Curse of dimensionality

1. Our intuitions about space are based on the 3D world we live in.
   - Some things are close by, some things are distant.
   - We can carve up space into areas such that: within an area things are close, distances between areas are large.
2. These two intuitions don't necessarily hold for high dimensions.
3. In particular: for a set of k uniformly distributed points, let d_{min} be the smallest distance between any two points and d_{max} be the largest distance between any two points.
4. Then

$$\lim_{d \to \infty} \frac{d_{max} - d_{min}}{d_{min}} = 0$$

kNN: Discussion

1. No training necessary
   - But linear preprocessing of documents is as expensive as training Naive Bayes.
   - We always preprocess the training set, so in reality training time of kNN is linear.
2. kNN is very accurate if training set is large.
3. Optimality result: asymptotically zero error if Bayes rate is zero.
4. But kNN can be very inaccurate if training set is small.

Linear classifiers

1. A linear classifier classifies documents as

Definition (Linear classifier)

A linear classifier computes a linear combination or weighted sum $\sum_i w_i x_i$ of the feature values. Classification decision: $\sum_i w_i x_i > \theta$? where θ (the threshold) is a parameter.

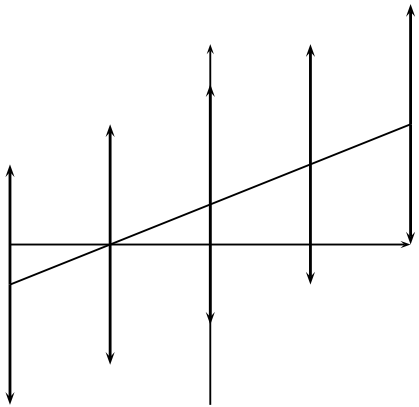
2. First, we only consider binary classifiers.
3. Geometrically, this corresponds to a line (2D), a plane (3D) or a hyperplane (higher dimensionality), the separator.
4. We find this separator based on training set.
5. Methods for finding separator: Perceptron, Rocchio, Naive Bayes – as we will explain on the next slides
6. Assumption: The classes are linearly separable.

A linear classifier in 1D

1. A linear classifier in 1D is a point described by the equation $w_1 d_1 = \theta$
2. The point at θ / w_1
3. Points (d_1) with $w_1 d_1 \geq \theta$ are in the class c.
4. Points (d_1) with $w_1 d_1 < \theta$ are in the complement class $\overline{c}$.

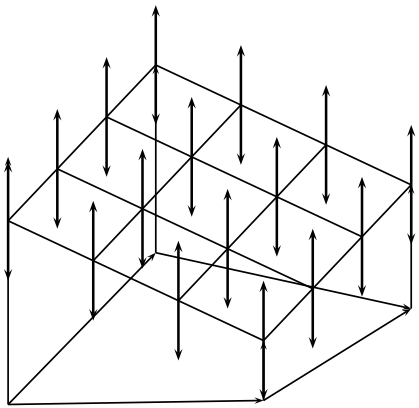
A linear classifier in 2D

1. A linear classifier in 2D is a line described by the equation $w_1 d_1 + w_2 d_2 = \theta$
2. Example for a 2D linear classifier
3. Points $(d_1\ d_2)$ with $w_1 d_1 + w_2 d_2 \geq \theta$ are in the class c.
4. Points $(d_1\ d_2)$ with $w_1 d_1 + w_2 d_2 < \theta$ are in the complement class $\overline{c}$.



A linear classifier in 3D

1. A linear classifier in 3D is a plane described by the equation $w_1 d_1 + w_2 d_2 + w_3 d_3 = \theta$
2. Example for a 3D linear classifier
3. Points $(d_1\ d_2\ d_3)$ with $w_1 d_1 + w_2 d_2 + w_3 d_3 \geq \theta$ are in the class c.
4. Points $(d_1\ d_2\ d_3)$ with $w_1 d_1 + w_2 d_2 + w_3 d_3 < \theta$ are in the complement class $\overline{c}$.



