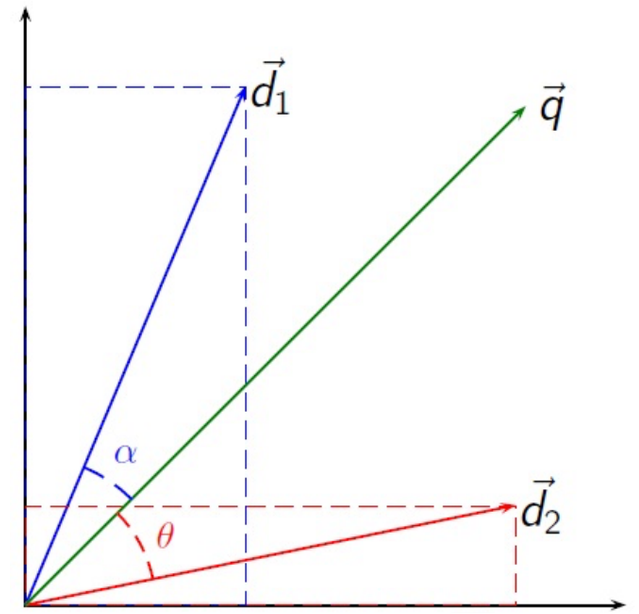
Learning to Rank

pointwise, pairwise and listless approaches

Road so Far - Ranking in IR

- Ranking documents important for information overload, quickly finding documents which are "relevant" for the query
- Interpretations and Modelling of relevance
  - Geometric Interpretation — Vector Space Similarity, Okapi BM25
  - Probabilistic interpretation — Probabilistic IR, Statistical LM



$$sim(d,q) = \frac{\sum_{i \in V} w_d(i).w_q(i)}{\sqrt{\sum_{i \in V} w_d(q_i)^2}\sqrt{\sum_{i \in V} w_q(i)^2}}$$

$$P(D|Q) \propto P(Q \mid D).P(D)$$

$$P(Q|D) = \prod_{w_i \in Q} \lambda.P(w_i|D) + (1-\lambda).P(w_i|C)$$

doc. contrib. corpus contrib.

laplace smoothing

$$P(Q|D) = \prod_{w_i \in Q} \frac{tf(w_i;D) + \mu.P(w_i|C)}{|D| + \mu}$$

doc. contrib. corpus contrib.

dirichlet smoothing

Other Ranking features

- Link analysis - Page rank, HITS
- Temporal ranking
  - Recency ranking
  - Historical features

How do we know which ranking method is better than the other?

- Query similarity to the document title
- Any other features that you can think of ?
- Anchor text similarity
- proximity of query terms in the document

How do you design a ranking function if you have many such features ?

Evaluation and Relevance Judgements

- Construct a test set containing a large number of (randomly sampled) queries, their associated documents, and relevance judgment for each (query, document) pair.
- Relevance judgement is the assessment of the relevance of documents for a query
- **Degree of relevance**
  - Binary: relevant vs. irrelevant
  - Multiple ordered categories: Perfect > Excellent > Good > Fair > Bad
- **Pairwise preference**
  - Document A is more relevant than document B
- **Total order**
  - Documents are ranked as {A,B,C,..} according to their relevance

Learning for IR

- What if we can learn a model from a set of features ?
  - Focus on developing interesting and novel features and less on how to put them together

- Why wasn't learning approaches used earlier ?
  - Limited training data
  - Machine learning techniques were not good enough
  - Traditional IR methods had small number of features and worked well
    - small features => easily hand tunable
  - Not many specialised IR problems

Supervised learning

- Given a set of training examples how do we learn a model to predict, classify, rank objects
  - classical tasks: regression, classification
  - Terminology: Input space, output space, hypothesis, loss function
  - **Input space:** (query, document) pairs
  - **Output space:** {relevant, nonrelevant}

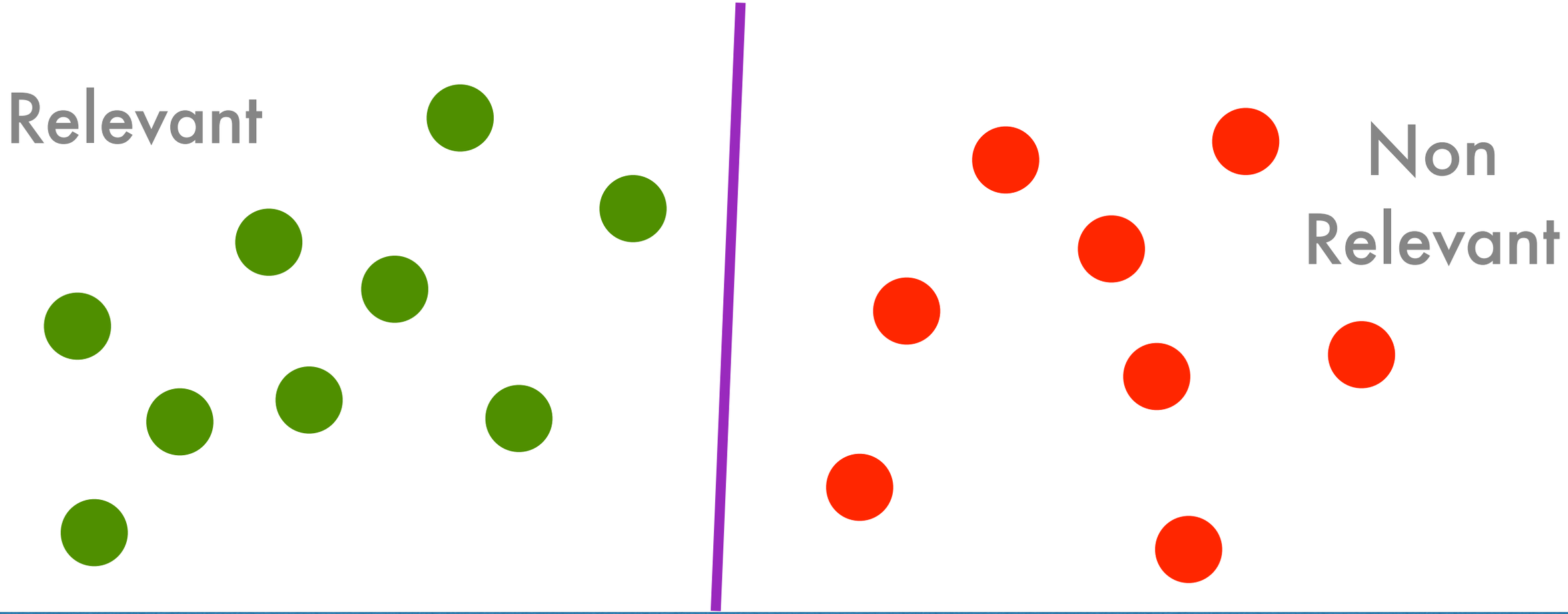
example	docID	query	cosine score	ω	judgment
Φ_1	37	linux operating system	0.032	3	*relevant*
Φ_2	37	penguin logo	0.02	4	*nonrelevant*
Φ_3	238	operating system	0.043	2	*relevant*
Φ_4	238	runtime environment	0.004	2	*nonrelevant*
Φ_5	1741	kernel layer	0.022	3	*relevant*
Φ_6	2094	device driver	0.03	2	*relevant*
Φ_7	3191	device driver	0.027	5	*nonrelevant*

