Linear Classifiers

- A linear classifier is a classifier such that classification is performed by a dot product between the two vectors representing the document and the category, respectively. Therefore it consists in a document-like representation of the category c_i
- Linear classifiers are thus very efficient at classification time
- Methods for learning linear classifiers can be partitioned in two broad classes
  - Incremental methods (IMs) (or on-line) build a classifier soon after examining the first document, as incrementally refine it as they examine new ones
  - Batch methods (BMs) build a classifier by analyzing Tr all at once.

LCs for Binary Classification

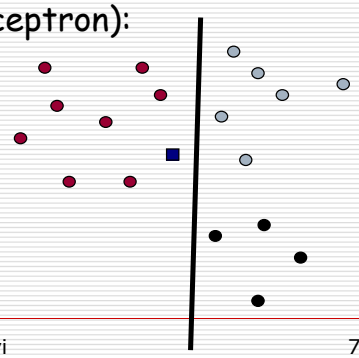
- Consider 2 class problems in VSM
  - Deciding between two classes, perhaps, sport and not sport
  - How do we define (and find) the separating surface (hyperplane)?
- How do we test which region a test doc is in?

Separation by Hyperplanes

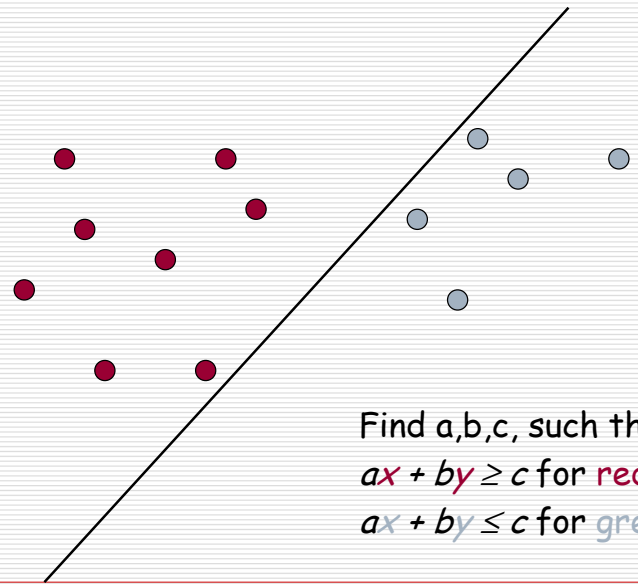
- A strong high-bias assumption is *linear separability*.
  - in 2 dimensions, can separate classes by a line
  - in higher dimensions, need hyperplanes
- Can find separating hyperplane by *linear programming*

  (or can iteratively fit solution via perceptron):
  - separator can be expressed as

    $ax + by = c$



Linear programming / Perceptron



Find a,b,c, such that

$ax + by \geq c$ for red points

$ax + by \leq c$ for green points.

Linear programming / Perceptron

$w^t x + b > 0$ — If x belongs to Sport ($y = +1$)

$w^t x + b < 0$ — Otherwise ($y = -1$)

You can write $y([w\ b]\ [x\ 1]^t) > 0$

Thus, we want to compute a vector W s.t.

$$y_i W^t X_i > 0$$

where $W=[w\ b]^t$ and $X_i=[x_i\ 1]^t$

The Perceptron Algorithm

A simple (but powerful) method is the perceptron

1. Initialize W=0
2. For all training examples (X_i,y_i), i=1,..,n
   If $(y_i\ W^t \cdot X_i <= 0)\ W = W + y_i X_i$
3. If no errors have been done in step 2, stop. Otherwise repeat step 2.

Linear classifier: Example

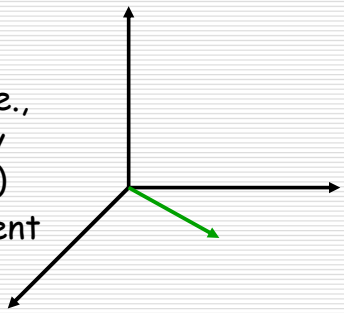
- Class: "interest" (as in interest rate)
- Example features of a linear classifier
w_i	t_i	w_i	t_i
0.70	prime	-0.71	dlrs
0.67	rate	-0.35	world
0.63	interest	-0.33	sees
0.60	rates	-0.25	year
0.46	discount	-0.24	group
0.43	bundesbank	-0.24	dlr
- To classify, find dot product of feature vector and weights
  - Score("rate discount dlrs world") = .67+.46-.71-.35=0.05 > 0
  - Score("prime dlrs") = .70-.71=-.01 < 0

Linear Classifiers

- Many common text classifiers are linear classifiers
  - Naïve Bayes
  - Perceptron
  - Rocchio
  - Logistic regression
  - Support vector machines (with linear kernel)
  - Linear regression
  - (Simple) perceptron neural networks
- Despite this similarity, large performance differences
  - For separable problems, there is an infinite number of separating hyperplanes. Which one do you choose?
  - What to do for non-separable problems?
  - Different training methods pick different hyperplanes
- Classifiers more powerful than linear often don't perform better. Why?

High Dimensional Data

- Documents are zero along almost all axes
- Most document pairs are very far apart (i.e., not strictly orthogonal, but only share very common words and a few scattered others)
- In classification terms: virtually all document sets are separable, for most any classification
- This is part of why linear classifiers are quite successful in this domain



Naive Bayes is a linear classifier

- Two-class Naive Bayes. We compute:

$$\log\frac{P(C\mid d)}{P(\overline{C}\mid d)}=\log\frac{P(C)}{P(\overline{C})}+\sum_{w\in d}\log\frac{P(w\mid C)}{P(w\mid\overline{C})}$$

- Decide class C if the odds ratio is greater than 1, i.e., if the log odds is greater than 0.
- So decision boundary is hyperplane:

$$\alpha+\sum_{w\in V}\beta_w\times n_w=0 \quad \text{where} \quad \alpha=\log\frac{P(C)}{P(\overline{C})};$$

$$\beta_w=\log\frac{P(w\mid C)}{P(w\mid\overline{C})}; \quad n_w=\#\text{ of occurrences of } w \text{ in } d$$

kNN vs. Linear Classifiers

- Bias/Variance tradeoff
  - Variance ≈ Capacity
- kNN has high variance and low bias.
  - Infinite memory
- LCs has low variance and high bias.
  - Decision surface has to be linear (hyperplane)
- Consider: Is an object a tree?
  - Too much capacity/variance, low bias
    - Botanist who memorizes
    - Will always say "no" to new object (e.g., # leaves)
  - Not enough capacity/variance, high bias
    - Lazy botanist
    - Says "yes" if the object is green
  - Want the middle ground

Bias vs. variance: Choosing the correct model capacity