Rocchio as a linear classifier

1. Rocchio is a linear classifier defined by (show it):

$$\sum_{i=1}^{M} w_i d_i = \vec{w}\vec{d} = \theta$$

where $\vec{w}$ is the normal vector $\vec{\mu}(c_1) - \vec{\mu}(c_2)$ and $\theta = 0.5 * (|\vec{\mu}(c_1)|^2 - |\vec{\mu}(c_2)|^2)$.

Naive Bayes as a linear classifier

1. Multinomial Naive Bayes is a linear classifier (in log space) defined by (show it):

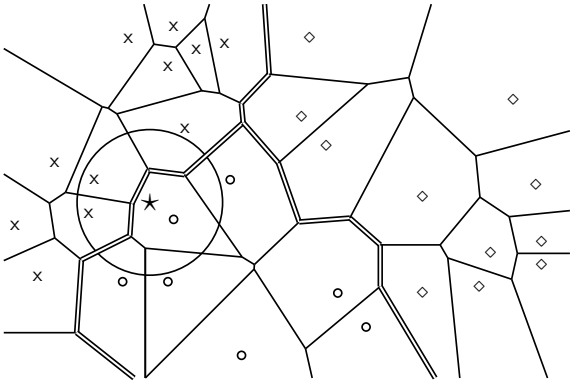
$$\sum_{i=1}^{M} w_i d_i = \theta$$

   where $w_i = \log[\hat{P}(t_i|c)/\hat{P}(t_i|\bar{c})]$, d_i = number of occurrences of t_i in d, and $\theta = -\log[\hat{P}(c)/\hat{P}(\bar{c})]$.

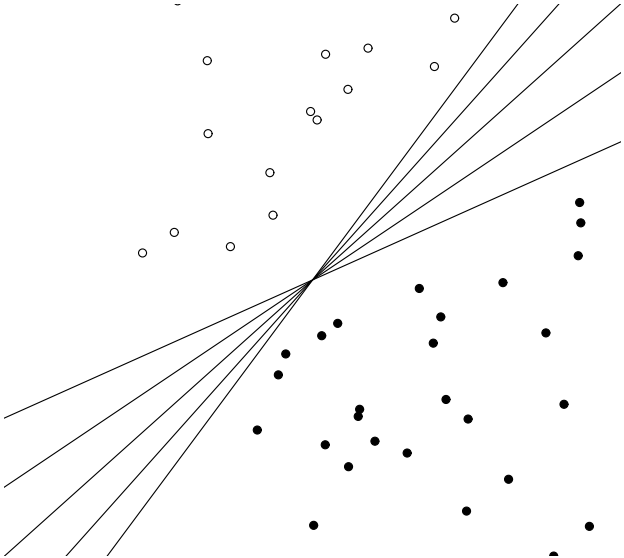
2. Here, the index i, $1 \leq i \leq M$, refers to terms of the vocabulary (not to positions in d as k did in our original definition of Naive Bayes)

kNN is not a linear classifier

1. Classification decision based on majority of k nearest neighbors.
2. The decision boundaries between classes are piecewise linear.
3. In general, they are not linear classifiers that can be described as $\sum_{i=1}^{M} w_i d_i = \theta$.



Which hyperplane?



1. In terms of actual computation, there are two types of learning algorithms.
   1.1 Simple learning algorithms that estimate the parameters of the classifier directly from the training data, often in one linear pass such as Naive Bayes, Rocchio, kNN are all examples of this.
   1.2 Iterative algorithms such as Perceptron
2. The best performing learning algorithms usually require iterative learning.

Perceptron update rule

1. Randomly initialize linear separator $\vec{w}$
2. Do until convergence:
   - Pick data point $\vec{x}$
   - If sign($\vec{w}^T\vec{x}$) is correct class (1 or -1): do nothing
   - Otherwise: $\vec{w} = \vec{w} - \text{sign}(\vec{w}^T\vec{x})\vec{x}$

Which hyperplane?

