Support Vector Machines

Some slides were borrowed from Andrew Moore’s PowetPoint slides on SVMs. Andrew’s PowerPoint repository is here: http://www.cs.cmu.edu/~awm/tutorials . Comments and corrections gratefully received.

Support Vector Machines

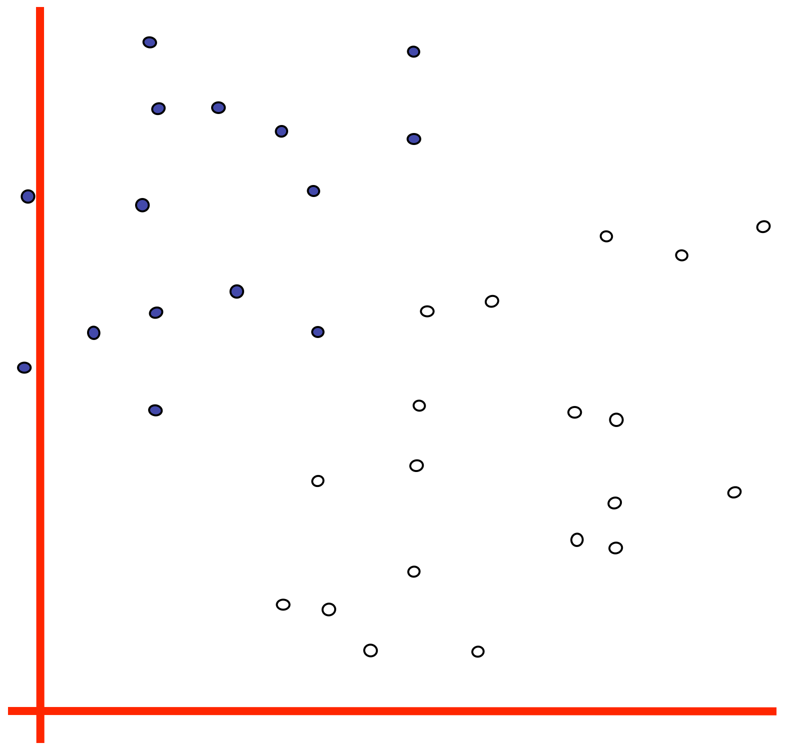
- Very popular ML technique
  - Became popular in the late 90s (Vapnik 1995; 1998)
  - Invented in the late 70s (Vapnik, 1979)
- Controls complexity and overfitting, so works well on a wide range of practical problems
- Can handle high dimensional vector spaces, which makes feature selection less critical
- Very fast and memory efficient implementations, e.g., svm_light
- Not always the best solution, especially for problems with small vector spaces

Linear Classifiers

$$\begin{tikzcd} & \alpha \arrow[d] & \\ x \arrow[r] & \boxed{f} \arrow[r] & y^{est} \end{tikzcd}$$

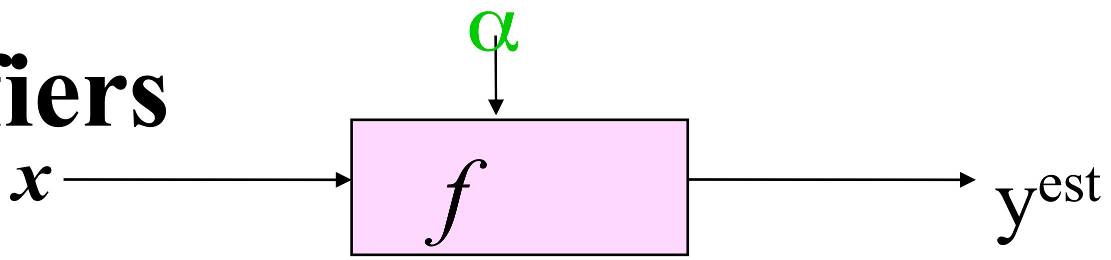
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

- ● denotes +1
- ○ denotes -1



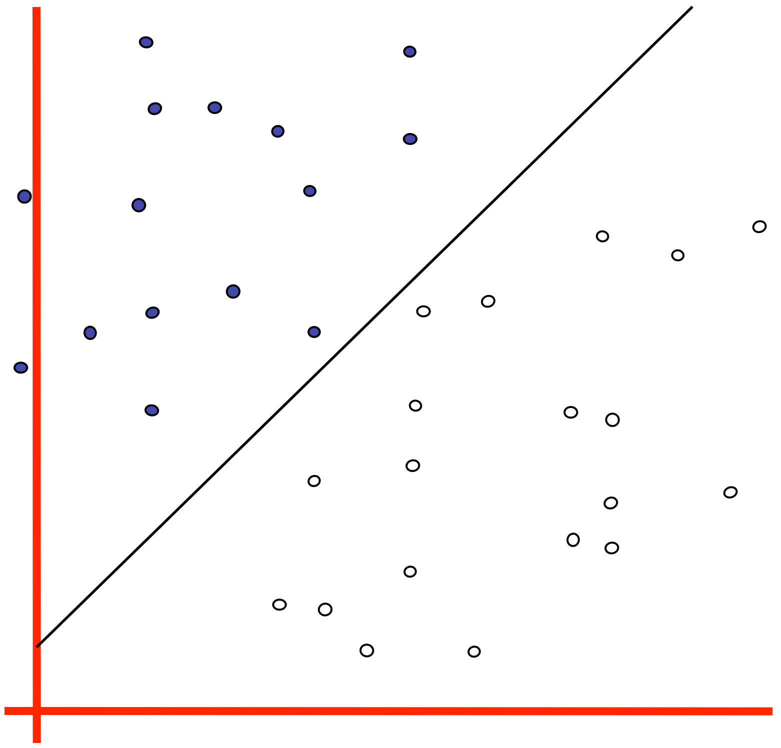
How would you classify this data?

