Online Learning and Bregman Divergences

Gunnar Rätsch[&] and Manfred Warmuth[$]

[&] *Australian National University, Canberra, Australia*

[$] *University of California at Santa Cruz, USA*

[&] `http://mlg.anu.edu.au/~raetsch`

[$] `http://www.cse.ucsc.edu/~manfred`

Help with the tutorial: Claudio Gentile, Jyrki Kivinen

Content of this tutorial

- L 1: Introduction to Online Learning
  - The Learning setting
  - Predicting as good as the best expert
  - Predicting as good as the best linear combination of experts
- L 2: Bregman divergences and Loss bounds
  - Introduction to Bregman divergences
  - Relative loss bounds for the linear case
  - Nonlinear case & matching losses
  - Duality and relation to exponential families
- L 3: Extensions, interpretations, applications
  - Online to Batch Conversions
  - Prior information on the weight vector
  - Some applications

Goal: How to learn efficiently in an online setting? How can one prove it?

Sources of Information

Internet http://mlg.anu.edu.au/~raetsch/online

Journals and papers Machine Learning, Journal of Machine Learning Research, Information and Computation, IEEE Transaction on Information Theory, ...
Small collection at http://mlg.anu.edu.au/~raetsch/online

Conferences Neural Information Processing Systems (NIPS), Computational Learning Theory (COLT)

People (incomplete) List at http://mlg.anu.edu.au/~raetsch/online

Software Not necessary, algorithms are too simple

The Online Learning setting

Protocol:

For $t = 1$ To T Do

Get instance	$\boldsymbol{x}_t \in \mathbf{R}^n$
Predict	$\hat{y}_t \in \mathcal{Y}$
Get label	$y_t \in \mathcal{Y}$
Incur loss	$L(y_t, \hat{y}_t, \boldsymbol{x}_t)$

Problem instances:

- Classification: $\hat{y}_t, y_t \in \{0, 1\}$, e.g. $L(y, \hat{y}, \boldsymbol{x}) = |y - \hat{y}|$
- Regression: $\hat{y}_t, y_t \in \mathbf{R}$, e.g. $L(y, \hat{y}, \boldsymbol{x}) = (y - \hat{y})^2$
- Density estimation: no label, e.g. $L(y, \hat{y}, \boldsymbol{x}) = -\log P(\boldsymbol{x}|\theta)$

Goal: small total loss $\sum_t L(y_t, \hat{y}_t, \boldsymbol{x}_t)$

Predicting almost as good as the best expert

	E_1	E_2	E_3	$\cdots$	E_n	prediction	*true label*	loss
day 1	1	1	0	$\cdots$	0	0	1	1
day 2	1	0	1	$\cdots$	0	1	0	1
day 3	0	1	1	$\cdots$	1	1	1	0
day t	$x_{t,1}$	$x_{t,2}$	$x_{t,3}$	$\cdots$	$x_{t,n}$	$\hat{y}_t$	y_t	$\lvert y_t - \hat{y}_t \rvert$