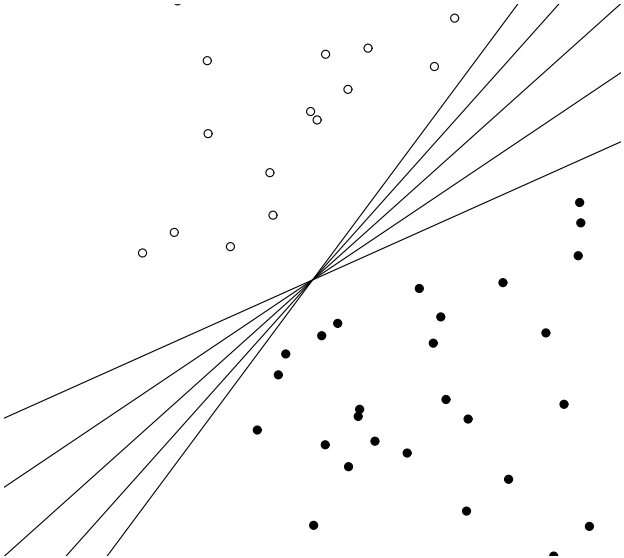
1. For linearly separable training sets: there are infinitely many separating hyperplanes.
2. They all separate the training set perfectly but they behave differently on test data.
3. Error rates on new data are low for some, high for others.
4. How do we find a low-error separator?
5. Perceptron: generally bad; Naive Bayes, Rocchio: ok; linear SVM: good

Support vector machines

What is a support vector machine

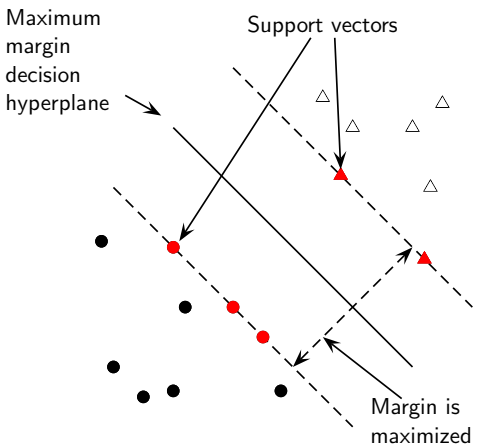
1. Vector space classification (similar to Rocchio, kNN, linear classifiers)
2. Difference from previous methods: large margin classifier
3. We aim to find a separating hyperplane (decision boundary) that is maximally far from any point in the training data
4. In case of non-linear-separability: We may have to discount some points as outliers or noise.

Which hyperplane?



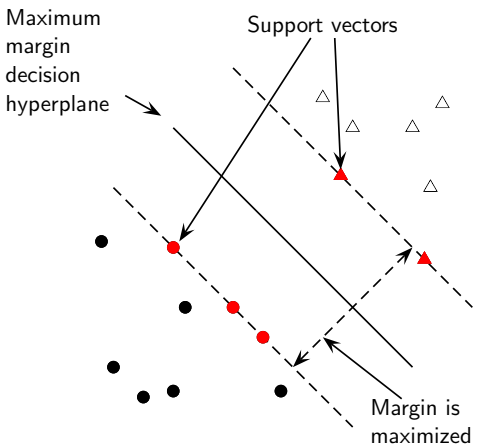
(Linear) Support Vector Machines

1. Binary classification problem
2. Decision boundary is linear separator.
3. Being maximally far away from any data point (determines classifier margin)
4. Vectors on margin lines are called support vectors
5. Set of support vectors are a complete specification of classifier



Why maximize the margin?