Linear Classifiers



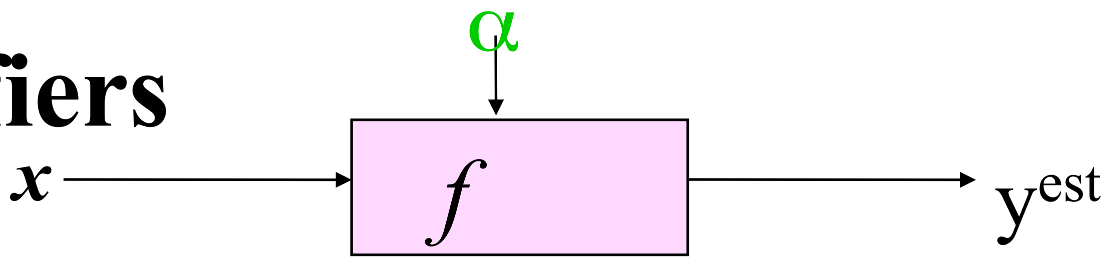
$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w} \cdot \mathbf{x} - b)$

- ● denotes +1
- ○ denotes -1



How would you classify this data?

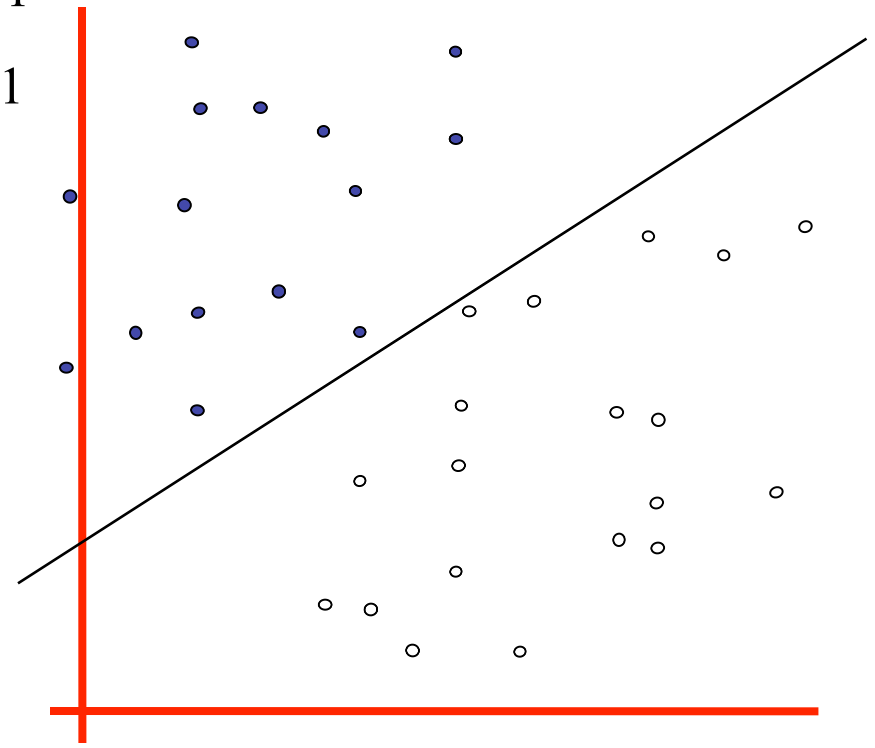
Linear Classifiers



$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

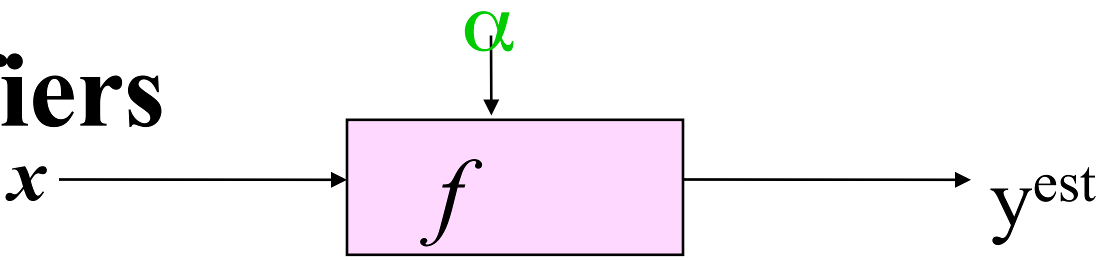
● denotes +1

○ denotes -1

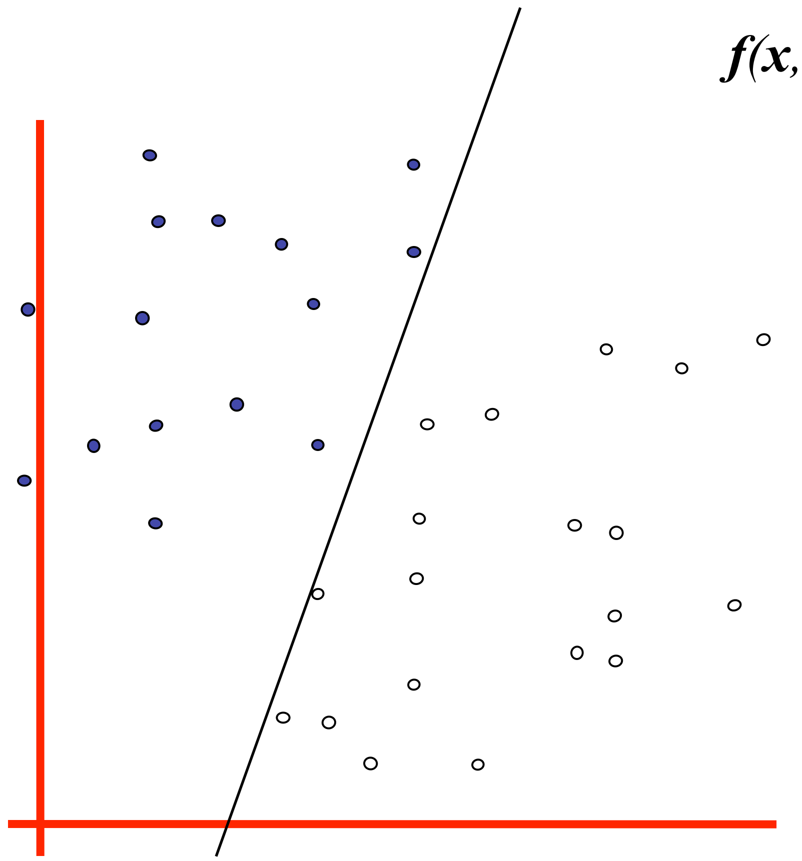


