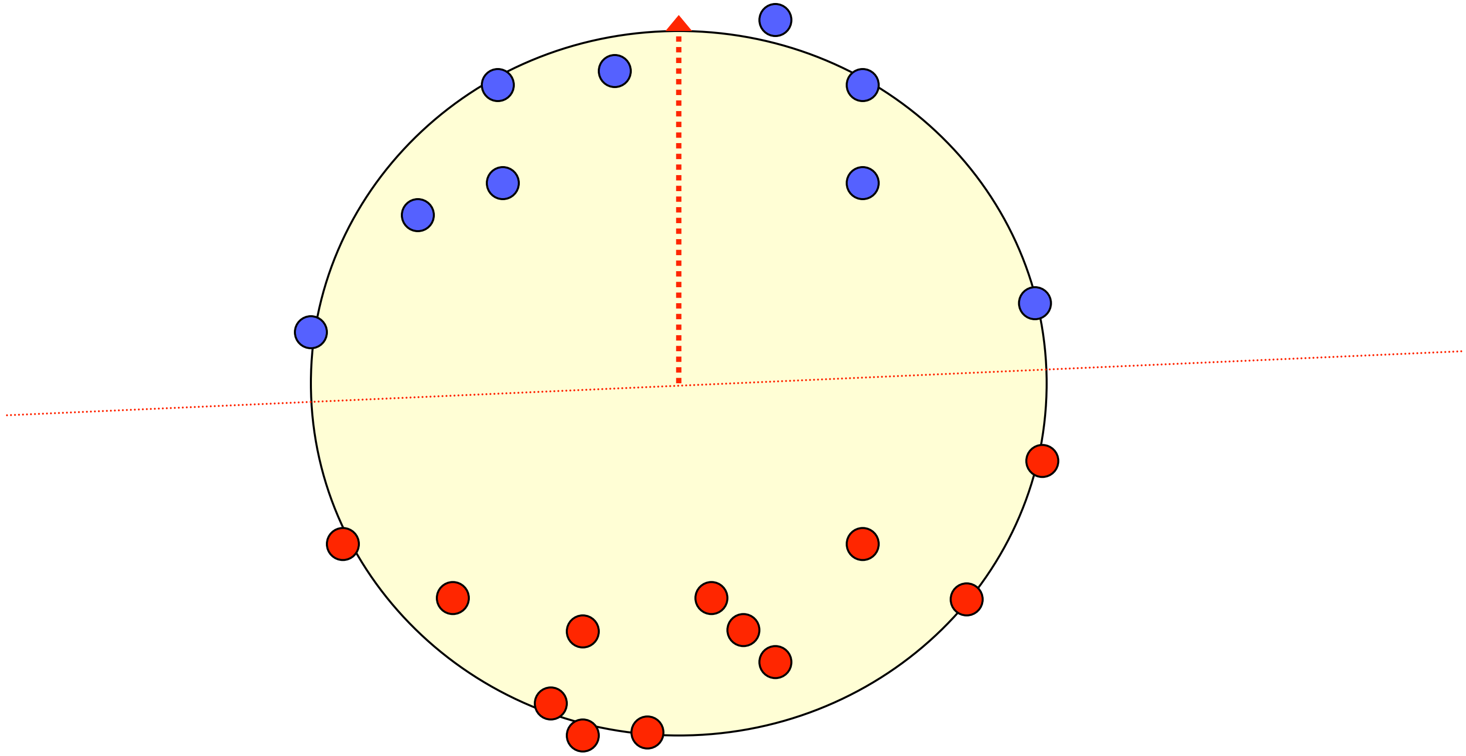
Sublinear Optimization

Elad Hazan @ Technion

Based on:

Clarkson, Hazan & Woodruff [FOCS '10,
Journal of the ACM '12]
+ Garber, Hazan, [NIPS '11]
Hazan, Koren, Srebro, [NIPS '11]

Linear Classification



Linear Classification

n vectors in d dimensions: $A_1,\ldots,A_n$ in R^d

Labels $y_1,\ldots,y_n$ in $\{-1,1\}$

Find vector x such that:

$$\forall i \in [n] \ . \ \mathrm{sign}(A_i^\top x) = y_i$$

Linear Classification

- Fundamental machine learning primitive
- Google(linear classification) ~ 126M [Bing~36M]
  Google(linear programming) ~ 50M [Bing~32M]
- Internet applications: spam detection, text categorization, image classification, …
- Reuters RCV1 dataset: 800K docs, 2M dimensions

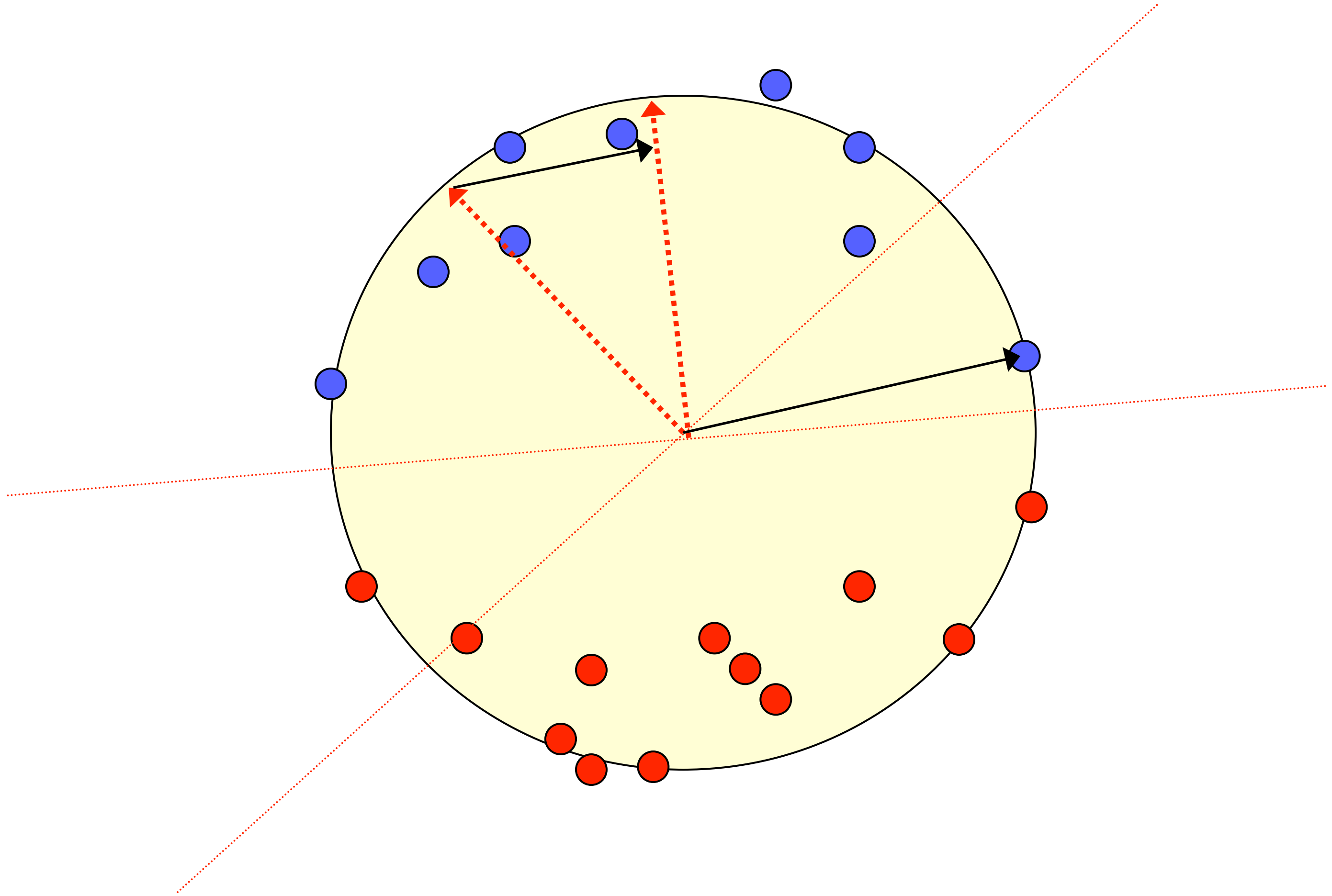
PROBLEMS ARE VERY LARGE !