1. Points near the decision surface are uncertain classification decisions.
2. A classifier with a large margin makes no low certainty classification decisions (on the training set).
3. Gives classification safety margin with respect to errors and random variation



Separating hyperplane (review)

Definition (Hyperplane)

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

Definition (Decision hyperplane)

Can be defined by:

- intercept term b (we were calling this θ before)
- normal vector $\vec{w}$ (weight vector) which is perpendicular to the hyperplane

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

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

Notation: Different conventions for linear separator

1. Used in SVM literature: $\vec{w}^T\vec{x} + b = 0$
2. Often used in perceptron literature, folds threshold into vector by adding a constant dimension (set to 1 or -1 for all vectors): $\vec{w}^T\vec{x} = 0$
3. A version we used in the last session for linear separators $\sum_{i=1}^{M} w_i d_i = \theta$

Definition (Training set)

Consider a binary classification problem:

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

For SVMs, the two classes are $y_i = +1$ and $y_i = -1$.

Definition (Linear classifier)

$$f(\vec{x}) = \text{sign}(\vec{w}^{\mathsf{T}}\vec{x} + b)$$

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

Functional margin of a point

SVM makes its decision based on the score $\vec{w}^{\mathsf{T}}\vec{x} + b$. Clearly, the larger $|\vec{w}^{\mathsf{T}}\vec{x} + b|$ is, the more confidence we can have that the decision is correct.

Definition (Functional margin)

- The functional margin of the vector $\vec{x_i}$ w.r.t the hyperplane $\langle \vec{w}, b \rangle$ 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.

Problem: We can increase functional margin by scaling $\vec{w}$ and b. (We need to place some constraint on the size of $\vec{w}$.)

Geometric margin

1. Geometric margin of the classifier equals to the maximum width of the band that can be drawn separating the support vectors of the two classes.
2. To compute the geometric margin, we need to compute the distance of a vector $\vec{x}$ from the hyperplane:

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

3. Distance is of course invariant to scaling: if we replace $\vec{w}$ by $5\vec{w}$ and b by $5b$, then the distance is the same because it is normalized by the length of $\vec{w}$.

Optimization problem solved by SVMs

1. Assume canonical "functional margin" distance
2. Assume that every data point has at least distance 1 from the hyperplane, then:
$$y_i(\vec{w}^{\mathsf{T}}\vec{x}_i + b) \geq 1$$
3. Since each example's distance from the hyperplane is $r_i = y_i(\vec{w}^{\mathsf{T}}\vec{x}_i + b)/|\vec{w}|$, the margin is $\rho = 2/|\vec{w}|$.
4. We want to maximize this margin. That is, we want to find $\vec{w}$ and b such that:
   - For all $(\vec{x}_i, y_i) \in \mathbb{D}$, $y_i(\vec{w}^{\mathsf{T}}\vec{x}_i + b) \geq 1$
   - $\rho = 2/|\vec{w}|$ is maximized

Optimization problem solved by SVMs

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:

Optimization problem solved by SVMs

Find $\vec{w}$ 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).

- The optimization problem for SVM is defined as

$$\textit{Minimize} \frac{1}{2}\|w\|^2 \qquad \text{subject to } y_n\left(w^T x_n + b\right) \geq 1 \text{ for all } n = 1, 2, \ldots, N$$

- To solve this problem, we use Lagrange multipliers $\alpha_n \geq 0$, with one multiplier α_n for each of the constraints

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{n=1}^{N} \alpha_n \left[y_n\left(w^T x_n + b\right) - 1\right]$$

where $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_N)^T$.