How would you classify this data?

Linear Classifiers



$f(\mathbf{x},\mathbf{w},b) = \mathit{sign}(\mathbf{w}.\ \mathbf{x} - b)$

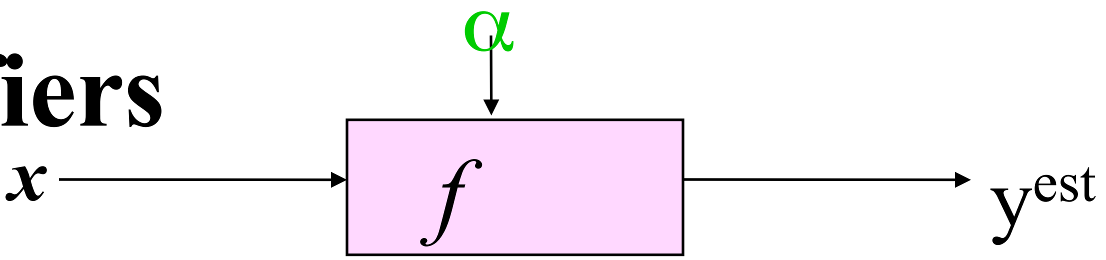


● denotes +1

○ denotes -1

How would you classify this data?

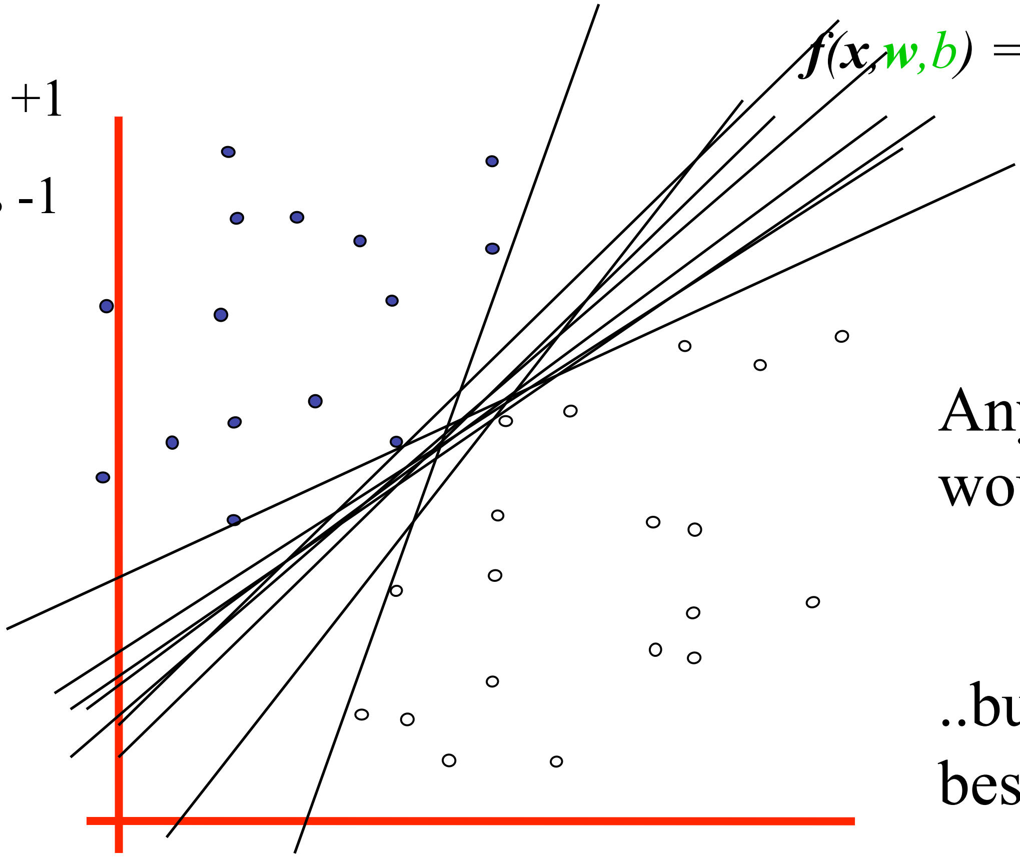
Linear Classifiers



$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

● denotes +1

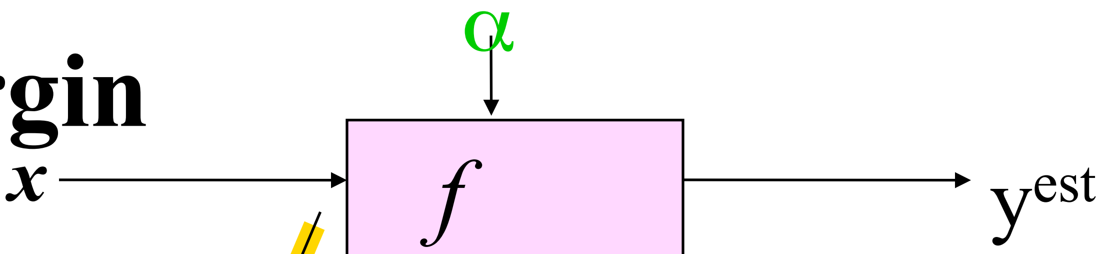
○ denotes -1



Any of these would be fine..