Master Algorithm

For $t = 1$ To T Do

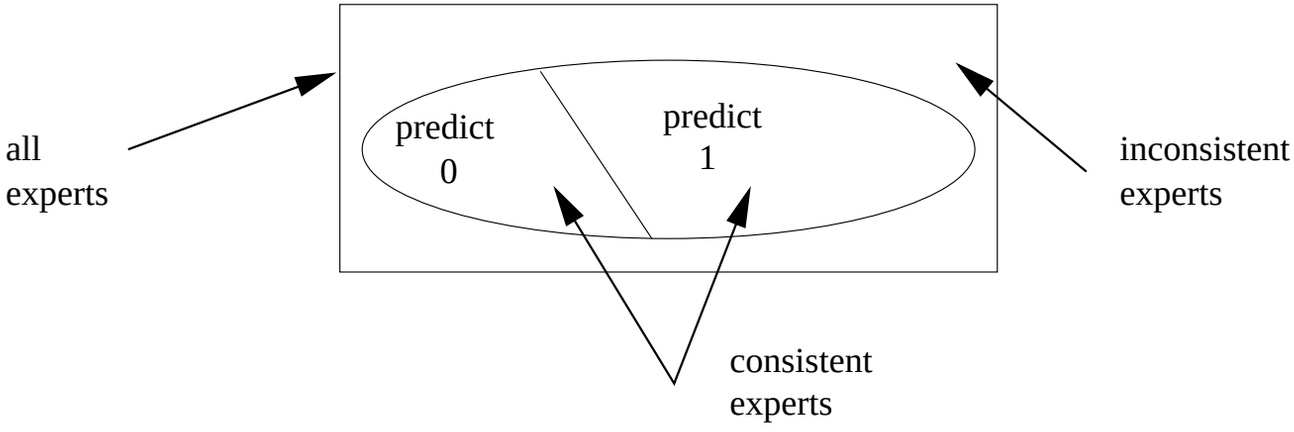
Get instance $\quad \boldsymbol{x}_t \in \{0,1\}^n$

Predict $\quad \hat{y}_t \in \{0,1\}$

Get label $\quad y_t \in \{0,1\}$

Incur loss $\quad |y_t - \hat{y}_t|$

Halving Algorithm [BF]



- Predicts with majority
- If mistake is made, then number of consistent experts is (at least) halved
- any mistake is "converted" into knowledge on the learning problem: *mistake driven learning*

A run of the Halving Algorithm

E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	*majority*	*true label*	loss
1	1	0	0	1	1	0	0	1	0	1
x	x	0	1	x	x	1	1	1	1	0
x	x	x	1	x	x	0	0	0	1	1
x	x	x	↑	x	x	x	x			
			consistent							

For any sequence with a consistent expert
Halving Algorithm makes at most $\leq \log_2 n$ mistakes

(Good bound, but not optimal)

What if no expert is consistent?

Sequence of examples $S = (\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_T, y_T)$

- total loss of algorithm A: $L_A(S) = \sum_{t=1}^{T} L(A(\boldsymbol{x}_t), y_t)$
- total loss of i-th expert E_i: $L_i(S) = \sum_{t=1}^{T} L(E_i(\boldsymbol{x}_t), y_t)$

Want bounds of the form:

$$\forall S: \quad L_A(S) \;\leq\; a \;\min_i L_i(S) + b \;\log(n)$$

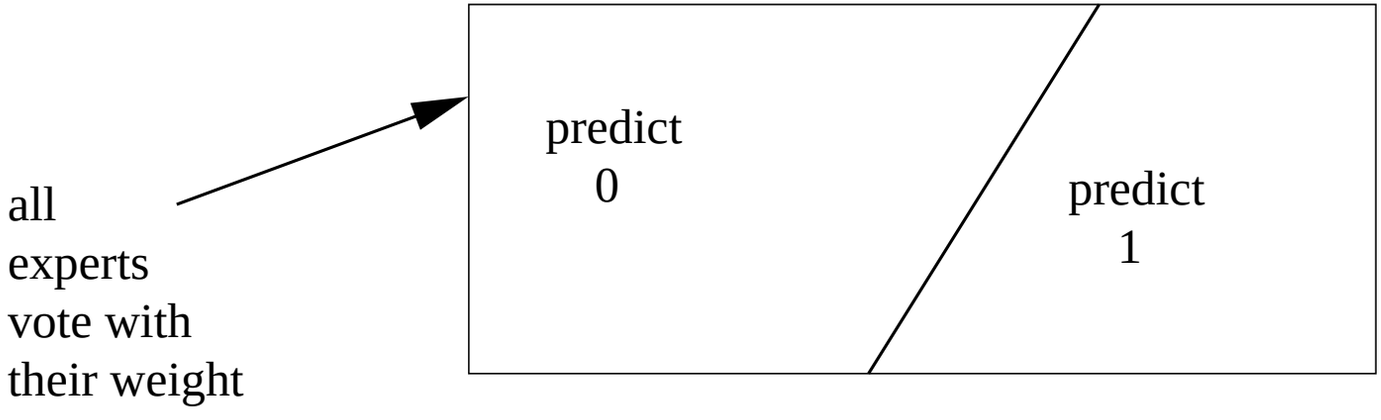
where a, b are constants

Bounds loss of algorithm relative to loss of best expert

Weighted Majority Algorithm [LW]

Can't wipe out experts!