- Note the minus sign in front of the Lagrange multiplier term, because we are minimizing with respect to w and b, and maximizing with respect to α.
- Setting the derivatives of $L(w, b, \alpha)$ with respect to w and b equal to zero,

$$\frac{\partial L}{\partial w} = 0 \Rightarrow w = \sum_{n=1}^{N} \alpha_n y_n y_n$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow 0 = \sum_{n=1}^{N} \alpha_n y_n$$

- L has to be minimized with respect to the primal variables w and b and maximized with respect to the dual variables α_n. Eliminating w and b from $L(w, b, a)$ using these conditions then gives the dual representation of the problem in which we maximize

$$\psi(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m y_n y_m x_n^T x_m$$

- We need to maximize $\psi(\alpha)$ subject to the following constraints

$$\begin{aligned} \alpha_n &\geq 0 \quad \forall n \\ \sum_{n=1}^{N} \alpha_n y_n &= 0 \end{aligned}$$

- The constrained optimization of this form satisfies the Karush-Kuhn-Tucker (KKT) conditions, which in this case require that the following three properties hold

$$\begin{aligned} \alpha_n &\geq 0 \\ y_n g(x_n) &\geq 1 \end{aligned}$$

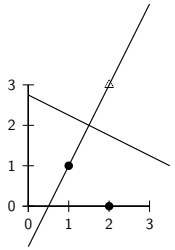
- For optimal α_n's,

$$\alpha_n \left[1 - y_n \left(W^T x_n + b\right)\right] = 0$$

- α_n is non-zero only if x_n lies on one of the two margin boundaries, i.e., for which $y_n(w^T x_n + b) = 1$
- These examples are called support vectors.
- To classify a data x using the trained model, we evaluate the sign of $g(x)$ defined by

$$g(x) = \sum_{n=1}^{N} \alpha_n y_n x_n^T x$$

- A simple example

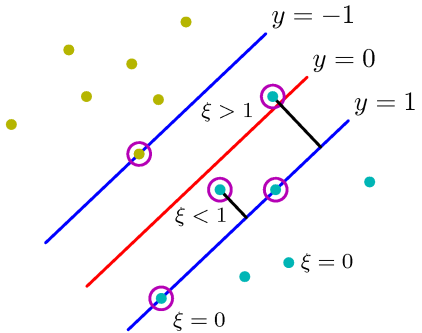


Soft margin classification

1. We have assumed that the training data are linearly separable in the feature space. The resulting SVM will give exact separation of the training data.
2. In the practice, the class-conditional distributions may overlap, in which the exact separation of the training data can lead to poor generalization.
3. 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 margin requirement
4. Pay cost for each misclassified example, depending on how far it is from meeting the
5. We need a way to modify the SVM so as to allow some training examples to be miss-classified.

Soft margin classification

1. We need a way to modify the SVM so as to allow some training examples to be miss-classified.
2. To do this, we introduce slack variables ($\xi_n \geq 0$); one slack variable for each training example.
3. The slack variables are defined by $\xi_n = 0$ for examples that are inside the correct boundary margin and $\xi_n = |y_n - g(\vec{x}_n)|$ for other examples.
4. Thus for data point that is on the decision boundary $g(\vec{x}_n) = 0$ will have $\xi_n = 1$ and the data points with $\xi_n \geq 1$ will be misclassified.



Soft margin classification

1. The exact classification constraints will be

$$y_n g(\vec{x}_n) \geq 1 - \xi_n \quad \text{for } n = 1, 2, \ldots, N$$

2. Our goal is now to maximize the margin while softly penalizing points that lie on the wrong side of the margin boundary. We minimize

$$C \sum_{n=1}^{N} \xi_n + \frac{1}{2} \|w\|^2$$

   $C > 0$ controls the trade-off between the slack variable penalty and the margin.