..but which is best?

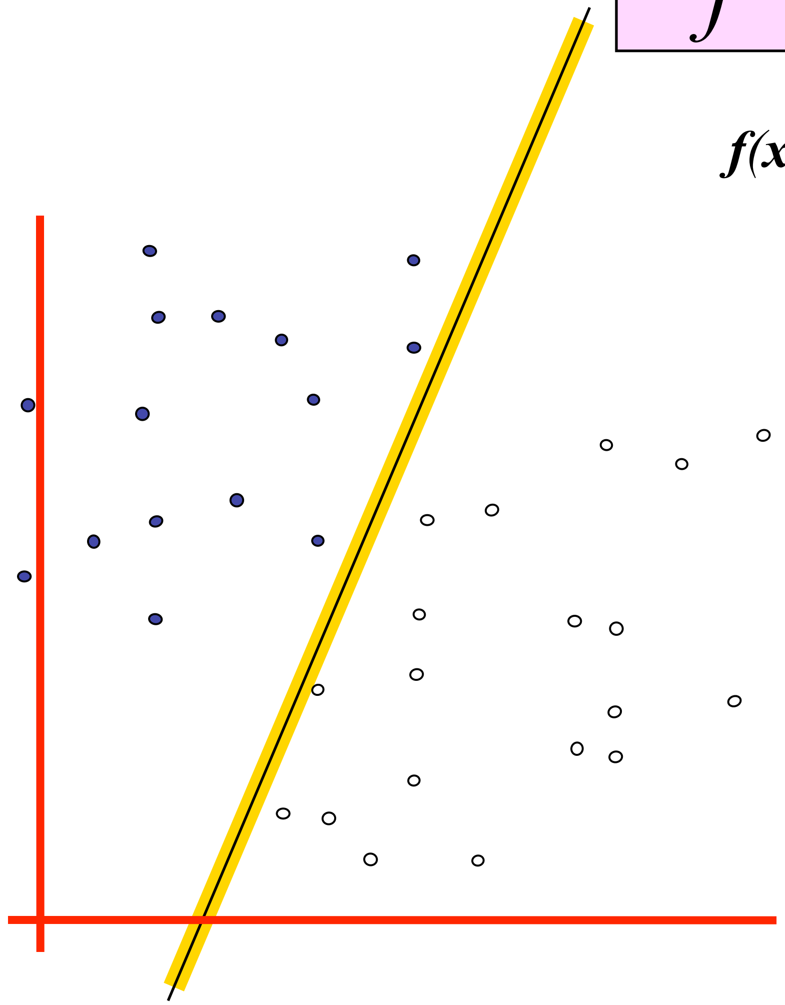
Classifier Margin



$f(\boldsymbol{x},\boldsymbol{w},b) = sign(\boldsymbol{w}.\ \boldsymbol{x} - b)$

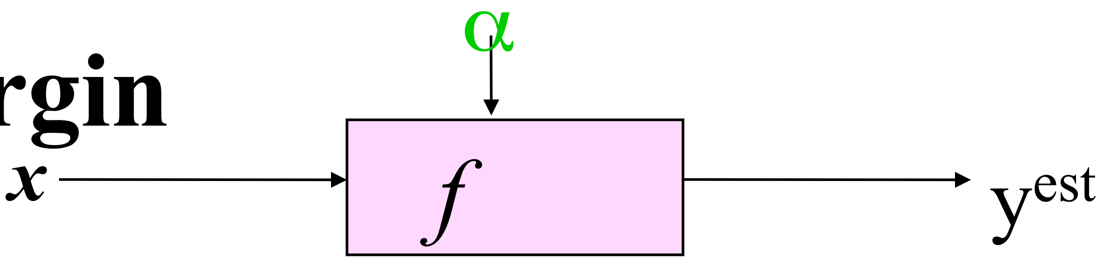
● denotes +1

○ denotes -1

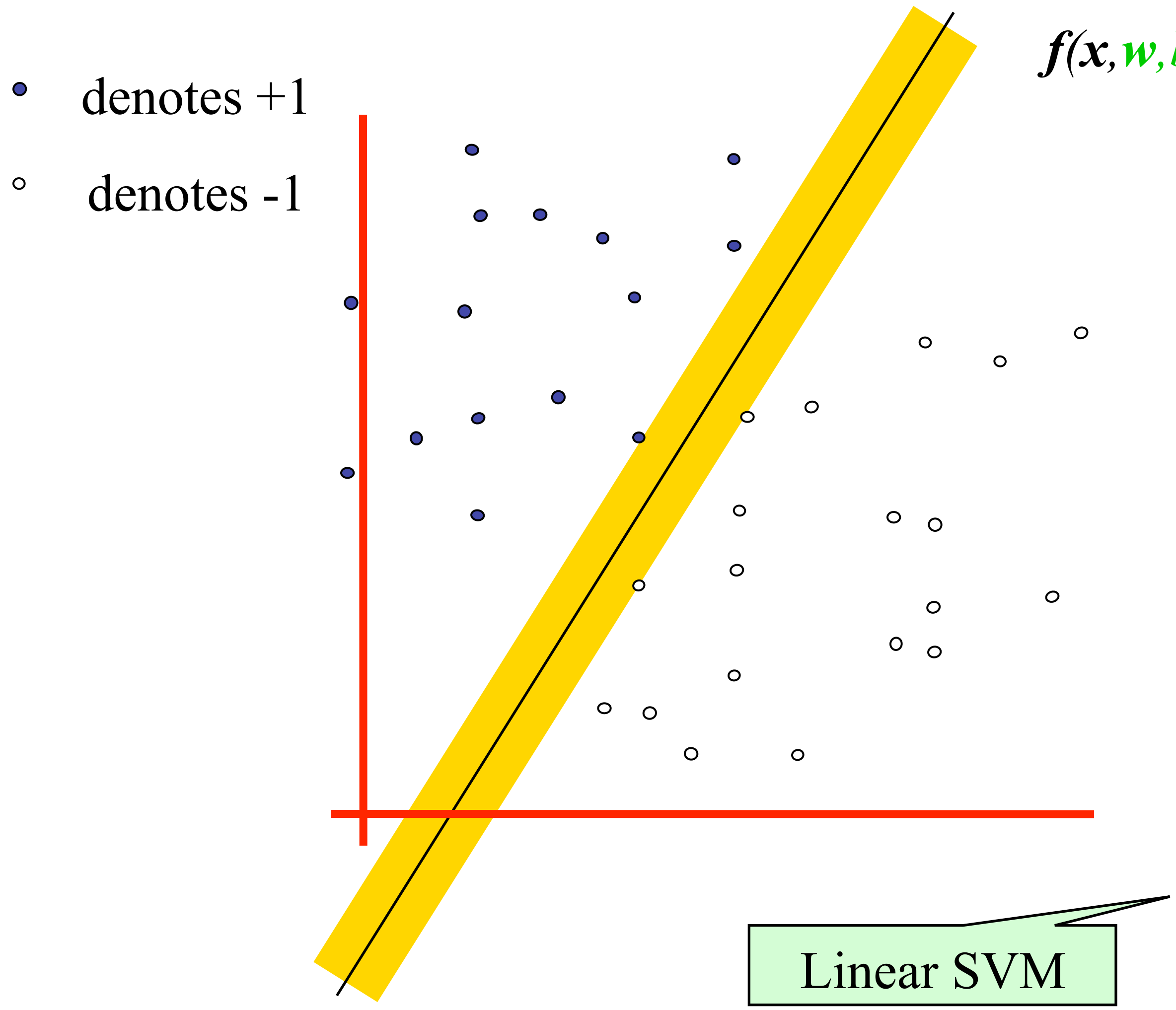


Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

Maximum Margin



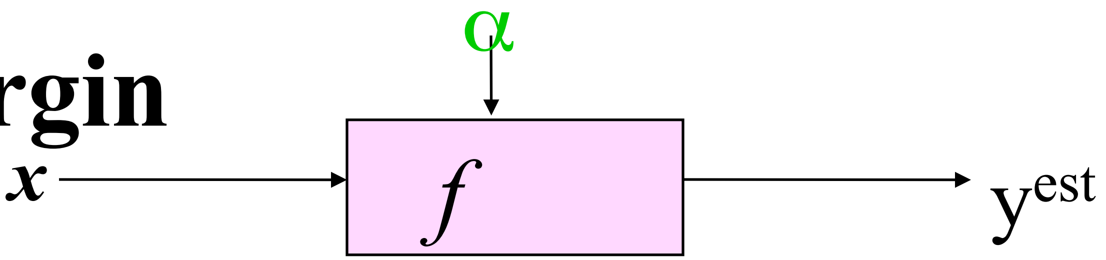
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$



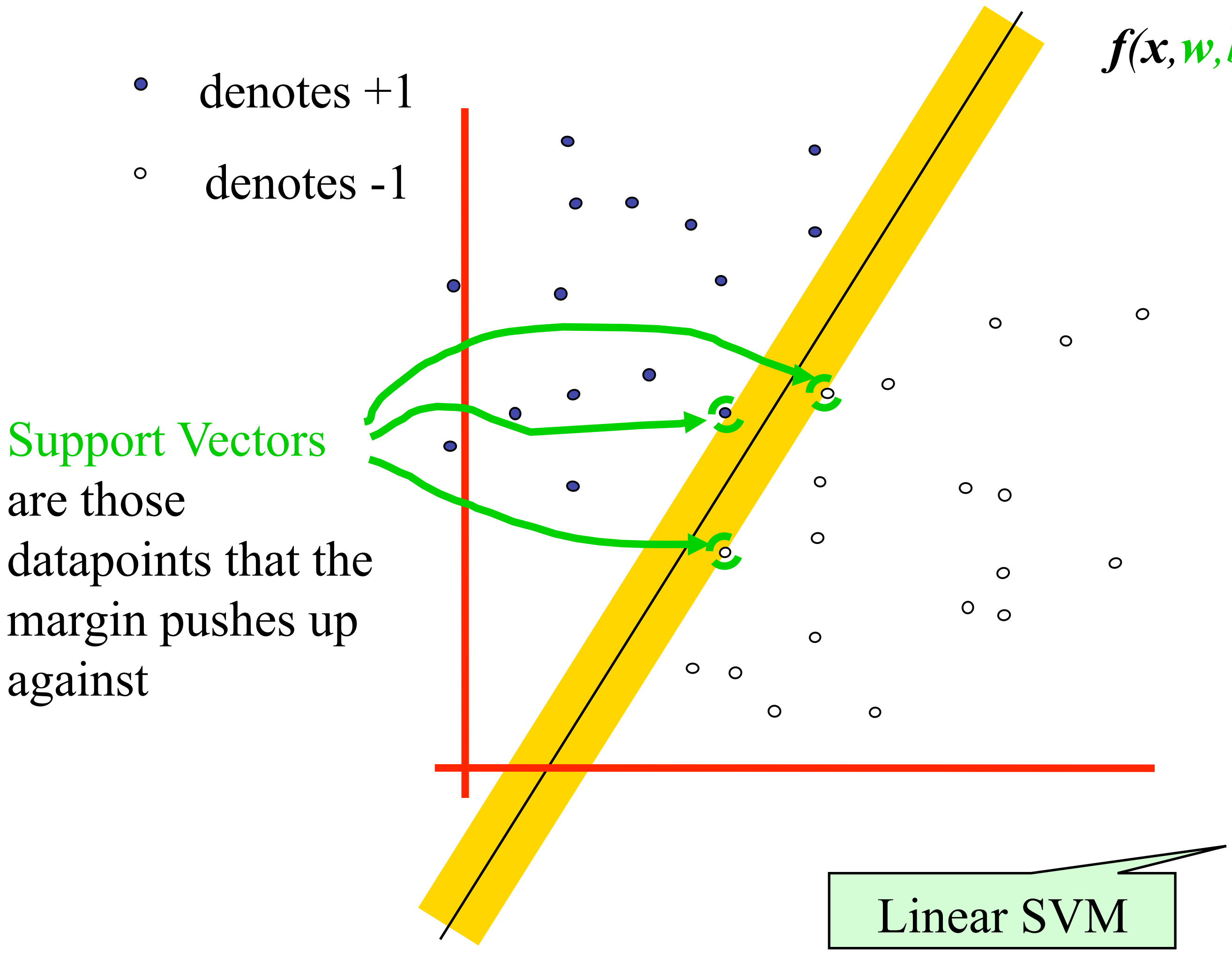
The maximum margin linear classifier is the linear classifier with the, um, maximum margin