["very-huge" on Nesterov's scale]

Nesterov: "think twice before adding two vectors"

The Perceptron Algorithm

- [Rosenblatt 1957, Novikoff 1962, Minsky&Papert 1969]

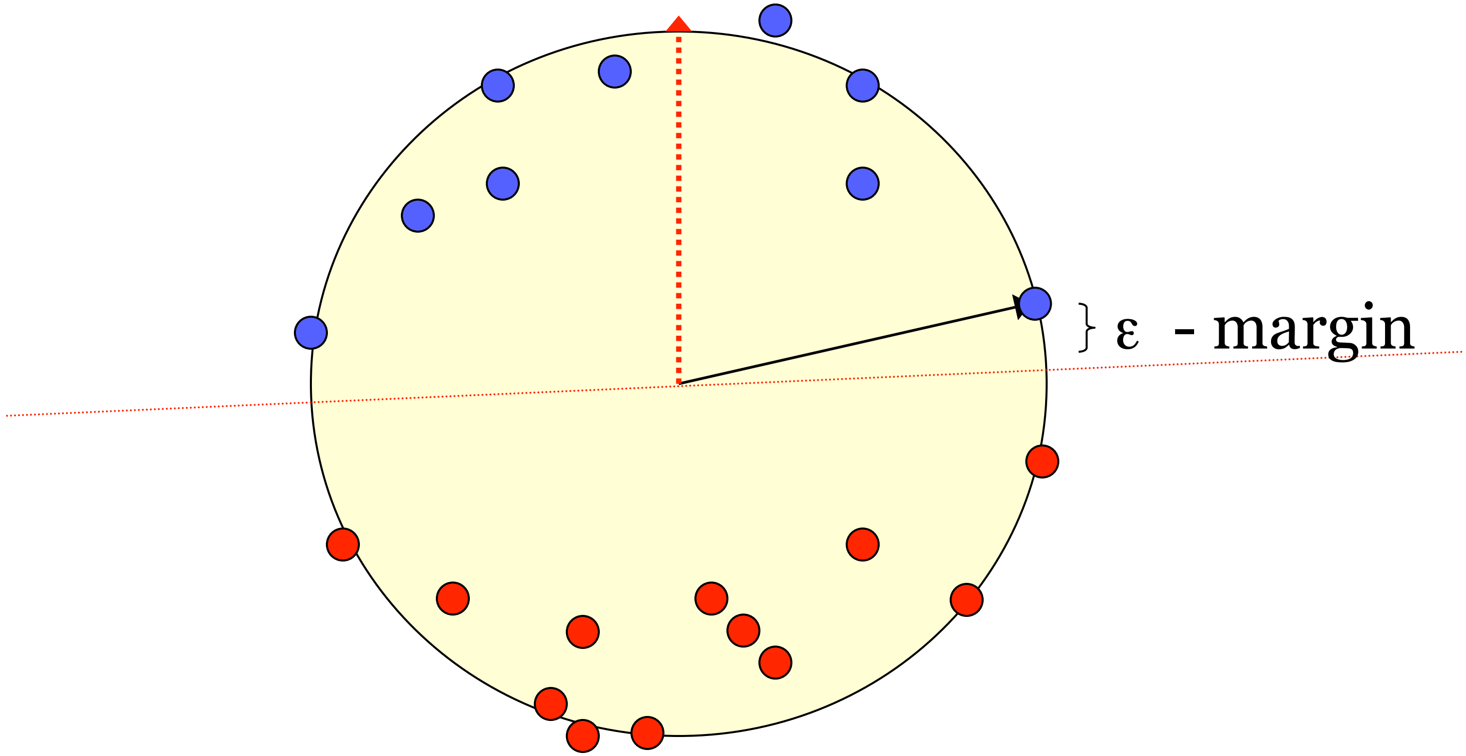


The Perceptron Algorithm

Iteratively:

1. Find vector A_i for which $\mathrm{sign}(A_i\, x) \neq y_i$
2. Add A_i to x:

$$x_{t+1} \leftarrow x_t + y_i A_i$$


} ε - margin

The Perceptron Algorithm

Thm [Novikoff 1962]: returns ε-approximate solution in

$1/\varepsilon^2$ iterations

(ε-far from optimal margin)

For n vectors in d dimensions:

$1/\varepsilon^2$ iterations

Each – $n \times d$ time total time: $O(\frac{nd}{\varepsilon^2})$

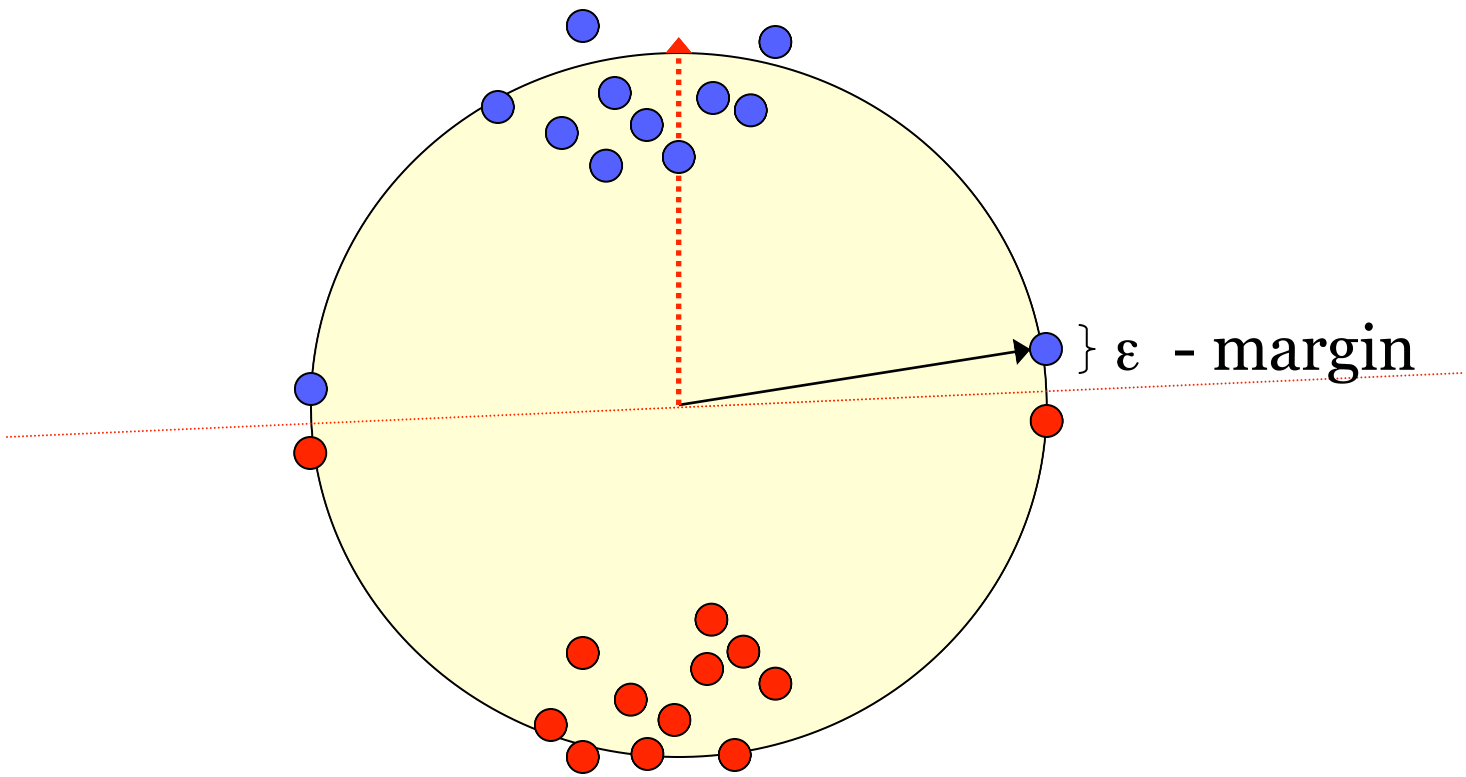
Our new algorithm: $O(\frac{n+d}{\varepsilon^2})$ $\Omega(\frac{n+d}{\epsilon^2})$

Sublinear (expected) time, leading order term improvement

(independently, Juditsky & Nemirovski $O(nd \log \frac{1}{\epsilon} + \frac{n+d}{\epsilon^2})$

(in running times, poly-log factors are omitted)

Why is it surprising ?



More results

- $O(\frac{n}{\varepsilon^2} + \frac{d}{\varepsilon})$ time alg for margin estimation / MEB

- Sublinear time kernel versions, i.e. polynomial kernel of deg q:

$$\tilde{O}(\frac{q(n+d)}{\varepsilon^2} + poly(\frac{1}{\varepsilon}))$$

- Poly-log space / low pass algorithms for these problems

All running times are tight up to polylog factors (information theoretic lower bounds)

More results

- Sublinear alg for semi-definite programming (w. Garber)

$$\tilde{O}\left(\frac{m}{\varepsilon^2} + \frac{n^2}{\varepsilon^{2.5}}\right) \qquad \Omega\left(\frac{m}{\varepsilon^2} + \frac{n^2}{\varepsilon^2}\right)$$

- Faster alg for soft-margin-SVM (w. Koren, Srebro)
- Generic sublinear-opt template for convex programming.

Talk outline

- describe new algorithm
- Analysis sketch
- Kernels
- Semidefinite programming / metric learning
- Errors (soft margin SVM)
- Lower bounds