Need one weight per expert



- Predicts with larger side
- Weights of wrong experts are multiplied by $\beta \in [0, 1)$

Number of mistakes of the WM algorithm

$$
\begin{aligned}
M_{t,i} &= \#\text{ of mistakes of } E_i \text{ before trial } t \\
w_{t,i} &= \beta^{M_{t,i}} \text{ weight of } E_i \text{ at beginning of trial } t \\
W_t &= \sum_{i=1}^{n} w_{t,i} \quad \text{total weight at trial } t
\end{aligned}
$$

Minority $\leq \frac{1}{2} W_t$, Majority $\geq \frac{1}{2} W_t$

If no mistake then minority multiplied by β:

$$W_{t+1} \leq 1 \, W_t$$

If mistake then majority multiplied by β:

$$W_{t+1} \leq \underbrace{1 \quad \tfrac{1}{2} W_t}_{\text{minority}} + \underbrace{\beta \quad \tfrac{1}{2} W_t}_{\text{majority}} = \frac{1+\beta}{2} \, W_t$$

M = Total number of mistakes of WM

M_i = Total number of mistakes of expert E_i

Hence

$$\underset{\substack{total\ \ final \\ weight}}{W_{T+1}} \quad \leq \quad \left(\frac{1+\beta}{2}\right)^M \quad W_1$$

$$W_{T+1} \quad = \quad \sum_{j=1}^{n} w_{T+1,j} \quad = \quad \sum_{j=1}^{n} \beta^{M_{T+1,j}} \quad \geq \quad \beta^{M_{T+1,i}}$$

where $i = \text{argmin}_j \, M_{T+1,j}$

We got:

$$\left(\frac{1+\beta}{2}\right)^M \quad \underbrace{W_1}_{n} \quad \geq \quad \beta^{M_i}$$

Relative Loss bound for Weighted Majority

Solving for M:

$$M \leq \frac{\ln \frac{1}{\beta}}{\ln \frac{2}{1+\beta}} M_i + \frac{1}{\ln \frac{2}{1+\beta}} \ln n$$

$$M \leq \underbrace{2.63}_{a} \min_i M_i + \underbrace{2.63}_{b} \ln n \qquad \beta = 1/e$$

For all sequences, loss of master algorithm
is comparable to loss of best expert
$\Rightarrow$ Relative loss bounds [Fr]

Other Loss Functions

absolute loss	$L(y,\hat{y})$	$=$	$\lvert y-\hat{y}\rvert$
square loss	$L(y,\hat{y})$	$=$	$(y-\hat{y})^2$
entropic loss	$L(y,\hat{y})$	$=$	$y\ln\frac{y}{\hat{y}}+(1-y)\ln\frac{1-y}{1-\hat{y}}$, $y,\hat{y}\in[0,1]$
entropic loss $\pm$	$L(y,\hat{y})$	$=$	$\frac{1+y}{2}\ln\frac{1+y}{1+\hat{y}}+\frac{1-y}{2}\ln\frac{1-y}{1-\hat{y}}$, $y,\hat{y}\in[-1,+1]$
hellinger loss	$L(y,\hat{y})$	$=$	$\frac{1}{2}\left(\sqrt{1-y}-\sqrt{1-\hat{y}}\right)^2+\frac{1}{2}\left(\sqrt{y}-\sqrt{\hat{y}}\right)^2$ $y,\hat{y}\in[0,1]$

How does it work with other loss functions?