Learning to Rank Approaches

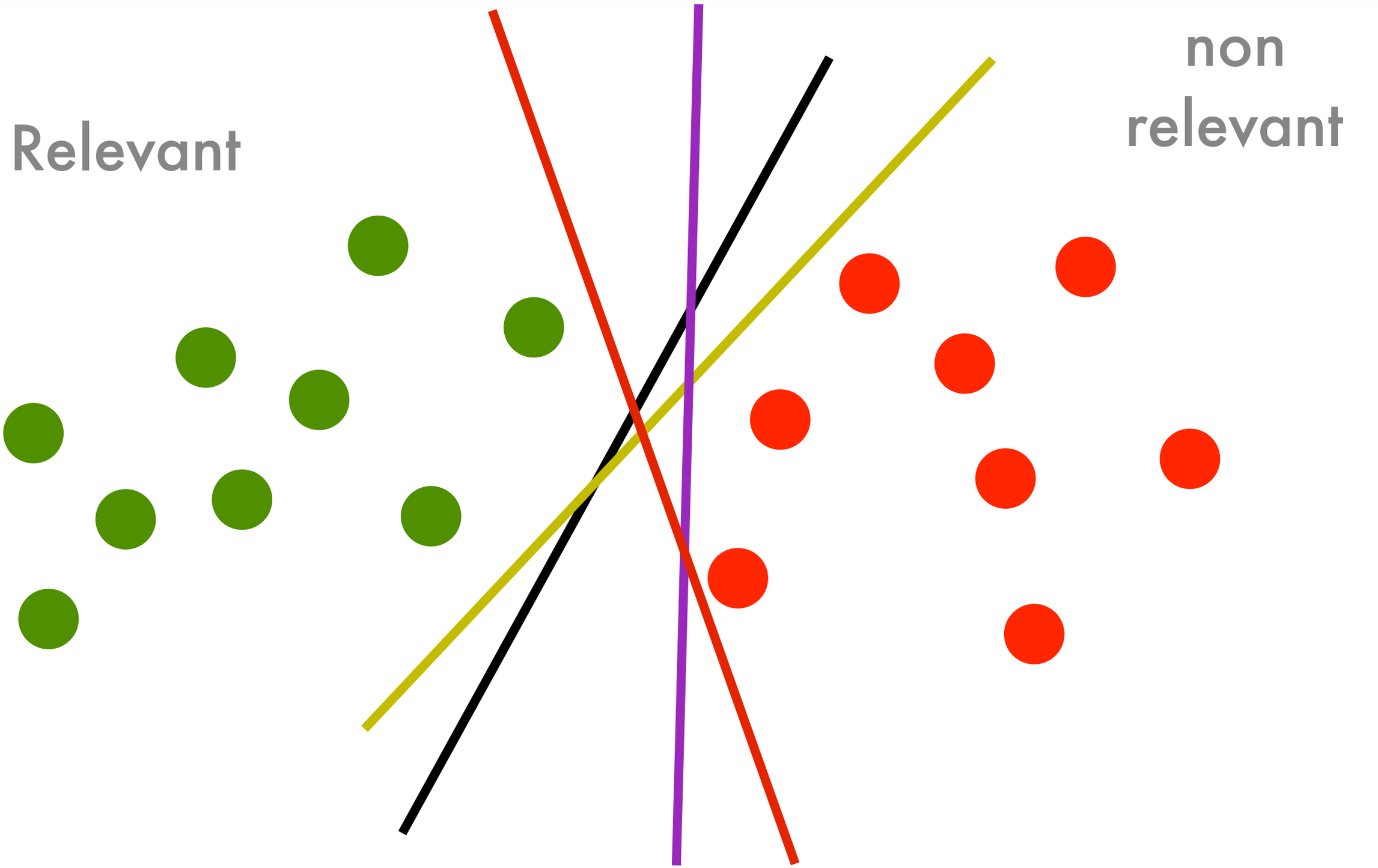
- **Pointwise Approaches**
  - Input space: single documents **d1** — if a doc. is relevant to a query
  - Output space: scores or relevant classes
- **Pairwise Approaches**
  - Input space: **(d1, d2)** — pairs of documents s.t. d1 > d2
  - Output space: preferences (yes/no) for a given doc. pair
- **Listwise Approaches**
  - Input space: Document set **{d}**
  - Output space: Permutation — ranking of documents
  - Optimize directly a IR quality measure — MAP, NDCG. etc

Classification for IR

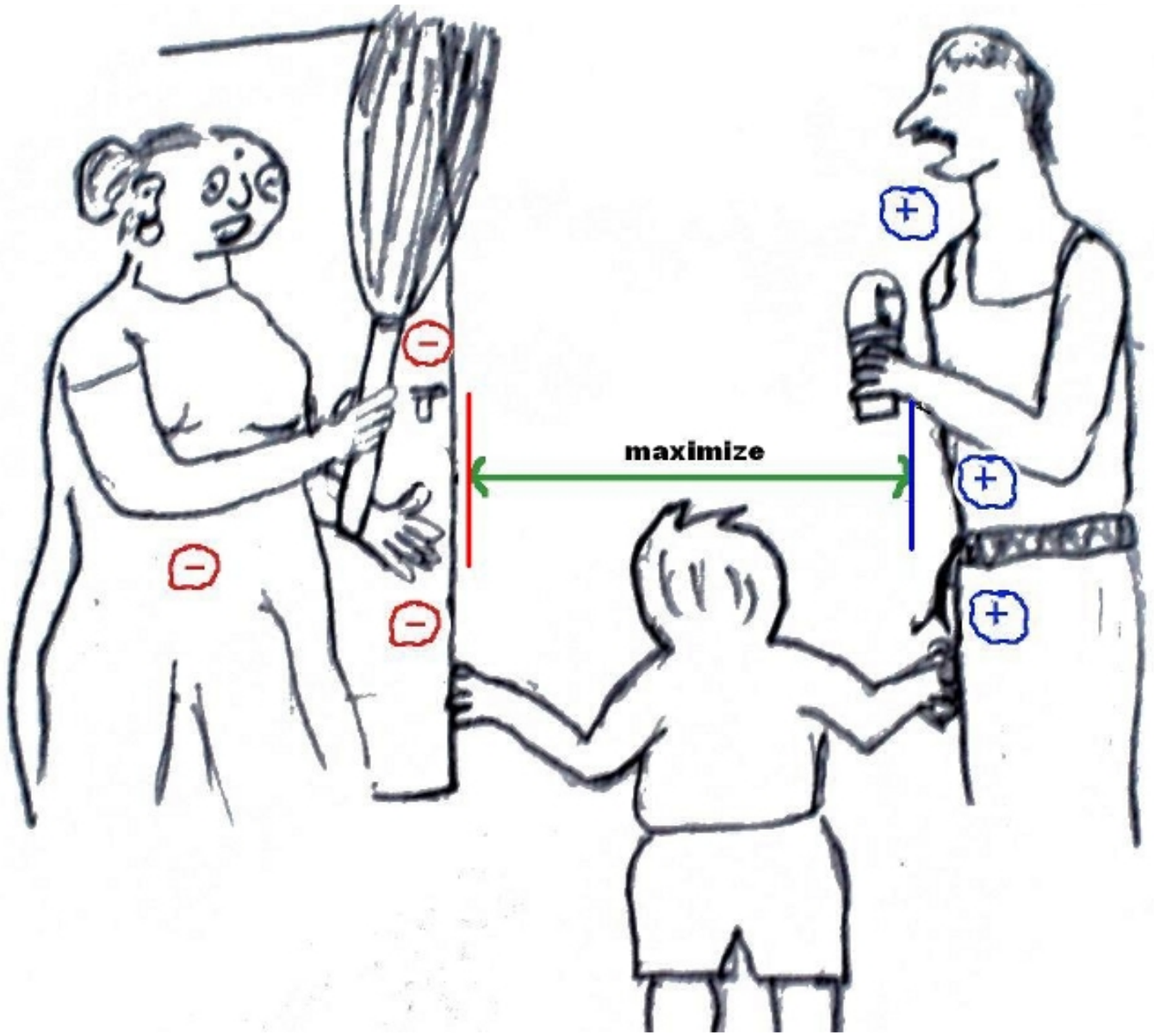
- Given a set of relevant and non-relevant documents find the **hypothesis** which best separates them
- Minimizes **loss -** e.g. number of misclassified documents (squared loss, hinge loss, cross entropy)
- Some classical classification approaches: Naive Bayes, decision trees, **SVMs**