This is the simplest kind of SVM (Called an LSVM)

Maximum Margin



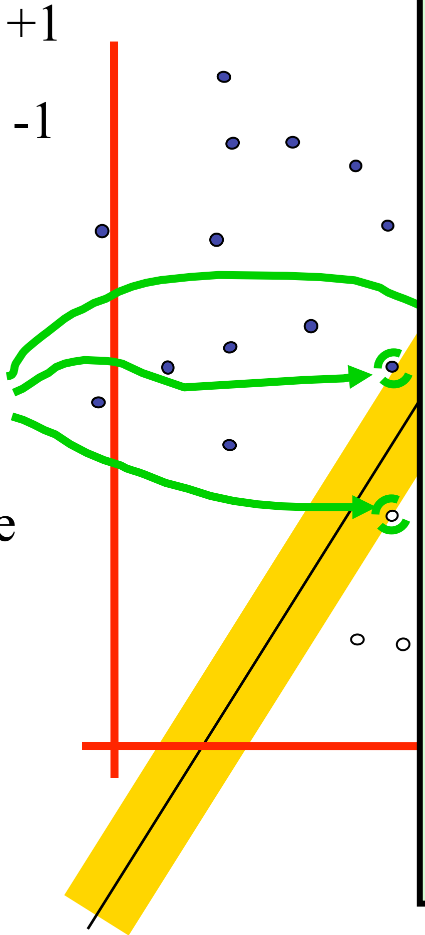
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\cdot \mathbf{x} - b)$


● denotes +1
○ denotes -1
Support Vectors are those datapoints that the margin pushes up against
Linear SVM

The maximum margin linear classifier is the linear classifier with the, um, maximum margin.

This is the simplest kind of SVM (Called an LSVM)

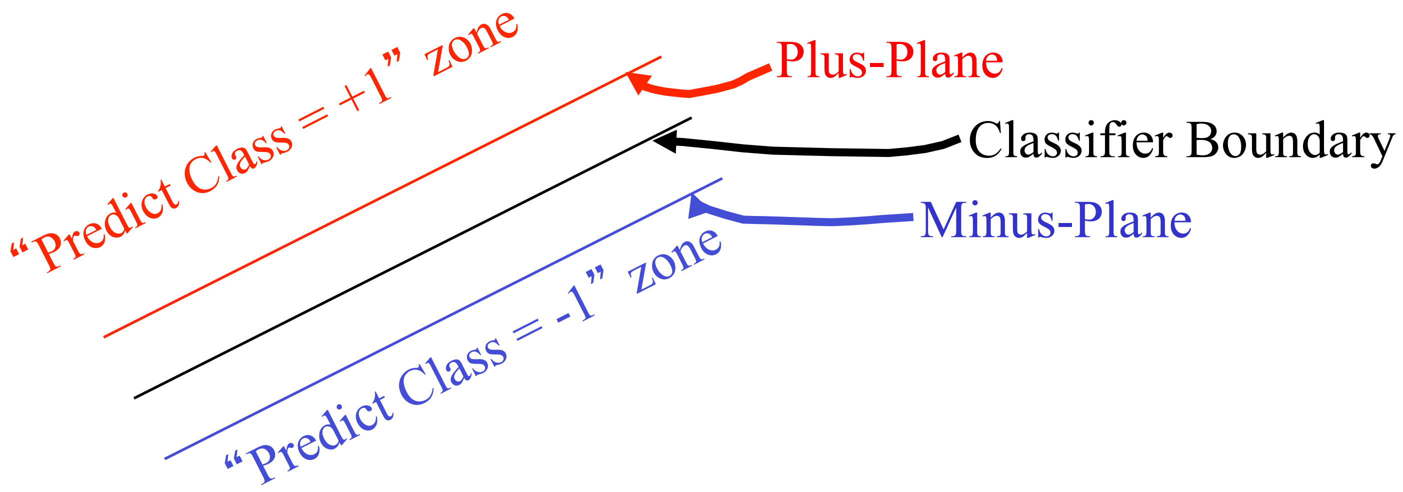
Why Maximum Margin?