One weight per expert: [V]

$$w_{t,i} = \beta^{\,L_{t,i}} = e^{-\eta\,L_{t,i}}$$

where $L_{t,i}$ is total loss of E_i before trial t
and η is a positive learning rate

Master predicts with the weighted average (WA) [KW]

$$v_{t,i} = \frac{w_{t,i}}{\sum_{i=1}^{n} w_{t,i}} \quad \text{normalized weights}$$

$$\hat{y}_t = \sum_{i=1}^{n} v_{t,i}\, x_{t,i} = \boldsymbol{v}_t \cdot \boldsymbol{x}_t$$

where $x_{t,i}$ is the prediction of E_i in trial t

Bounds for other Loss Functions

$\forall$ sequences S of examples $\langle(\boldsymbol{x}_t, y_t)\rangle_{1\le t\le T}$
where $\boldsymbol{x}_t \in [0,1]^n$ and $y_t \in [0,1]$

$$L_{WA}(S) \le \min_i \underbrace{1}_{a} L_i(S) + \underbrace{1/\eta}_{b} \ln(n)$$

$1/\eta$	WA	fancy
entropic	1	1
square	2	1/2
hellinger	1	.71

- Improved constants of $1/\eta$ when Master uses fancier pred. [V]
- For the discrete loss and the absolute loss: $a > 1$

Proof

Key inequality: $L(y, \boldsymbol{v}_t \cdot \boldsymbol{x}_t) \leq \frac{1}{\eta} \ln W_t - \frac{1}{\eta} \ln W_{t+1}$

$= -\frac{1}{\eta} \ln \frac{W_{t+1}}{W_t}$

Telescoping:

$$
\begin{aligned}
L_{\mathrm{WA}}(S) &\leq -\frac{1}{\eta} \ln \frac{W_{T+1}}{W_1} \\
&= -\frac{1}{\eta} \ln \sum_{j=1}^{n} \frac{1}{n} e^{-\eta L_{T+1,j}(S)} \\
&\leq -\frac{1}{\eta} \ln \frac{1}{n} e^{-\eta L_i(S)} \qquad\qquad i = \operatorname*{argmin}_j L_{T+1,j} \\
&= -\frac{1}{\eta} \ln \frac{1}{n} e^{-\eta L_{T+1,i}(S)} \\
&= L_{T+1,i}(S) + \frac{1}{\eta} \ln n
\end{aligned}
$$

Usefulness:

- Easy to combine many pretty good experts (algorithms) so that Master is guaranteed to be almost as good as the best
- Bounds logarithmic in number of experts (multiplicative updates)

Questions:

- How to obtain algorithms that do well compared to
  best linear combination or
  best thresholded linear combination of experts?
- How to motivate the updates?
- What are good measures of progress?
- What are good loss functions?
- Methods for proving relative loss bounds?

A more general setting

Example	**Prediction of alg A**	**Label**	**Loss of alg A**
$\boldsymbol{x}_1$	$\hat{y}_1$	y_1	$L(y_1, \hat{y}_1)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\boldsymbol{x}_t$	$\hat{y}_t$	y_t	$L(y_t, \hat{y}_t)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\boldsymbol{x}_T$	$\hat{y}_T$	y_T	$L(y_T, \hat{y}_T)$
		Total Loss	$L_A(S)$

Sequence of examples $S = (\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_T, y_T)$

Comparison class $\{\boldsymbol{u}\}$

Relative loss $\quad L_A(S) - \inf_{\{\boldsymbol{u}\}} L_{\boldsymbol{u}}(S)$

Goal: Bound relative loss for arbitrary sequence of examples

Example: Learning Disjunctions of Experts

variables/experts

E_1	E_2	E_3	E_4	*true* *label*	$E_1 \vee E_3$	$E_3 \vee E_4$
1	1	0	0	0	1	0
1	0	1	0	1	1	1
0	1	1	1	0	1	1
0	1	0	0	1	0	0
$x_{t,1}$	$x_{t,2}$	$x_{t,3}$	$x_{t,4}$		↑	↑
					3	2
					mistakes	

$E_1 \vee E_3$ becomes $\boldsymbol{u} = (1, 0, 1, 0)$

$E_1 \vee E_3$ is one on $\boldsymbol{x}_t \in \{0, 1\}^n$ iff $\boldsymbol{u} \cdot \boldsymbol{x}_t \geq 1$