A Primal-dual Perceptron

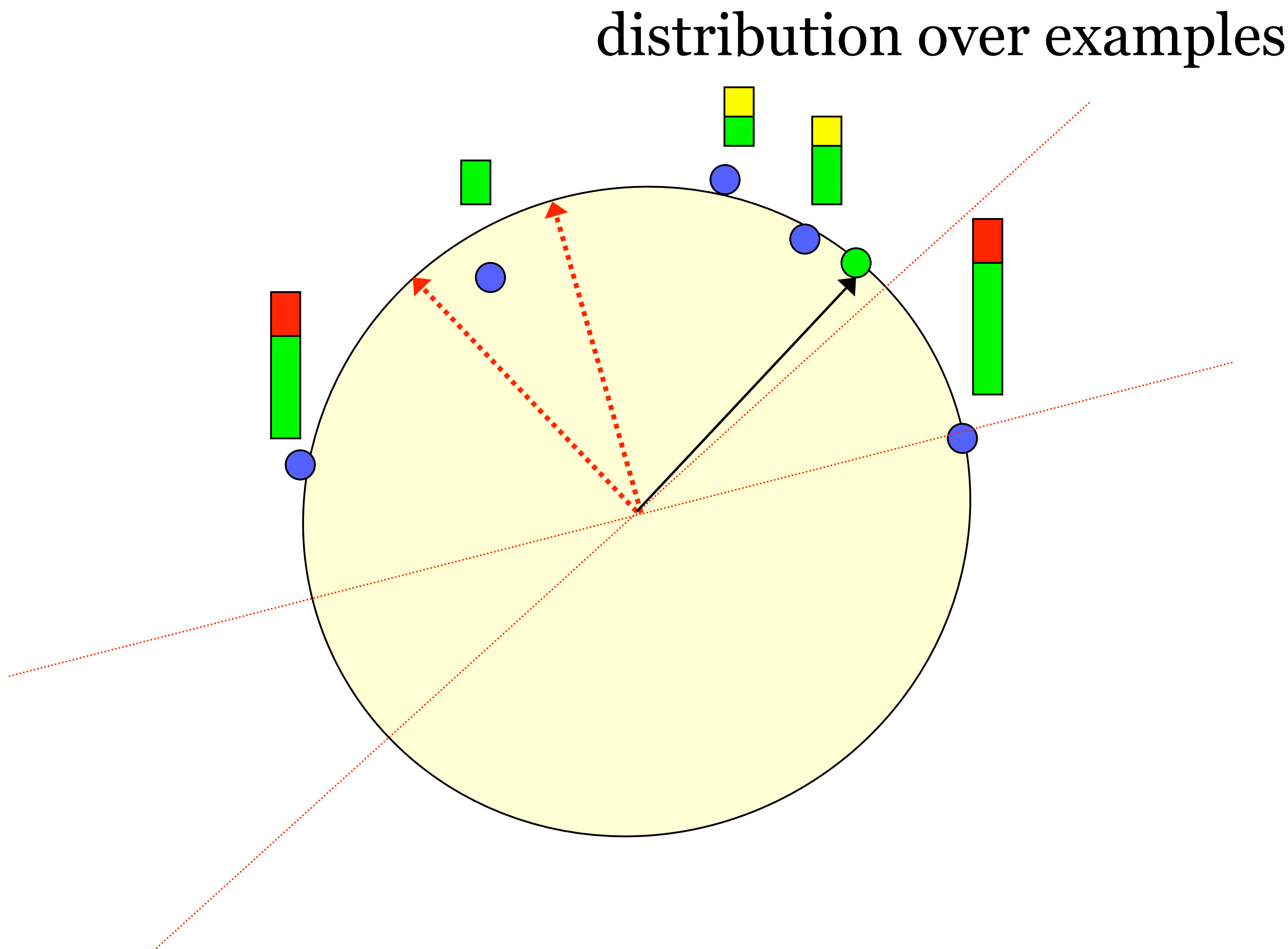
Iteratively:

1. Primal player supplies hyperplane x_t
2. Dual player supplies distribution p_t
3. Updates:

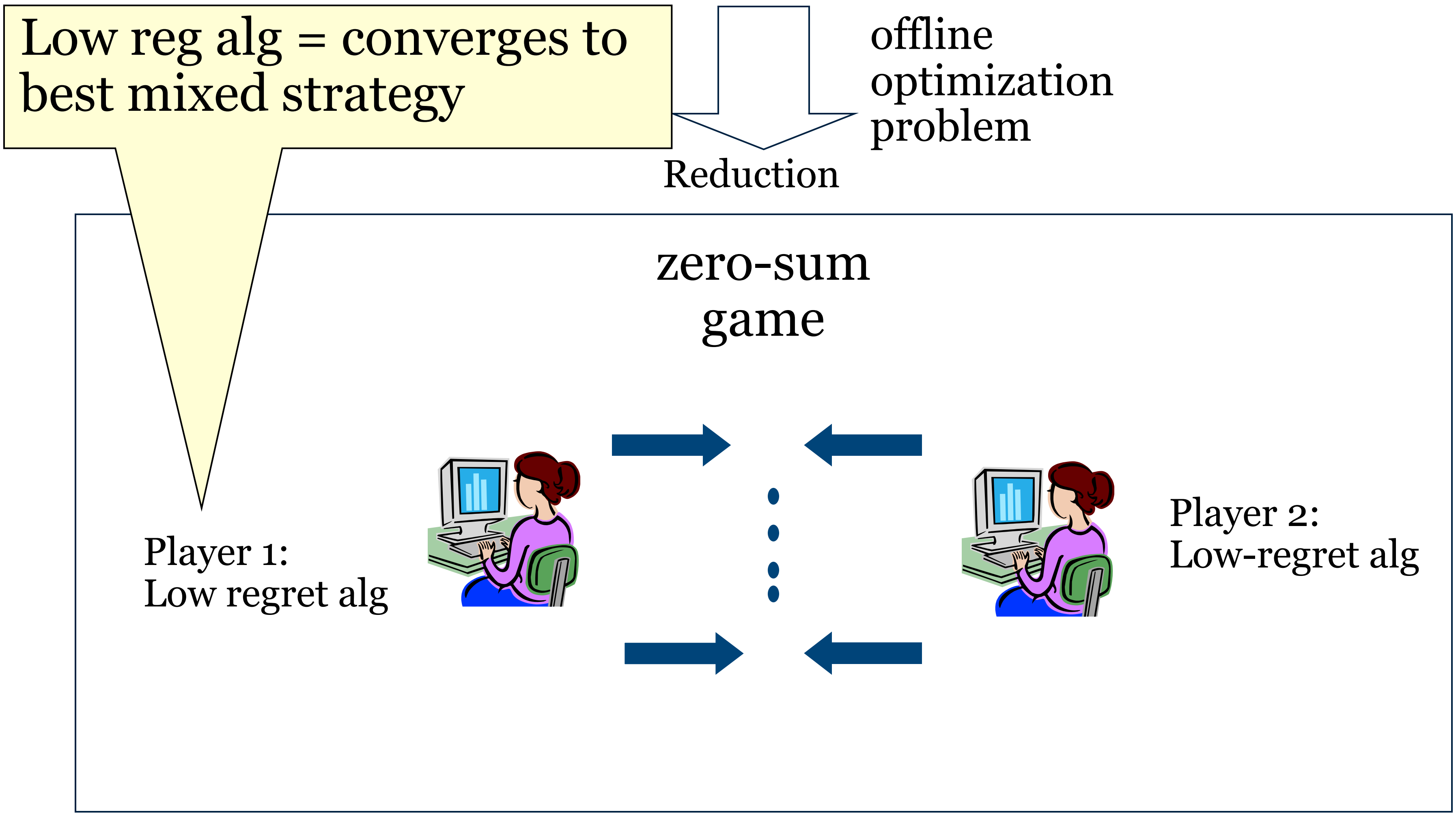
$$x_{t+1} \leftarrow x_t + \eta \sum_{i=1}^{n} p_t(i) A_i$$

$$p_{t+1}(i) \leftarrow p_t(i) \times e^{-\eta A_i x_t}$$

The Primal-dual Perceptron



Optimization via game solving



Converges to the min-max solution

Thm : time to converge to ε-approximate solution is bounded by T for which:

$$\frac{\text{Regret}_1 + \text{Regret}_2}{T} \leq \varepsilon$$

Total time = # iterations × time-per-iteration

Advantages:

- Generic optimization
- Learning algorithms (regret) are robust
- Easy to apply randomization

A Primal-dual Perceptron

Iteratively:

1. Primal player supplies hyperplane x_t
2. Dual player supplies distribution p_t
3. Updates:

$$x_{t+1} \leftarrow x_t + \eta \sum_{i=1}^{n} p_t(i) A_i$$

$$p_{t+1}(i) \leftarrow p_t(i) \times e^{-\eta A_i x_t}$$

iterations via regret of OGD/MW: $\text{Regret}_1 \leq 2\sqrt{T}$, $\text{Regret}_2 \leq \sqrt{T \log n}$

$$\frac{\text{Regret}_1 + \text{Regret}_2}{T} \leq \varepsilon \qquad \# \text{ iters} \leq \frac{\log n}{\varepsilon^2}$$

A Primal-dual Perceptron

Total time ? $$\# \text{ iters} \times \text{updates} = \frac{nd \log n}{\varepsilon^2}$$

Speed up via randomization:

1. Sufficient to look at one example
2. Sufficient to obtain crude estimates of inner products (main difficulty, new variance-MW lemma, Nemirovski: "you go inside the prox")

l_2 sampling

Consider two vectors from the d-dim sphere u,v

- Sample coordinate i w.p. v_i^2
- Return $X = \frac{u_i}{v_i}$

Notice that

- Expectation is correct $E[X] = \sum_i v_i^2 \times \frac{u_i}{v_i} = v^\top u$
- Variance at most one (magnitude can be d)

$$E[X^2] = \sum_i v_i^2 \times (\frac{u_i}{v_i})^2 = \sum_i u_i^2 = 1$$

- Time: O(d)

The Sublinear Perceptron

Iteratively:

1. Primal player supplies hyperplane x_t, l_2 sample from x_t
2. Dual player supplies distribution p_t, sample from it j_t
3. Updates:

$$x_{t+1} \leftarrow x_t + A_{j_t}$$

$$p_{t+1}(i) \leftarrow p_t(i) \times e^{-\eta \ell_2\text{-sample}(A_i x_t)}$$

$$p_{t+1}(i) \leftarrow p_t(i) \times (1 - \eta l_2\text{-sample}(A_i x_t) + \eta^2 l_2\text{-sample}(A_i x_t)^2)$$

Important: preprocess x_t only once for all estimates

Running time: $O(\frac{n+d}{\varepsilon^2})$

Analysis

Difficulties:

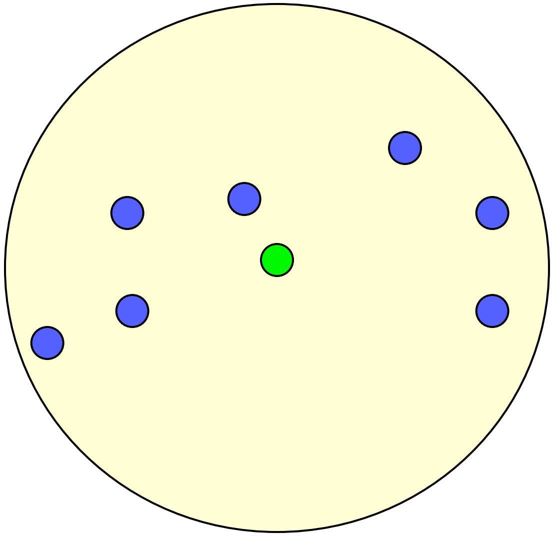
1. MW regret proportional to magnitude (which can be d)
2. Overall magnitude of updates is not bounded w.h.p (only with constant prob.)
3. Verifying a solution (naively) takes $O(nd)$ time

-> New multiplicative update for bounded variance

-> Result holds w.p. ½

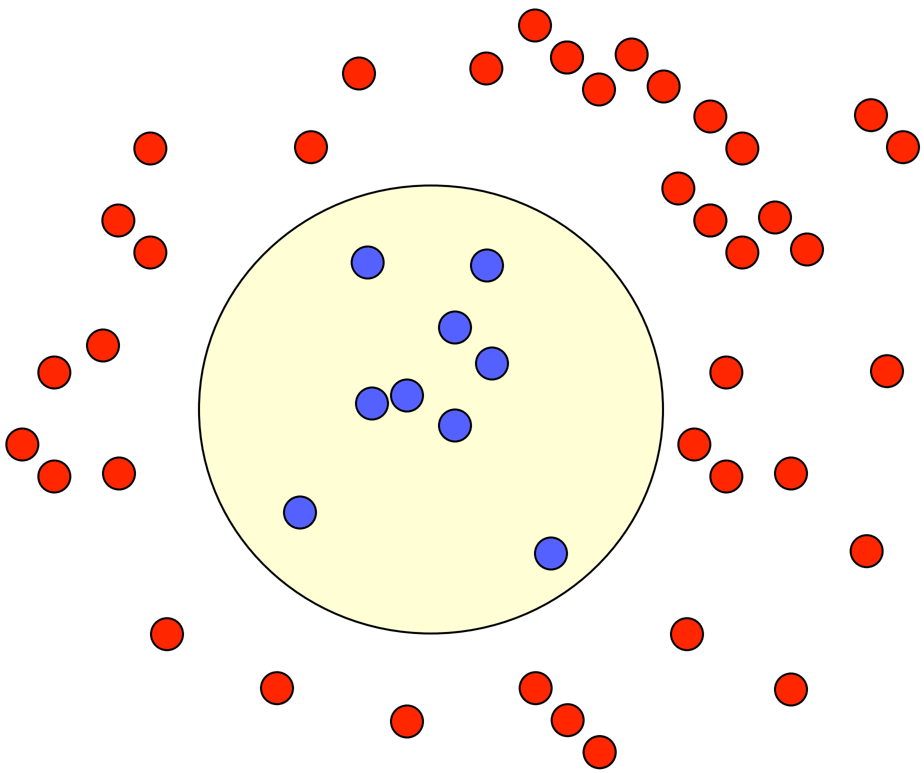
-> Exponential tail bounds possible

MEB (minimum enclosing ball)