Support Vector Machines



Support Vector Machines



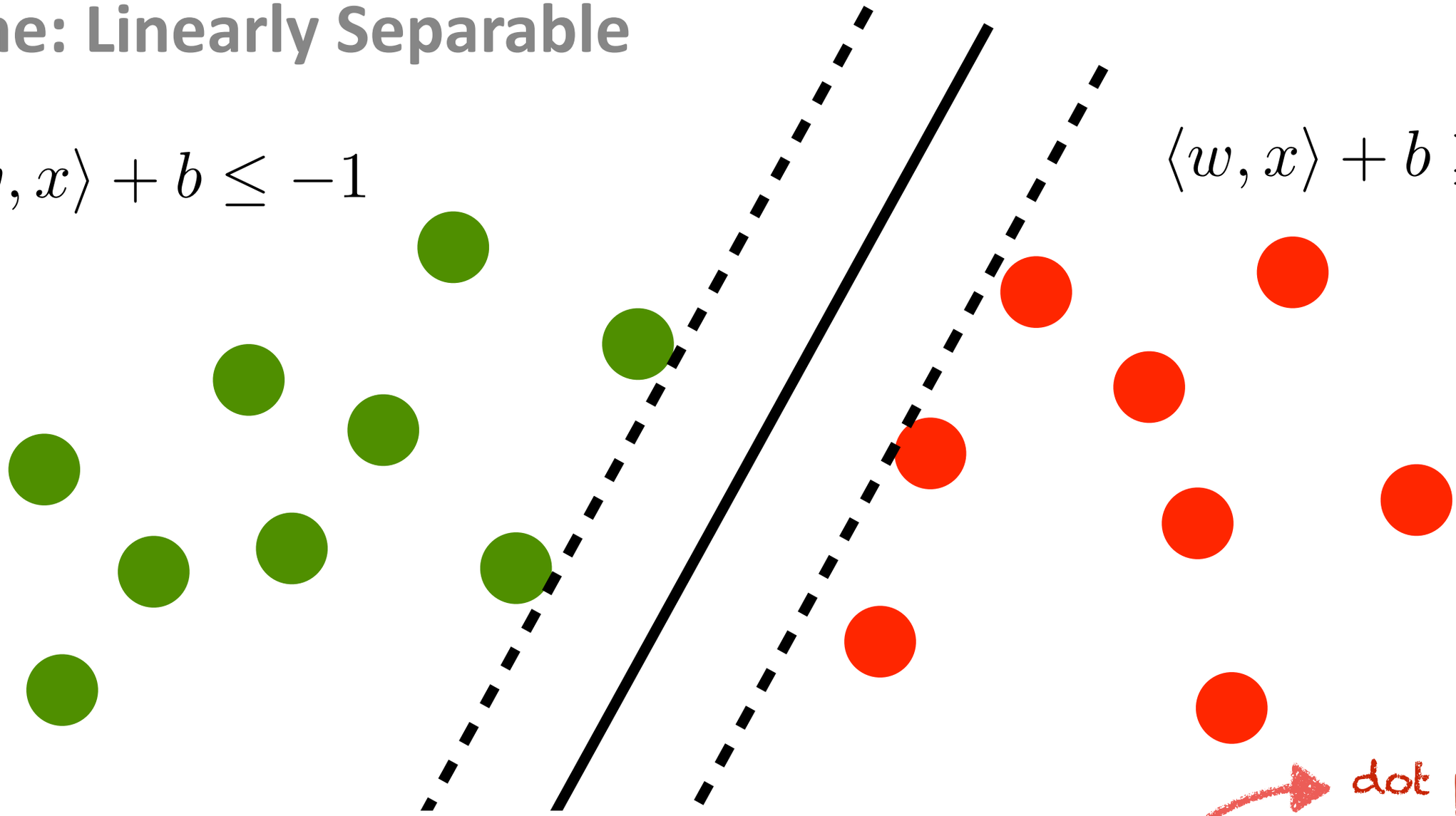
Large Margin Classifiers

Support Vector Machines

Assume: Linearly Separable

$$\langle w, x \rangle + b \leq -1$$

$$\langle w, x \rangle + b \geq 1$$



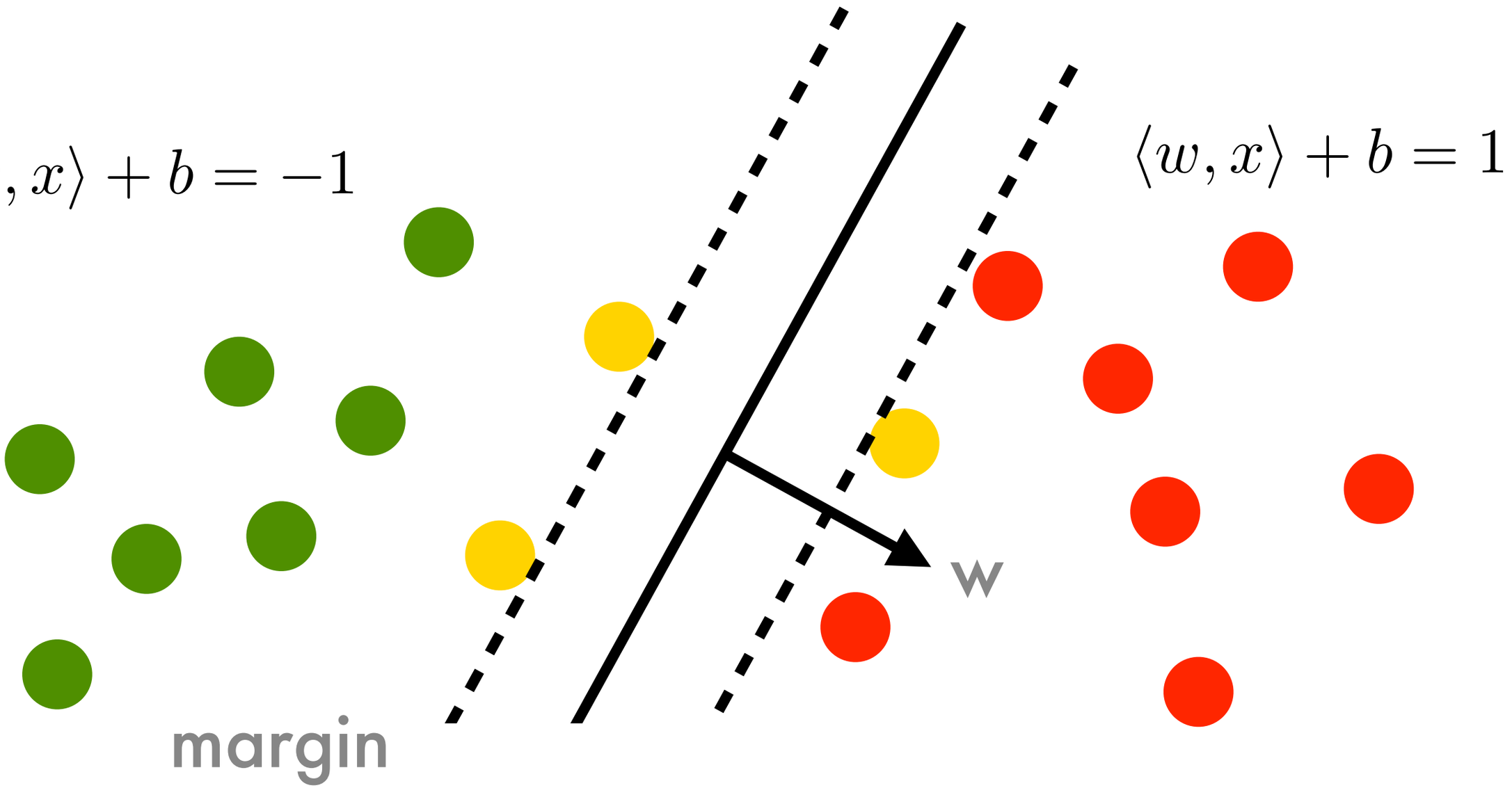
linear function $$f(x) = \langle w, x \rangle + b$$