3. We now wish to solve the following optimization problem.

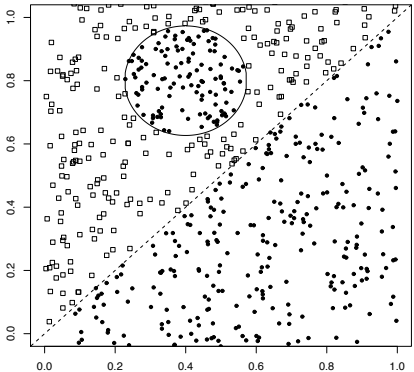
$$\min_w \frac{1}{2} \|w\|^2 + C \sum_{n=1}^{N} \xi_n \quad \text{s.t. } y_n g(\vec{x}_n) \geq 1 - \xi_n \text{ for all } n = 1, 2, \ldots, N$$

Linear classifiers: Discussion

1. Many common text classifiers are linear classifiers: Naive Bayes, Rocchio, logistic regression, linear support vector machines etc.
2. Each method has a different way of selecting the separating hyperplane
3. Huge differences in performance on test documents
4. Can we get better performance with more powerful nonlinear classifiers?
5. Not in general: A given amount of training data may suffice for estimating a linear boundary, but not for estimating a more complex nonlinear boundary.

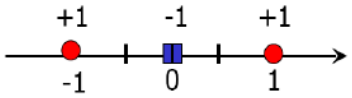
A nonlinear problem

1. Nolinear classifiers create nonlinear boundaries

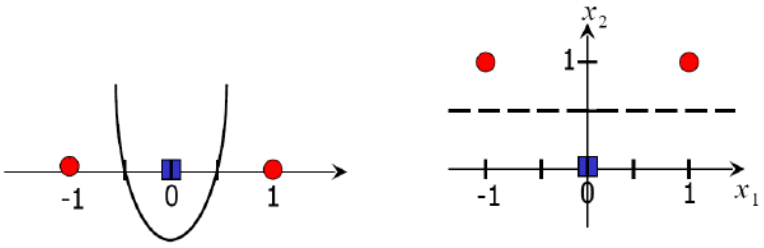


2. Linear classifier like Rocchio does badly on this task.
3. kNN will do well (assuming enough training data)

- Most data sets are not linearly separable, for example



- Instances that are not linearly separable in $1-$dimension, may be linearly separable in $2-$ dimensions, for example



- In this case, we have two solutions
  - Increase dimensionality of data set by introducing mapping ϕ.
  - Use a more complex model for classifier.

- The SVM uses the following discriminant function.

$$g(x) = \sum_{n=1}^{N} \alpha_n t_n x_n^T x$$

- This solution depends on the dot-product between two pints x_n and x.
- The operation in high dimensional space $\phi(x)$ don't have performed explicitly if we can find a function $K(x_i, x_j)$ such that $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$.
- $K(x_i, x_j)$ is called kernel in the SVM.
- Suppose $x, z \in \mathbb{R}^D$ and consider the following kernel

$$K(x, z) = \left(x^T z\right)^2$$

- It is a valid kernel because

$$K(x, z) = \left(\sum_{i=1}^{D} x_i z_i\right)\left(\sum_{j=1}^{D} x_j z_j\right) = \sum_{i=1}^{D}\sum_{j=1}^{D} (x_i x_j)(z_i z_j) = \phi(x)^T \phi(z)$$

where the mapping ϕ for $D = 2$ is

$$\phi(x) = (x_1 x_1, x_1 x_2, x_2 x_1, x_2 x_2)^T$$

Non-linear support vector machine (cont.)

- Show that kernel $K(x, z) = (x^T z + c)^2$ is a valid kernel.
- Kernel K is valid if there is mapping ϕ such that $K(x, z) = \phi(x)^T \phi(z)$.
- Assume that K is a valid kernel. Consider a set of N points, K is $N \times N$ square matrix defined as

$$K = \begin{pmatrix} k(x_1, x_1) & k(x_1, x_2) & \cdots & k(x_1, x_N) \\ k(x_2, x_1) & k(x_2, x_2) & \cdots & k(x_2, x_N) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_N, x_1) & k(x_N, x_2) & \cdots & k(x_N, x_N) \end{pmatrix}$$

  K is called kernel matrix.
- If K is a valid kernel then

$$k_{ij} = k(x_i, x_j) = \phi(x_i)^T \phi(x_j) = \phi(x_j)^T \phi(x_i) = k(x_j, x_i) = k_{ji}$$