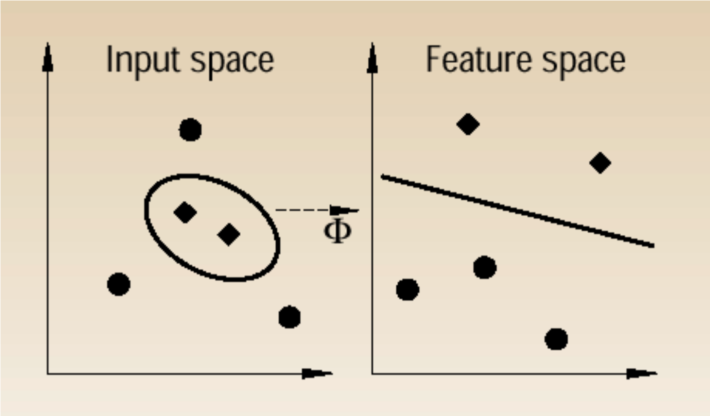
$$\min_{x \in R^d} \max_{i \in [n]} \|x - A_i\|^2$$

Kernels



Kernels

Map input to higher dimensional space via non-linear mapping. i.e. polynomial:



$$\Phi(x_1, x_2) = (x_1, x_2, \sqrt{2}\, x_1 \cdot x_2)$$

Classification via linear classifier in new space.

Efficient classification and optimization if inner products can be computer efficiently (the "kernel function")

$$K(x, y) = \Phi(x) \times \Phi(y)$$

$$\text{polynomial:}\;\; K(x, y) = (x^\top y)^q$$

The Sublinear Perceptron

Iteratively:

1. Primal player supplies hyperplane x_t,
   l_2 sample from x_t
2. Dual player supplies distribution p_t, sample from it j_t
3. Updates:

$$x_{t+1} \leftarrow x_t + \eta A_{j_t}$$

$$p_{t+1}(i) \leftarrow p_t(i) \times (1 - \eta l_2\text{-sample}(A_i x_t) + \eta^2 l_2\text{-sample}(A_i x_t)^2)$$

The Sublinear Kernel Perceptron

Iteratively:

1. Primal player supplies hyperplane x_t,
   l_2 sample from x_t
2. Dual player supplies distribution p_t, sample from it j_t
3. Updates:

$$x_{t+1} \leftarrow x_t + \eta \Phi(A_{j_t})$$

$$p_{t+1}(i) \leftarrow p_t(i) \times (1 - \eta l_2\text{-sample}(\Phi(A_i)\Phi(x_t)) + \eta^2 l_2\text{-sample}(\Phi(A_i)\Phi(x_t))^2)$$

l_2 sampling for kernels

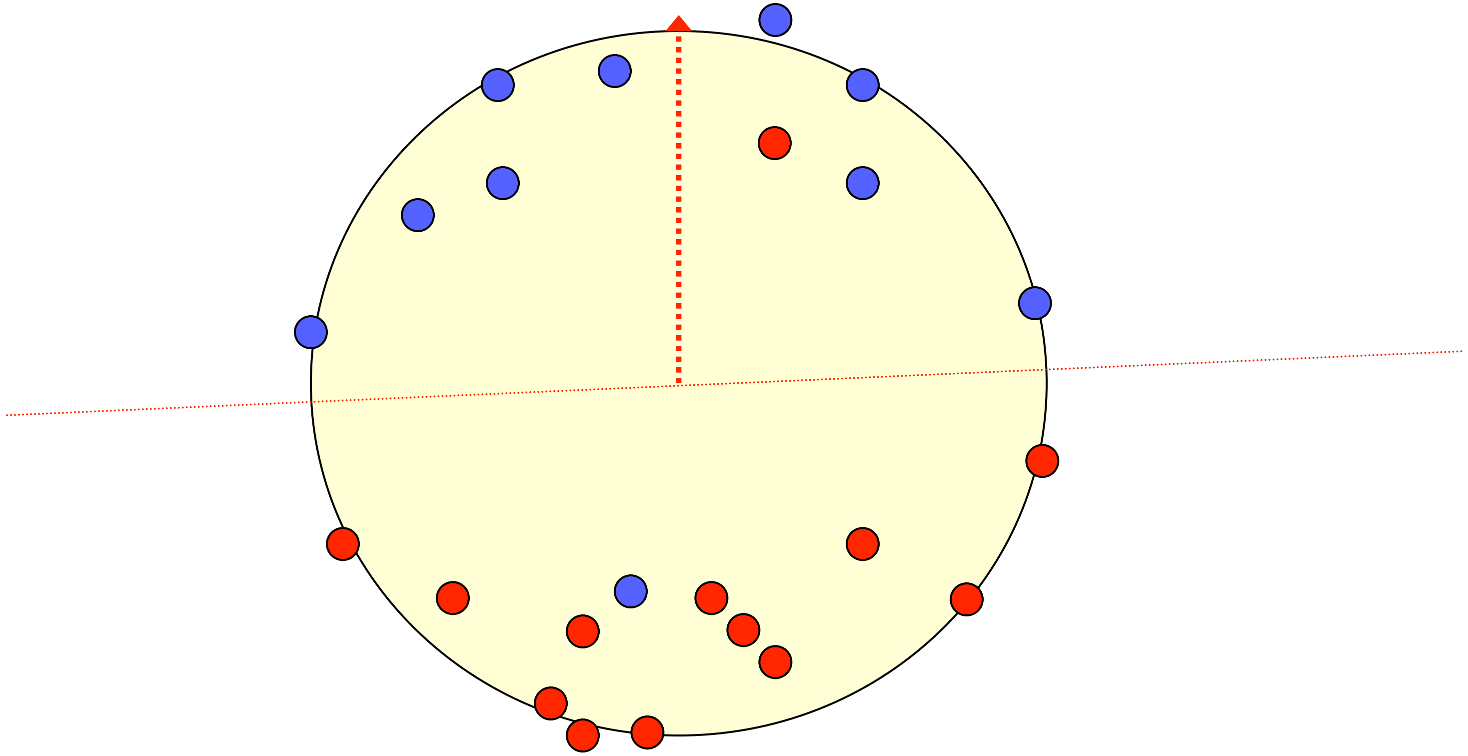
Polynomial kernel: $K(x, y) = (x^T y)^q$

Kernel l_2 sample = q independent l_2 samples of $x^T y$

Running time decreases by q

Efficient sampling for Gaussian, Exponential kernels

Soft-margin SVM [Cortez-Vapnik'95]