dot product

normal vector for separator

feature vector for (q,d) pair

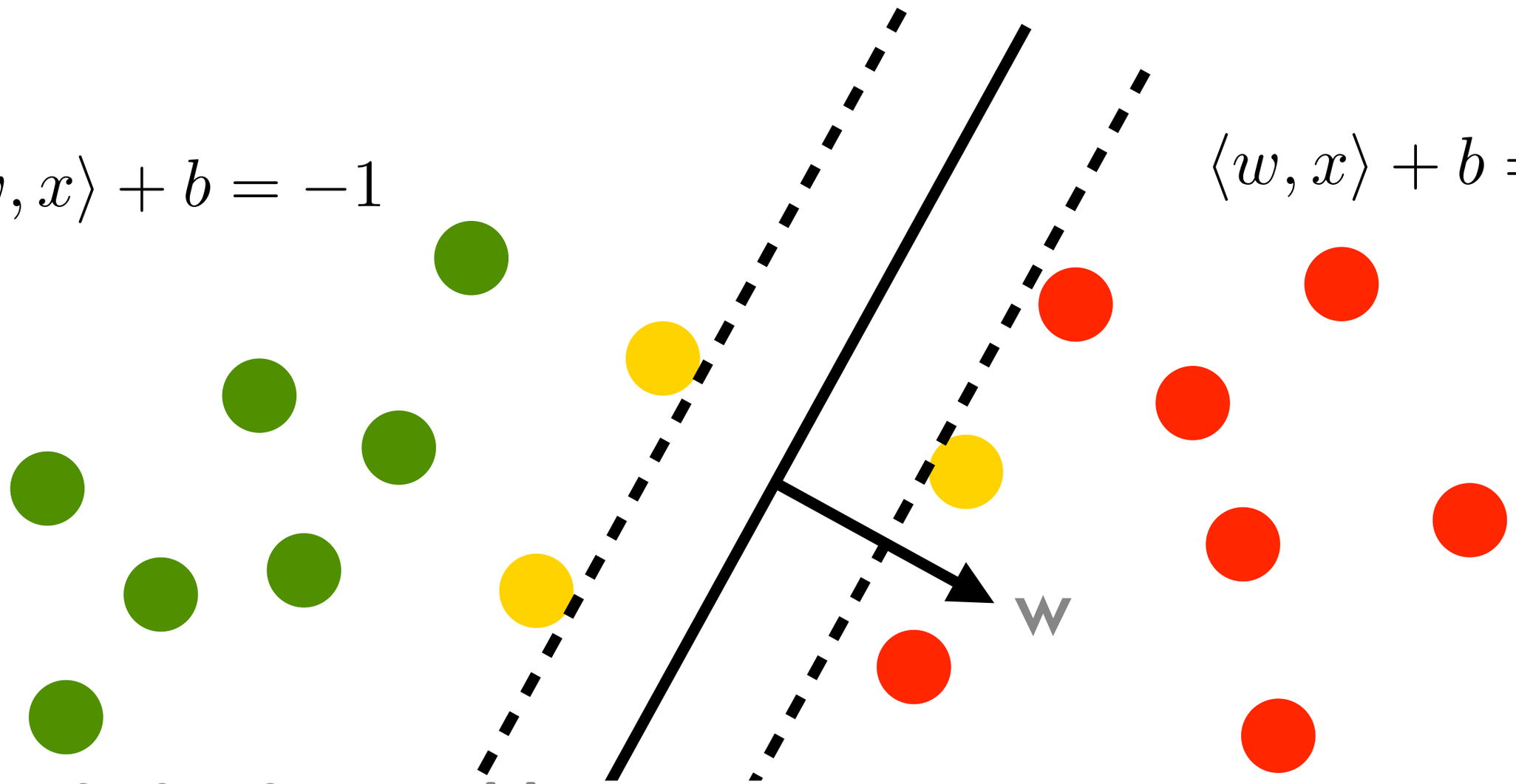
Support Vector Machines



$$\frac{\langle x_+ - x_-, w \rangle}{2\|w\|} = \frac{1}{2\|w\|}\left[[\langle x_+, w \rangle + b] - [\langle x_-, w \rangle + b]\right] = \frac{1}{\|w\|}$$

Support Vector Machines

$\langle w, x \rangle + b = -1$

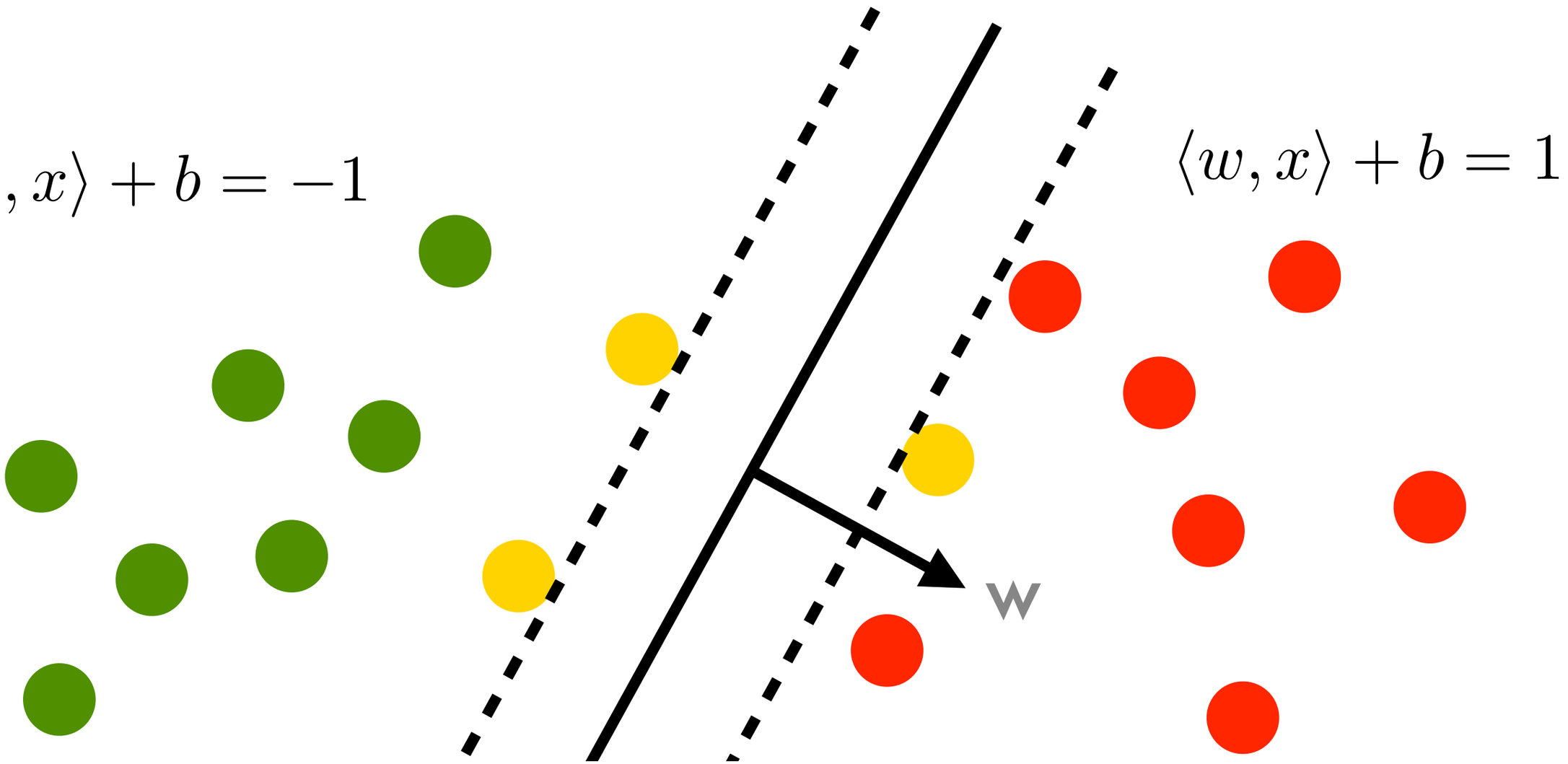


$\langle w, x \rangle + b = 1$

optimization problem

$$\underset{w,b}{\text{maximize}} \frac{1}{\|w\|} \text{ subject to } y_i \left[\langle x_i, w \rangle + b\right] \geq 1$$