Support Vectors are those datapoints that the margin pushes up against

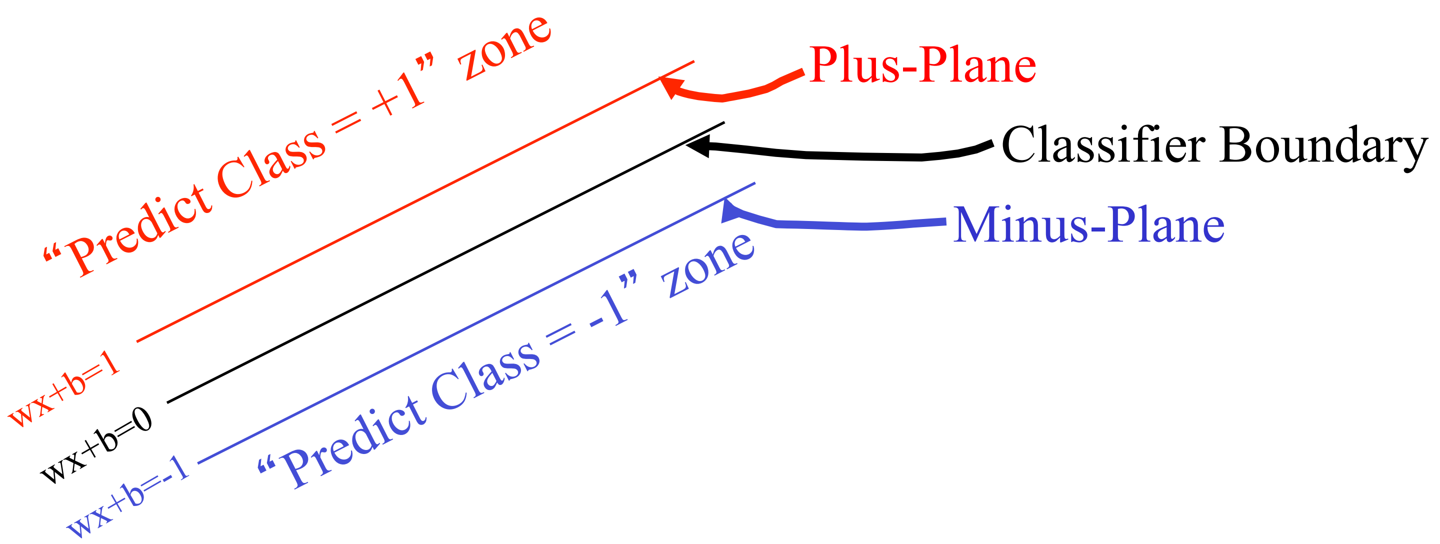
1. Intuitively this feels safest
2. If we've made a small error in the location of the boundary (it's been jolted in its perpendicular direction) this gives us least chance of causing a misclassification
3. LOOCV is easy since the model is immune to removal of any non-support-vector datapoints
4. There's some theory (using VC dimension) that is related to (but not the same as) the proposition that this is a good thing
5. Empirically it works very very well

Specifying a line and margin



- How do we represent this mathematically?
- …in *m* input dimensions?

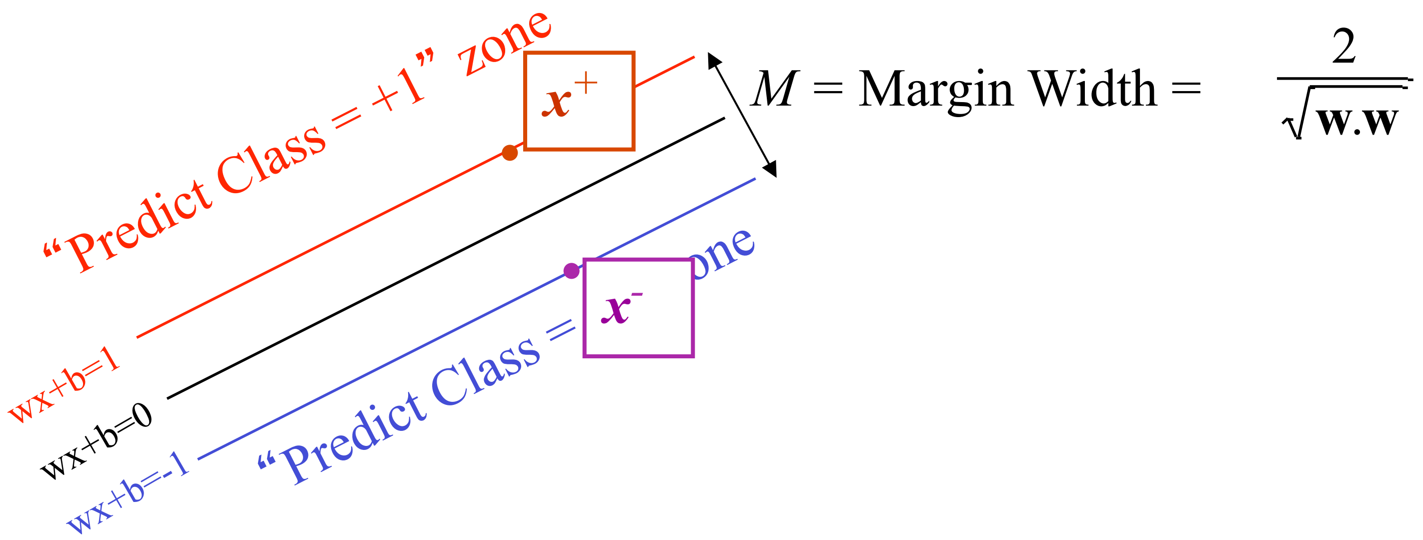
Specifying a line and margin



- Plus-plane = $\{ \mathbf{x} : \mathbf{w} \, . \, \mathbf{x} + b = +1 \}$
- Minus-plane = $\{ \mathbf{x} : \mathbf{w} \, . \, \mathbf{x} + b = -1 \}$