Soft-margin SVM

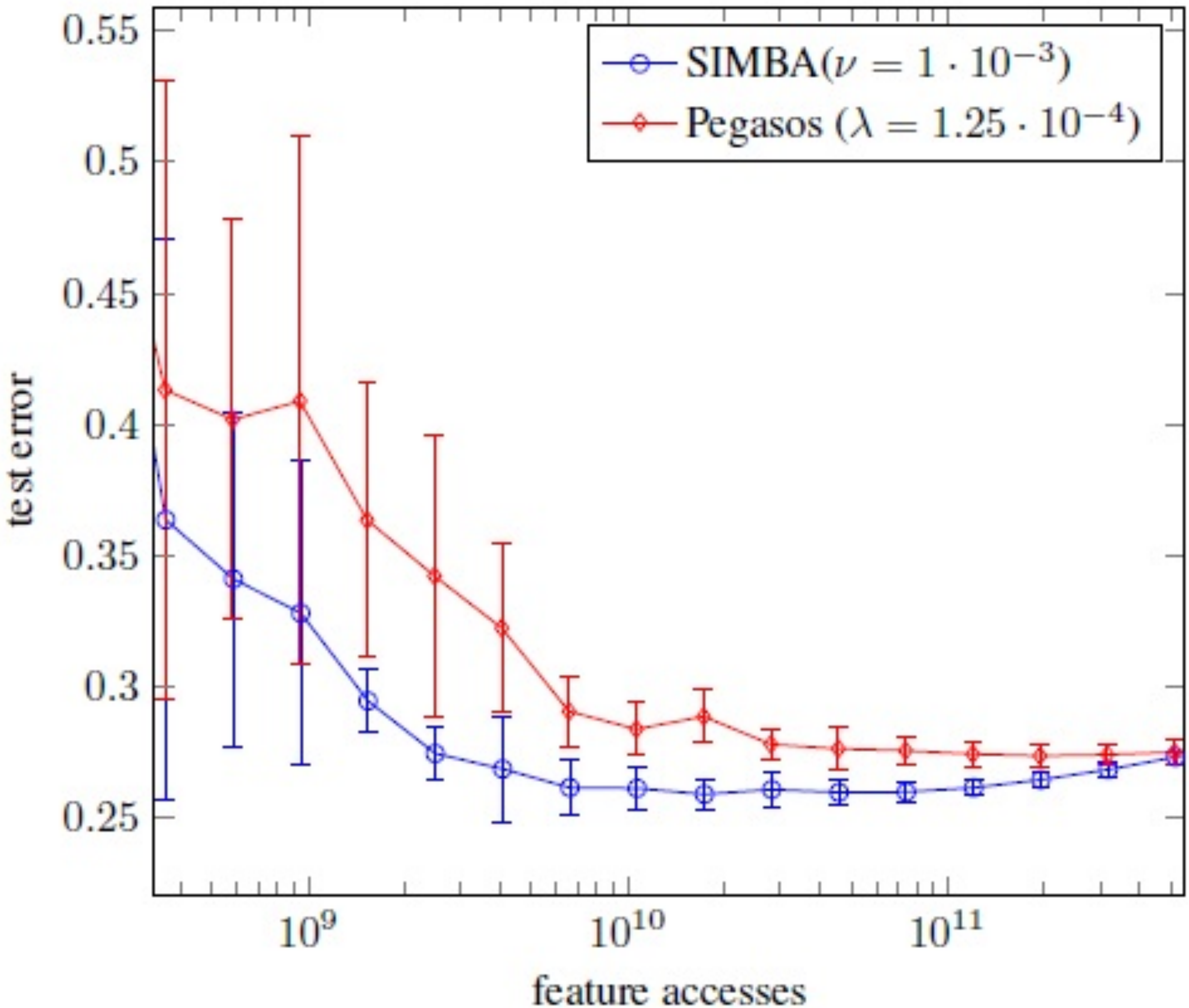
Minimize soft margin formulation:

$$\min_{w \in \mathcal{R}^d} \left\{ \frac{1}{n} \sum_{i=1}^{n} \max\{0, 1 - y_i \cdot w^\top x_i\} + \frac{\lambda}{2} \|w\|^2 \right\}$$

Via stochastic gradient descent – easy to get ε-approximation in time d/ ε²

This is tight in the “example model” !!

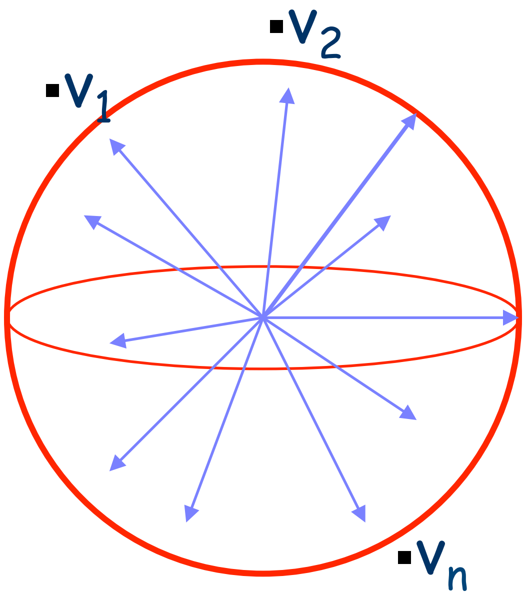
In RA model, get faster running time to get δ generalization error.



SIMBA($\nu = 1 \cdot 10^{-3}$)
Pegasos ($\lambda = 1.25 \cdot 10^{-4}$)
test error
feature accesses
0.55
0.5
0.45
0.4
0.35
0.3
0.25
10^9
10^{10}
10^{11}

Semidefinite Programming

$$A_1 \bullet X \geq b_1$$
$$A_2 \bullet X \geq b_2$$
$$\dots$$
$$A_m \bullet X \geq b_m$$
$$X \in \mathbb{R}^{n \times n}$$
$$X \succeq 0$$
$$\mathrm{Tr}(X) \leq 1$$



- $X_{ij} = v_i \times v_j$

Semidefinite Programming

- Machine Learning: learning pseudo-metrics , HUGE instances
- Interior point methods [Nesterov & Nemirovski, Alizadeh] :

$$\tilde{O}(n^3\sqrt{m}\log\frac{1}{\epsilon})$$

- Approximation algorithms: [Klein-Lu, AHK, Iyengar-Phillips-Stein]:

$$\tilde{O}(\frac{mn^2}{poly(\epsilon)})$$

- Our new alg: $\tilde{O}(\frac{m}{\varepsilon^2}+\frac{n^2}{\varepsilon^{2.5}})$