Support Vector Machines

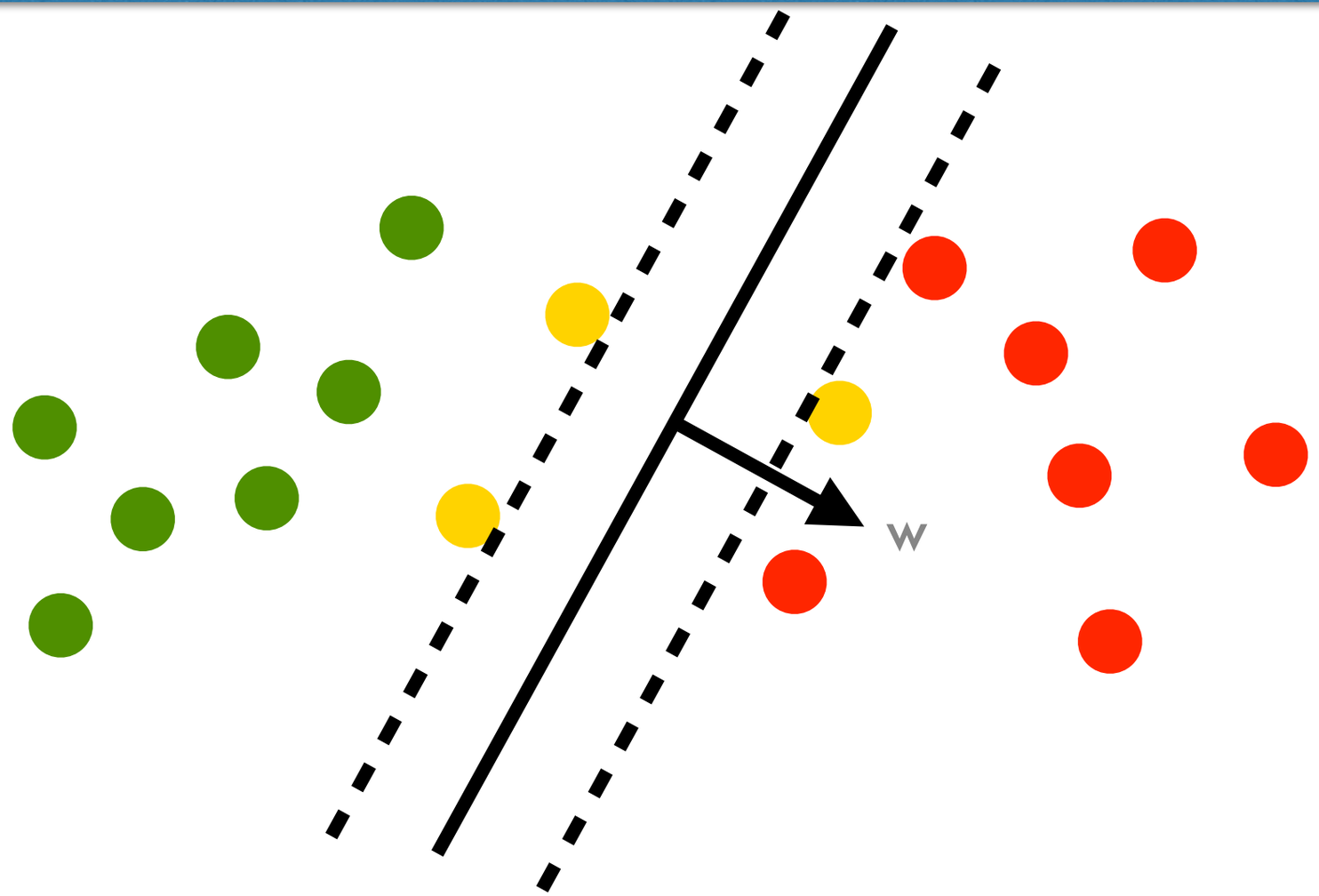


optimization problem

+1 if relevant , -1 o.w.

$$\underset{w,b}{\text{minimize}} \frac{1}{2} \|w\|^2 \text{ subject to } y_i \left[\langle x_i, w \rangle + b\right] \geq 1$$

Support Vector Machines

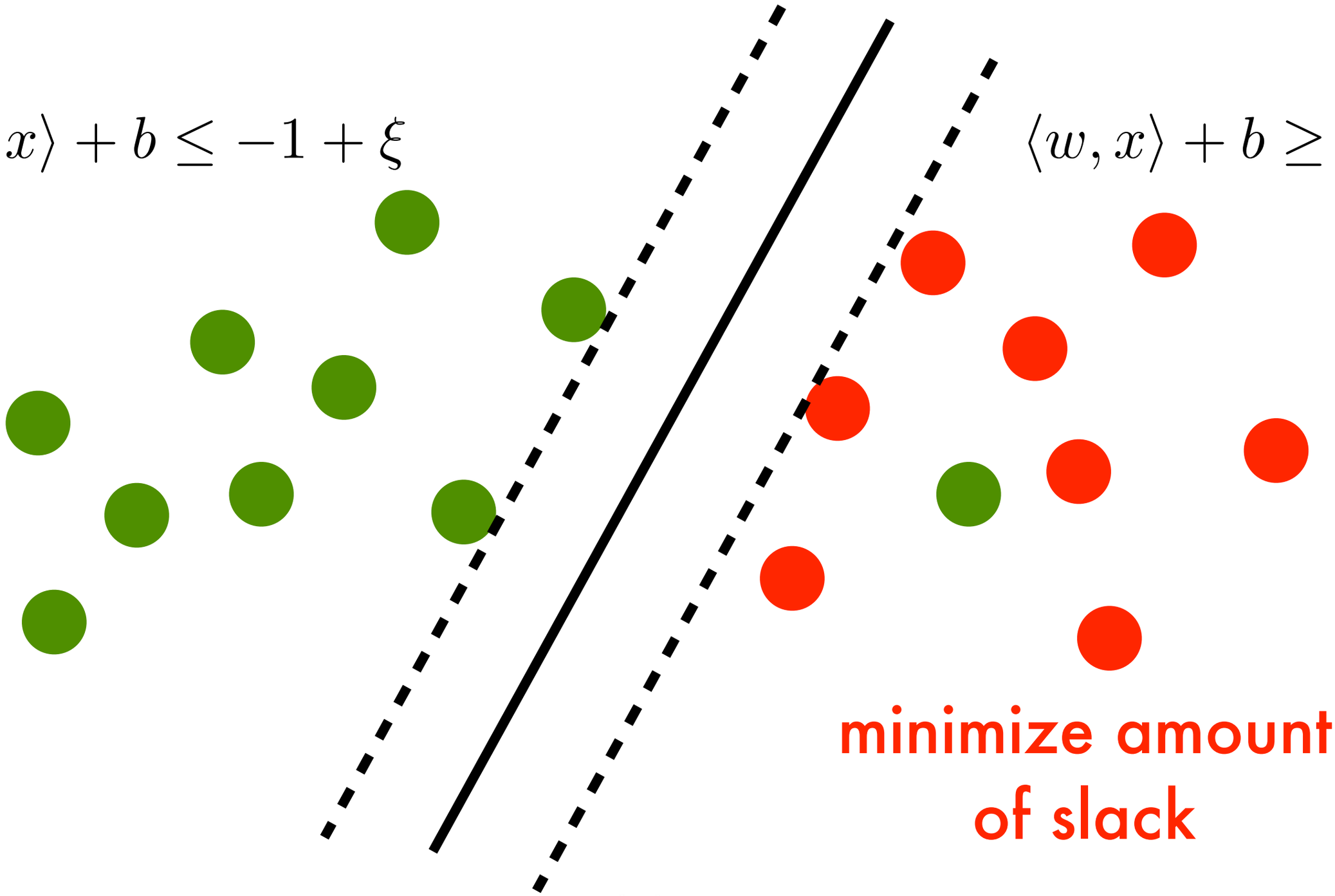


- Weight vector **w** as weighted linear combination of instances
- Only points on margin matter (ignore the rest and get same solution)
- Only inner products matter
  - Quadratic program
- Keeps instances away from the margin