Classify as..	+1	if	$\mathbf{w} \, . \, \mathbf{x} + b >= 1$
	-1	if	$\mathbf{w} \, . \, \mathbf{x} + b <= -1$
	Universe explodes	if	$-1 < \mathbf{w} \, . \, \mathbf{x} + b < 1$

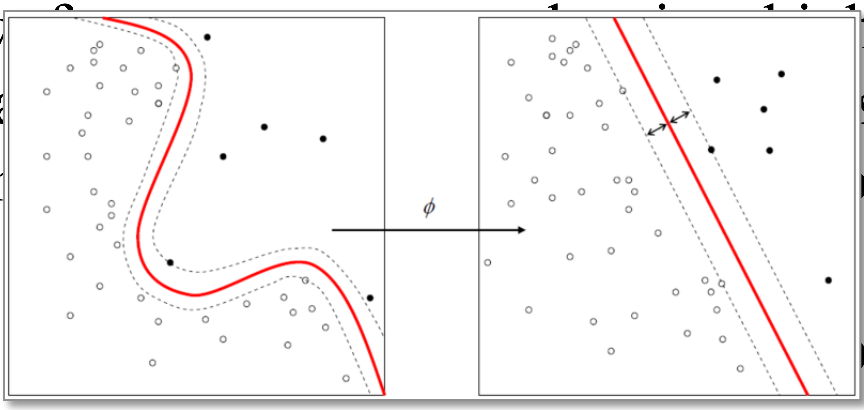
Learning the Maximum Margin Classifier


"Predict Class = +1" zone
x^+
M = Margin Width = $\frac{2}{\sqrt{\mathbf{w}.\mathbf{w}}}$
wx+b=1
wx+b=0
wx+b=-1
"Predict Class = -1" zone
x^-

- Given a guess of ***w*** and *b* we can
  - Compute whether all data points in the correct half-planes
  - Compute the width of the margin
- Write a program to search the space of **w**s and *b*s to find widest margin matching all the datapoints.
- *How?* -- Gradient descent? Simulated Annealing? Matrix Inversion? EM? Newton's Method?

Learning SVMs

- Trick #1: Just find the points that would be closest to the optimal separating plane ("support vectors") and work directly from those instances
- Trick #2: Represent as a **quadratic optimization problem**, and use quadratic programming techniques
- Trick #3 ("kernel trick"):
  - Instead of using the rav n- dimensional feature sp s *functions* (e.g., polynor of the base features)
  - Find separating plane / ace
  - Voila: A nonlinear classifier!



SVM Performance

- Can handle very large features spaces (e.g., 100K features)
- Relatively fast
- Anecdotally they work very very well indeed
- Example: They are currently the best-known classifier on a well-studied hand-written-character recognition benchmark

Binary vs. multi classification

- SVMs can only do binary classification
- Two approaches to multi classification:
  - One-vs-all: can turn an n-way classification into n binary classification tasks
    - E.g., for zoo problem, do mammal vs. not-mammal, fish vs. not-fish, …
    - Pick the one that results in the highest score
  - N*(N-1)/2 One-vs-one classifiers that vote on results
    - Mammal vs. fish, mammal vs. reptile, etc…

Feature Engineering for Text Classification

- Typical features: words and/or phrases along with term frequency or (better) TF-IDF scores
- ΔTFIDF amplifies the training set signals by using the ratio of the IDF for the negative and positive collections
- Results in a significant boost in accuracy

Text: The quick brown fox jumped over the lazy white dog.
Features: the 2, quick 1, brown 1, fox 1, jumped 1, over 1, lazy 1, white 1, dog 1, the quick 1, quick brown 1, brown fox 1, fox jumped 1, jumped over 1, over the 1, lazy white 1, white dog 1

ΔTFIDF BoW Feature Set

- Value of feature t in document d is $$C_{t,d} * \log_2\left(\frac{N_t}{P_t}\right)$$
- Where
  - $C_{t,d}$ = count of term t in document d
  - N_t = number of negative labeled training docs with term t
  - P_t = number of positive labeled training docs with term t
- Normalize to avoid bias towards longer documents
- Gives greater weight to rare (significant) words
- Downplays very common words
- Similar to Unigram + Bigram BoW in other aspects

Example: ΔTFIDF vs TFIDF vs TF

SPECIAL · EDITION
NICOLAS CAGE
MEG RYAN
SHE DIDN'T BELIEVE IN ANGELS UNTIL SHE FELL IN LOVE WITH ONE.
CITY OF ANGELS
DVD

15 features with highest values for a review of *City of Angels*

Δtfidf	tfidf	tf
, city	angels	,
cage is	angels is	the
mediocrity	, city	.
criticized	of angels	to
exhilarating	maggie ,	of
well worth	city of	a
out well	maggie	and
should know	angel who	is
really enjoyed	movie goers	that
maggie ,	cage is	it
it's nice	seth ,	who
is beautifully	goers	in
wonderfully	angels ,	more
of angels	us with	you
Underneath the	city	but

Improvement over TFIDF (Uni- + Bi-grams)

- **Movie Reviews:** 88.1% Accuracy vs. 84.65% at 95% Confidence Interval
- **Subjectivity Detection** (Opinionated or not): 91.26% vs. 89.4% at 99.9% Confidence Interval
- **Congressional Support for Bill** (Voted for/ Against): 72.47% vs. 66.84% at 99.9% Confidence Interval
- **Enron Email Spam Detection**: (Spam or not): 98.917% vs. 96.6168 at 99.995% Confidence Interval
- All tests used 10 fold cross validation
- At least as good as mincuts + subjectivity detectors on movie reviews (87.2%)