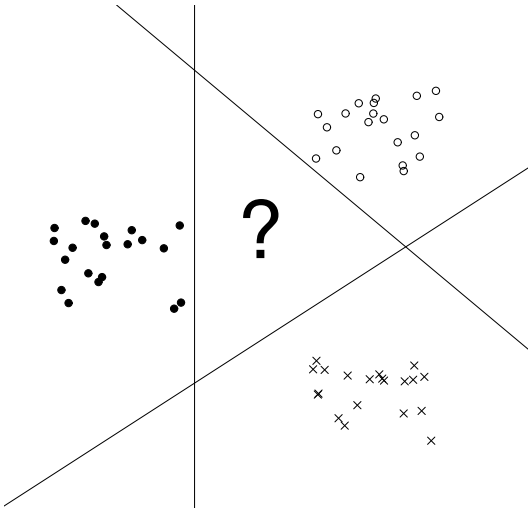
- Thus K is symmetric. It can also be shown that K is positive semi-definite.
- Thus if K is a valid kernel, then the corresponding kernel matrix is symmetric positive semi-definite. It is both necessary and sufficient conditions for K to be a valid kernel.

Advantages and disadvantages of SVM

- Advantages
  - The problem doesn't have local minima and we can found its optimal solution in polynomial time.
  - The solution is stable, repeatable, and sparse (it only involves the support vectors).
  - The user must select a few parameters such as the penalty term C and the kernel function and its parameters.
  - The algorithm provides a method to control complexity independently of dimensionality.
  - SVMs have been shown (theoretically and empirically) to have excellent generalization capabilities.
- Disadvantages
  - There is no method for choosing the kernel function and its parameters.
  - It is not a straight forward method to extend SVM to multi-class classifiers.
  - Predictions from a SVM are not probabilistic.
  - It has high algorithmic complexity and needs extensive memory to be used in large-scale tasks.

Multi classes classification

How to combine hyperplanes for multi classes classification?



Multi classes classification

1. In classification, the goal is to find a mapping from inputs X to outputs $t \in \{1, 2, \ldots, C\}$ given a labeled set of input-output pairs.
2. We can extend the binary classifiers to C class classification problems or use the binary classifiers.
3. For C-class, we have four extensions for using binary classifiers.
   - One-against-all: This approach is a straightforward extension of two-class problem and considers it as a of C two-class problems.
   - One-against-one: In this approach, $C(C-1)/2$ binary classifiers are trained and each classifier separates a pair of classes. The decision is made on the basis of a majority vote.
   - Single $C-$class discriminant: In this approach, a single $C-$class discriminant function comprising C linear functions are used.
   - Hierarchical classification: In this approach, the output space is hierarchically divided i.e. the classes are arranged into a tree.
   - Error correcting coding: For a $C-$class problem a number of L binary classifiers are used,where L is appropriately chosen by the designer. Each class is now represented by a binary code word of length L.

Which classifier do I use for a given TC problem?

1. Is there a learning method that is optimal for all text classification problems?
2. No, because there is a tradeoff between bias and variance.
3. Factors to take into account:
   - How much training data is available?
   - How simple/complex is the problem? (linear vs. nonlinear decision boundary)
   - How noisy is the problem?
   - How stable is the problem over time?
   - For an unstable problem, it's better to use a simple and robust classifier.

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

1. Use hand-written rules!

Example

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

2. 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.
3. With careful crafting, the accuracy of such rules can become very high (high 90% precision, high 80% recall).
4. Nevertheless the amount of work to create such well-tuned rules is very large.
5. 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 the training set is small

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

Good amount of labeled data, but not huge

Use everything that we have presented about text classification. Consider 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.

References

Reading

1. Chapters 14 and 15 of Information Retrieval Book[2]

[2] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze (2008). *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press.

References


Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze (2008). *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press.

Questions?