Soft Margins

$\langle w, x \rangle + b \leq -1 + \xi$

$\langle w, x \rangle + b \geq 1 - \xi$



minimize amount of slack

Convex optimization problem

Soft Margins

- **Hard margin problem**

$$\underset{w,b}{\text{minimize}} \frac{1}{2} \|w\|^2 \text{ subject to } y_i \left[\langle w, x_i \rangle + b\right] \geq 1$$

- **With slack variables**

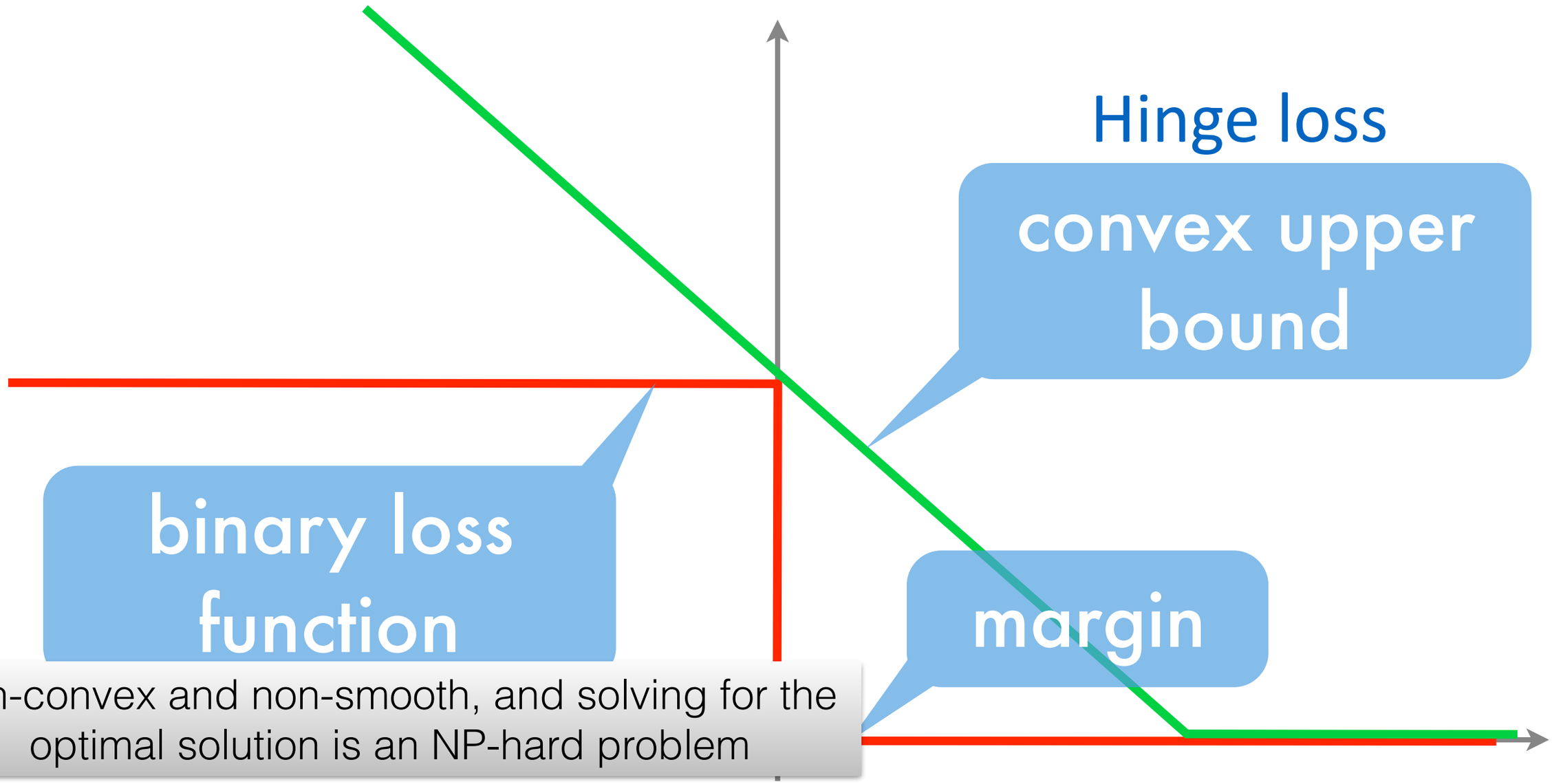
$$\underset{w,b}{\text{minimize}} \frac{1}{2} \|w\|^2 + C \sum_i \xi_i$$

$$\text{subject to } y_i \left[\langle w, x_i \rangle + b\right] \geq 1 - \xi_i \text{ and } \xi_i \geq 0$$

Whats the loss function here ?

Loss Function Perspective

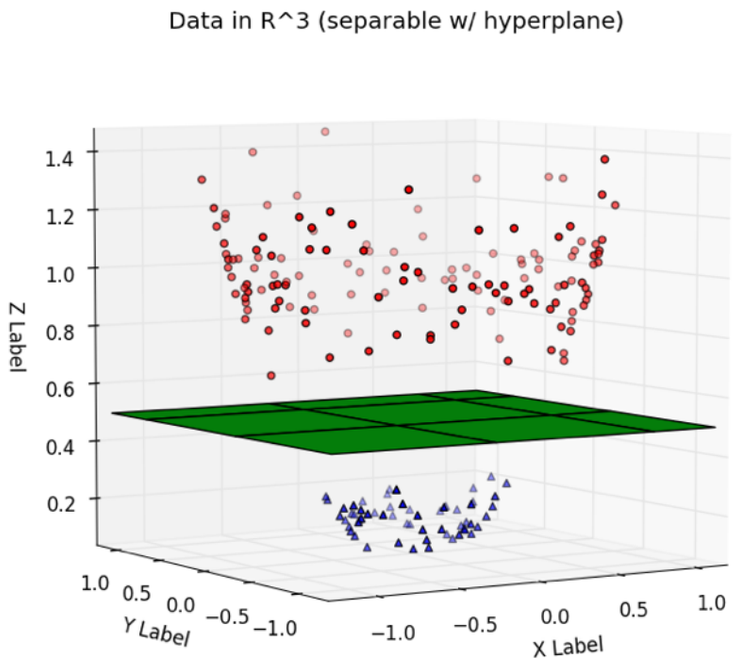
- Soft margin loss $\max(0, 1 - yf(x))$ Hinge loss
- Binary loss $\{yf(x) < 0\}$

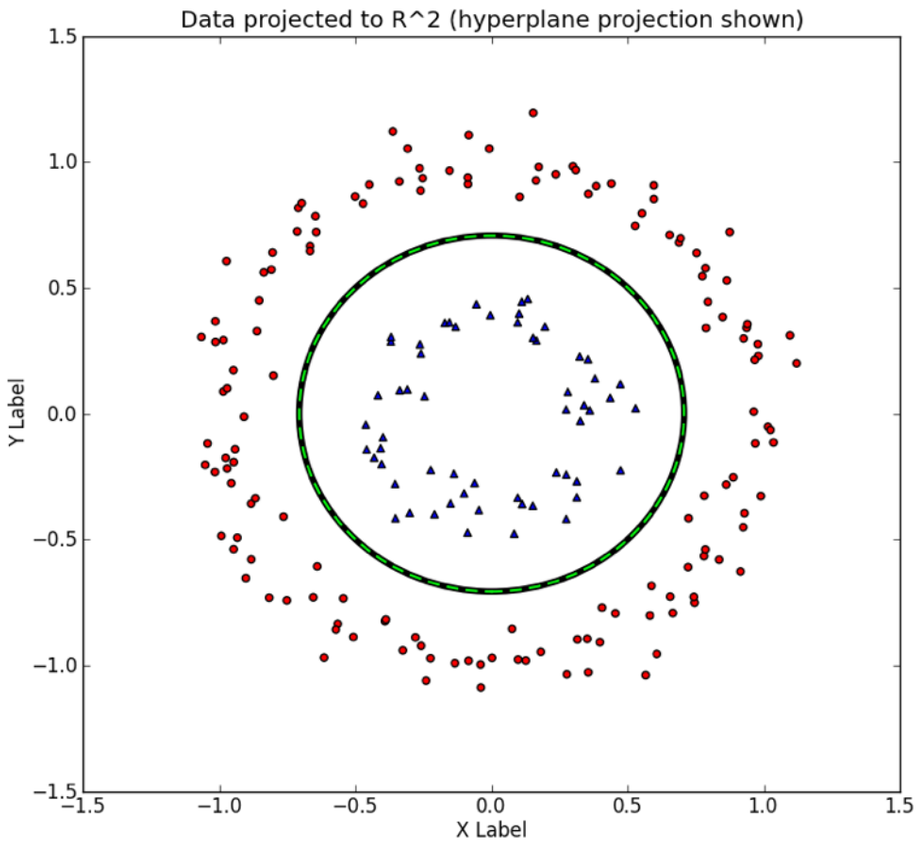


non-convex and non-smooth, and solving for the optimal solution is an NP-hard problem