- Technology: Frank-Wolfe technology, Matrix Bernstein thm [Recht '09]

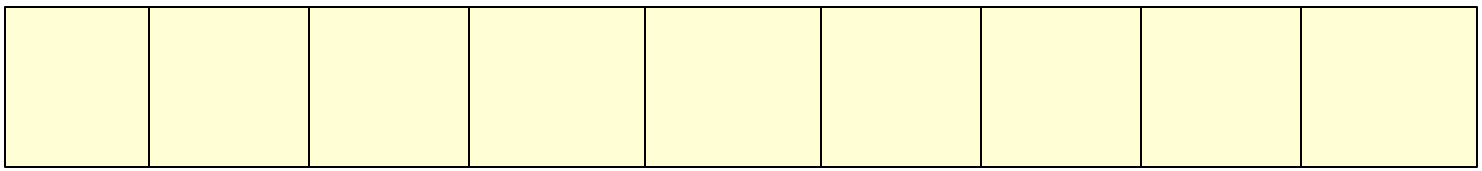
Semidefinite Programming

- Min-max formulation:

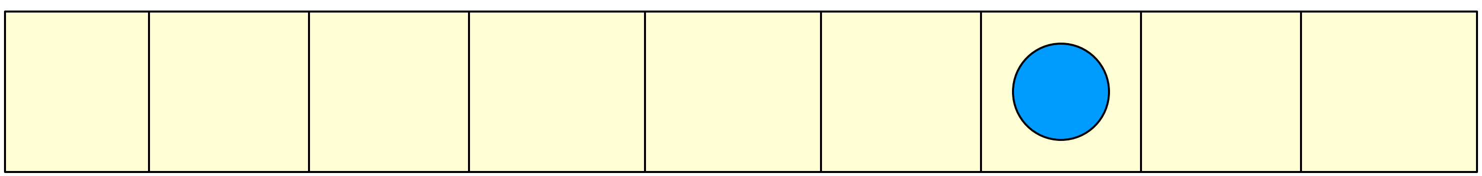
$$\max_{X} \min_{p,Z} \left\{ \sum_{i=1}^{m} p_i (A_i \bullet X - b_i) + Z \bullet X \right\}$$

- Gradient ascent primal step (no need to project onto SDP cone)
- Hybrid dual step: MW on p, optimization on Z (eigenvector)

Lower bounds

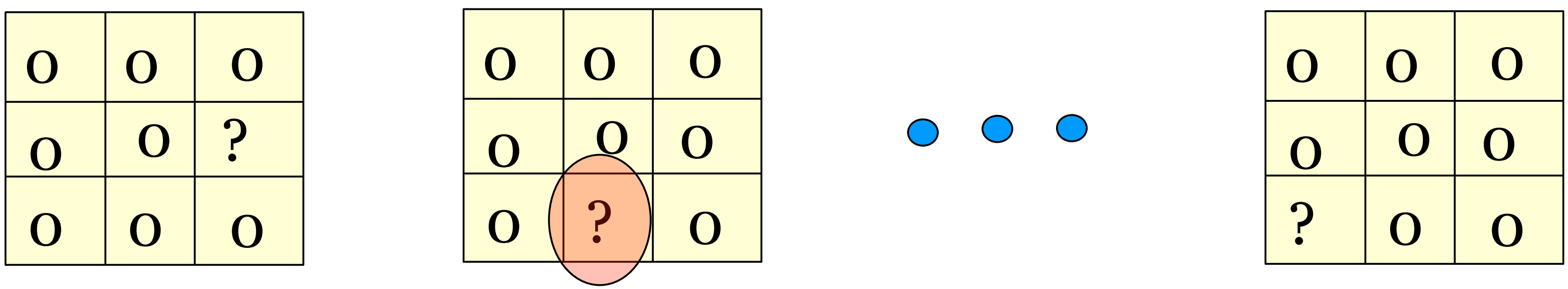


OR ??



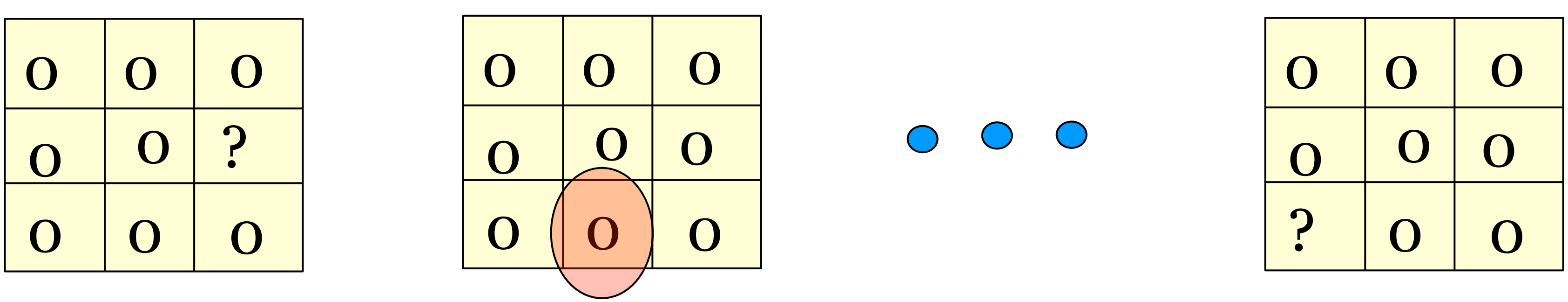
To decide w.p. >= 2/3 , need to see >= ½ bins

“good SDP”



$$\exists X \in P \,.\, A_i \bullet X \geq \epsilon$$

“bad SDP”



$$\exists i \in [m] \,.\, A_i = 0 \Rightarrow \forall X \in P \,.\, A_i \bullet X \leq 0$$

Summary / further directions / open questions

- First sublinear algs for optimization of classifiers, LP, faster SVM, SDP approximation.
- Optimize any convex opt. problem in "information-limit" time!
- Assumptions on data that permit faster optimization ?
- Exploit computer architecture (non-RAM)
- Resolve "early global minimum" in experiments
- What if one pass is allowed ? (not truly sublinear)

100 € question

- Solve linear classification in time:

$$\tilde{O}\left(nd \log \frac{1}{\epsilon} + \frac{poly(\log n, \log d)}{\epsilon^2}\right)$$

- Resolution = I pay ticket to para-gliding @ Chamonix

Thank you!