Kernels

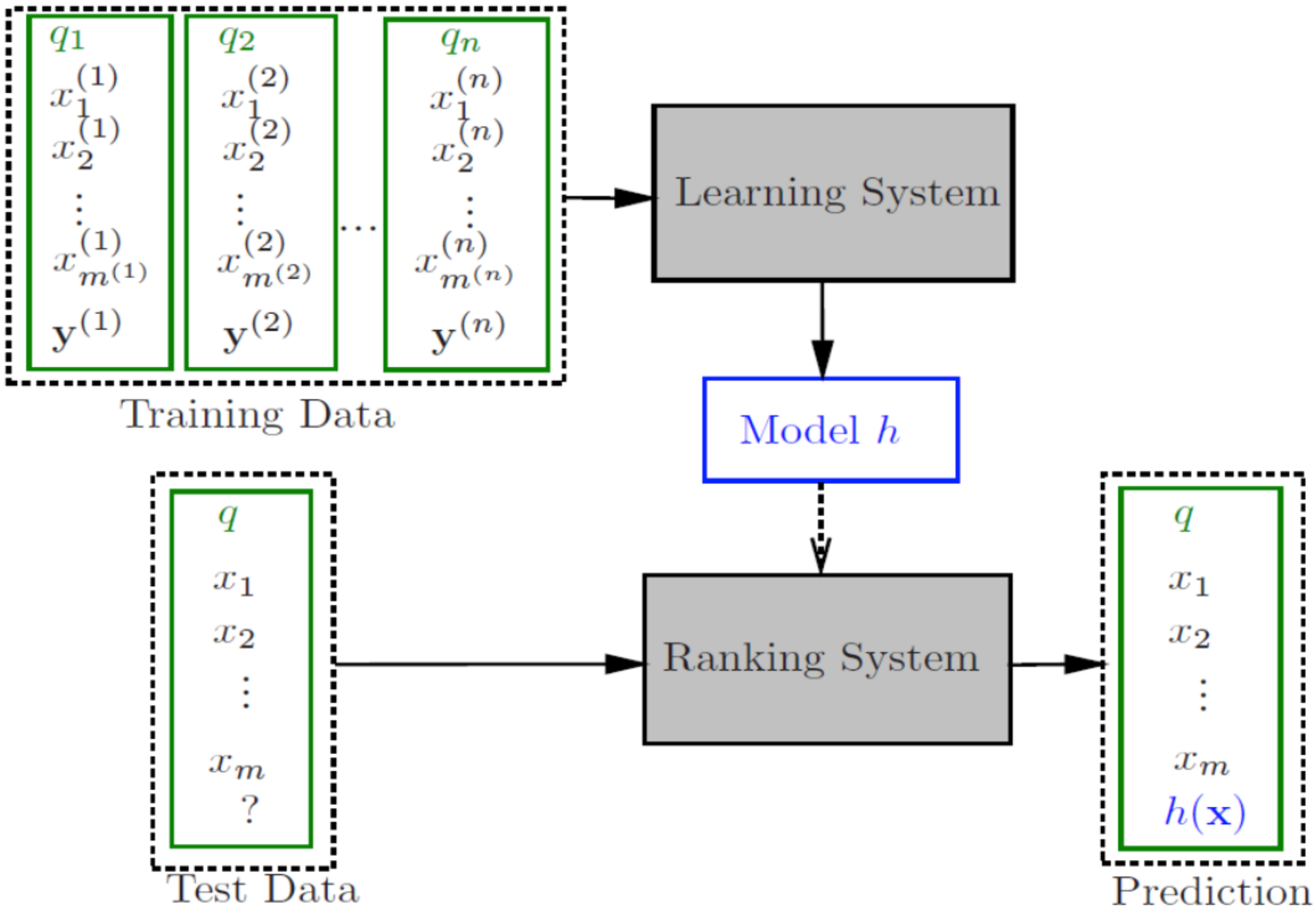
- **What if the points are not linearly separable?**





- **Kernel tricks - project the points in a higher dimensional space and find the dot product there.**
  - **Linear Kernel**
  - **RBF kernel**
  - **Polynomial kernel**

Learning to Rank Framework



Learning to Rank Approaches

- **Pointwise Approaches**
  - Input space: single documents **d1** — if a doc. is relevant to a query
  - Output space: scores or relevant classes
- **Pairwise Approaches**
  - Input space: **(d1, d2)** — pairs of documents s.t. d1 > d2
  - Output space: preferences (yes/no) for a given doc. pair
- **Listwise Approaches**
  - Input space: Document set **{d}**
  - Output space: Permutation — ranking of documents
  - Optimize directly a IR quality measure — MAP, NDCG. etc

Pointwise Ranking - large margin ranking

(A. Shashua and A. Levin, NIPS 2002)

Original SVM formulation

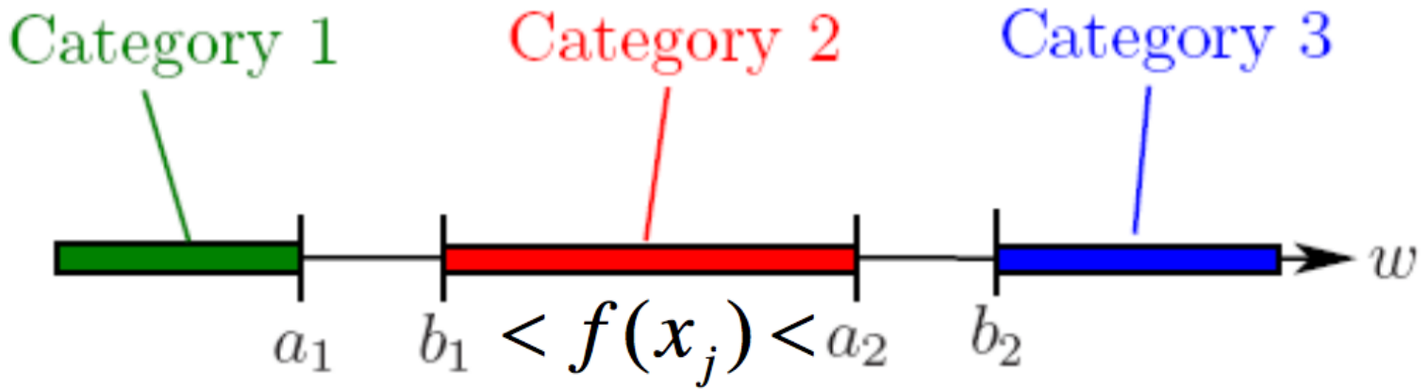
$$\underset{w,b}{\text{minimize}} \ \frac{1}{2}\|w\|^2 + C\sum_i \xi_i$$

margin

$$\text{subject to } y_i\left[\langle w, x_i\rangle + b\right] \geq 1 - \xi_i \text{ and } \xi_i \geq 0$$

Fixed Margin Strategy

- Margin: $\sum_k (b_k - a_k)$
- Constraints: Every document is correctly placed in the corresponding category



if $y_j = 2$

Problems with Pointwise ranking

- Some queries have large number of documents and dominate the learned hypothesis
- Does not optimise for the standard IR measures like NDCG, MAP, ERR etc
- Position of the documents in the ranked list is not modelled by the loss function
- Approaches might emphasise more on the unimportant documents
  - Sampling is essential (if non relevant documents >> relevant documents)

Pairwise Ranking

- Convert input into preferences
- Use preferences as input to learn the classes
- Define loss **L(.)** on the difference vector and minimise loss while learning **f(.)**

query

d1 (d1 > d2) (d1 > d5) (d1 > d6) (d1 > d3) .....

d2

d3 $(x_1 - x_2)$

d4

d5 $L(f; x_u, x_v, y_{u,v}) = \exp(-y_{u,v}(f(x_u) - f(x_v)))$

d6

[Rank Boost 2003]

Original SVM formulation

$$\underset{w,b}{\text{minimize}} \ \frac{1}{2}\|w\|^2 + C\sum_i \xi_i$$

$$\text{subject to } y_i\left[\langle w, x_i\rangle + b\right] \geq 1 - \xi_i \text{ and } \xi_i \geq 0$$

Ranking SVM

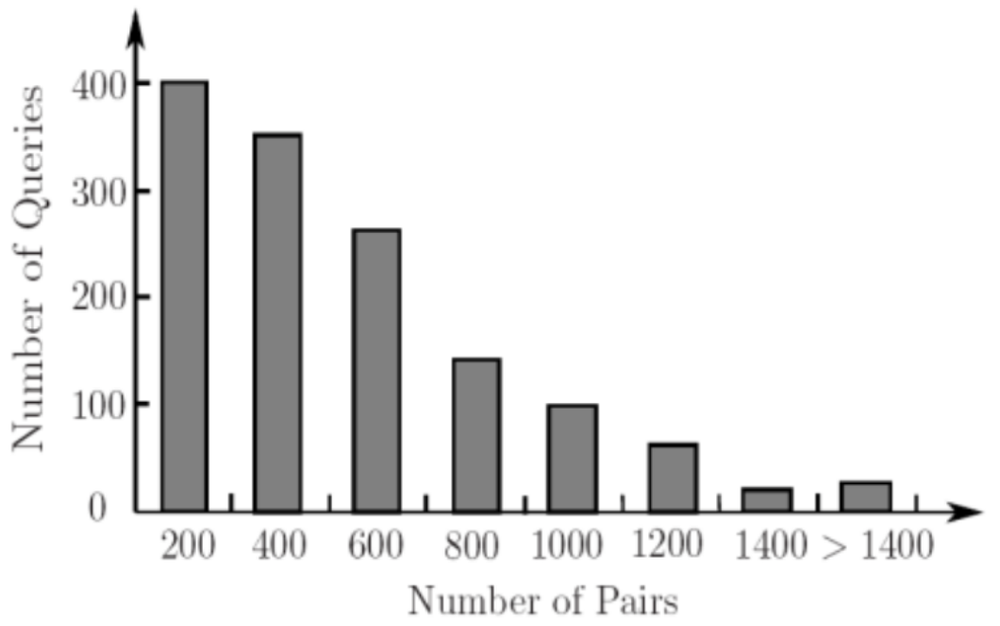
(R. Herbrich, T. Graepel, et al. , Advances in Large Margin Classifiers, 2000; T. Joachims, KDD 2002)

$$\underset{w,b}{\text{minimize}} \ \frac{1}{2}||w||^2 + C\sum_i\sum_{u,v} \xi^i_{u,v}$$

$$\text{subject to} \quad y_i[\langle w, (x^i_u - x^i_v)\rangle + b] \geq 1 - \xi^i_{u,v} \ and \ \xi^i_{u,v} \geq 0$$

Pros and Cons - Pairwise Approach

- Pros
  - One step closer to ranking documents compared to predicting class labels
- Cons
  - Distribution of document pairs is more skewed than distribution of document w.r.t queries
  - Some queries have more query pairs than others
  - Still does not optimise for IR measures
  - Rank ignorant — (d1 > d2) does not encode which ranks are being compared. Rank 1 vs Rank 2 or Rank 99 vs Rank 1000



Listwise Approaches

- Direct optimization of IR measures
  - Try to optimize IR evaluation measures, or at least something correlated to the measures.

NDCG, ERR, MAP

- Listwise loss minimization
  - Minimize a loss function defined on permutations, which is designed by considering the properties of ranking for IR.

Learning to Rank — In Action

extract features from document collection and queries

feature file

create model

Execute model on test data for results

Feature List of Microsoft Learning to Rank Datasets		
feature id	**feature description**	**stream**
1	covered query term number	body
2		anchor
3		title
4		url
5		whole document
6	covered query term ratio	body
7		anchor
8		title
9		url
10		whole document
11	stream length	body
12		anchor
13		title
14		url
15		whole document
16	IDF(Inverse document frequency)	body
17		anchor
18		title
19		url

Learning to Rank - in Action

Example of a feature file

```
0 qid:20140102 2:0.0 3:0.0 4:0.0 5:0.020997914892367332 6:0.0 7:0.0 8:0.0 9:0.0 16:0.170268745957 17:0.072727270424
0 qid:20140102 2:0.0 3:0.0 4:0.0 5:0.225088833733223695 6:0.0 7:0.0 8:0.0 9:0.0 16:0.0327586196363 17:0.071428574621
0 qid:20140102 2:0.0 3:0.0 4:0.0 5:0.225088833733223695 6:0.0 7:0.0 8:0.0 9:0.0 16:0.1705834477 17:0.0277777779847 2
0 qid:20140102 2:0.0 3:0.0 4:0.0 5:0.225088833733223695 6:0.0 7:0.0 8:0.0 9:0.0 16:0.05330988988235 17:0.00261780107
0 qid:20140102 2:0.0 3:0.0 4:0.0 5:1.4683856568089043E-5 6:0.0 7:0.0 8:0.0 9:0.0 16:0.0261627901345 17:0.0819672122
1 qid:20140102 2:0.0 3:0.0 4:0.0 5:1.4683856568089043E-5 6:0.0 7:0.0 8:0.0 9:0.0 16:0.0659081215273 17:0.0943396240
0 qid:20140102 2:0.0 3:0.0 4:0.0 5:1.4683856568089043E-5 6:0.0 7:0.0 8:0.0 9:0.0 16:0.0969899697323 17:0.0208333339
1 qid:20140102 2:0.0 3:0.0 4:0.0 5:1.4683856568089043E-5 6:0.0 7:0.0 8:0.0 9:0.0 16:1.0 17:0.166666671634 22:0.0
0 qid:20140102 2:0.0 3:0.0 4:0.0 5:1.4683856568089043E-5 6:0.0 7:0.0 8:0.0 9:0.0 16:0.10218905694 17:0.018181817606
0 qid:20140102 2:0.0 3:0.0 4:0.0 5:1.4683856568089043E-5 6:0.0 7:0.0 8:0.0 9:0.0 16:0.127912826976 17:0.02631578966
0 qid:20140102 2:0.0 3:0.0 4:0.0 5:1.4683856568089043E-5 6:0.0 7:0.0 8:0.0 9:0.0 16:0.086005521384 17:0.05882352963
```

References and Further Readings

Books

Liu, Tie-Yan. *Learning to rank for information retrieval*. Vol. 13. Springer, 2011.

Li, Hang. "Learning to rank for information retrieval and natural language processing." *Synthesis Lectures on Human Language Technologies* 4.1 (2011): 1-113.

Helpful pages

http://en.wikipedia.org/wiki/Learning_to_rank

Packages

RankingSVM: http://svmlight.joachims.org/

RankLib: http://people.cs.umass.edu/~vdang/ranklib.html

Data sets

LETOR http://research.microsoft.com/en-us/um/beijing/projects/letor//

Yahoo! Learning to rank challenge http://learningtorankchallenge.yahoo.com