Weighted Majority on k-literal Disjunctions

Do as well as best k out of n literal (monotone) disjunction

Each disjunction is an expert

Keep one weight per disjunction: $\binom{n}{k}$ weights

$$\begin{array}{c}\#\text{ of mistakes} \\ \text{of WM}\end{array} \quad \leq \quad 2.63\, M \;+\; 2.63\, k \ln \frac{n}{k}$$

M is # of mistakes of best

Time (and space) **exponential** in k

Efficient algorithm have only one weight per literal

The Perceptron Algorithm

In trial t: Get instance $\boldsymbol{x}_t \in \{0,1\}^n$

If $\boldsymbol{w}_t \cdot \boldsymbol{x}_t \geq 1/2$ then $\hat{y}_t = 1$

else $\hat{y}_t = 0$

Get label $y_t \in \{0,1\}$

If mistake then

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta\,(\hat{y}_t - y_t)\,\boldsymbol{x}_t$$

k-literal Disjunctions with Perceptron

Perceptron Convergence Theorem ($\eta = \frac{1}{2n}$)

$$\#\text{ of mistakes} \leq 4\,A\ +\ 4\,k\,n$$

where A is # of attribute errors of best disjunction of size k, i.e., the minimum # of attributes that need to be flipped to make the disjunction consistent

$$A \leq kM$$

Lower bound for rotation invariant algorithms: [KWA]

$$\#\text{mistakes} = \Omega(n)$$

The Winnow Algorithm [L]

In trial t: Get instance $\boldsymbol{x}_t \in \{0,1\}^n$

If $\boldsymbol{w}_t \cdot \boldsymbol{x}_t \geq \theta$ then $\hat{y}_t = 1$

else $\hat{y}_t = 0$

Get label $y_t \in \{0,1\}$

If mistake then

$$w_{t+1,i} = w_{t,i}\, e^{-\eta\,(\hat{y}_t - y_t)\, x_{t,i}}$$

Mistake bound ($e^{-\eta} = 1/3$, $\theta = \frac{3 \ln 3}{8}$) [AW]

$$\#\text{ of mistakes} \leq 4\, A \;+\; 3.6\, k \ln \frac{n}{k}$$

Not rotation invariant!

k-term DNF via Feature Expansion [KW]

$$\boldsymbol{x} = \begin{matrix} x_1 \\ \vdots \\ x_n \end{matrix} \quad \Rightarrow \quad \Phi(\boldsymbol{x}) = \begin{matrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_1 x_2 \\ \vdots \\ x_1\, x_2 \ldots x_n \end{matrix}$$

n inputs $\qquad\qquad$ 2^n features

k-term DNF in input space is k-literal disjunction in feature space

$$\underbrace{\Phi(\boldsymbol{x})}_{2^n} \cdot \underbrace{\Phi(\boldsymbol{y})}_{2^n} = \underbrace{\prod_{i=1}^{n}(1 + x_i\, y_i)}_{O(n) \text{ time}} = K(\underbrace{\boldsymbol{x}}_{n}, \underbrace{\boldsymbol{y}}_{n})$$

(Simple ANOVA kernel)

k-term DNF via Feature Expansion: Perceptron

Perceptron: $\boldsymbol{w}_t = \sum_{q \text{ mistake}} \alpha_q \Phi(\boldsymbol{x}_i)$

Prediction:

$$\boldsymbol{w}_t \cdot \Phi(\boldsymbol{x}) = \left(\sum_{q \text{ mistake}} \alpha_q \Phi(\boldsymbol{x}_q) \right) \cdot \Phi(\boldsymbol{x})$$

$$= \sum_{q \text{ mistake}} \alpha_q \Phi(\boldsymbol{x}_q) \cdot \Phi(\boldsymbol{x})$$

$$= \underbrace{\sum_{q \text{ mistake}} \alpha_q K(\boldsymbol{x}_q, \boldsymbol{x})}_{\text{time: } O(n \cdot \#\text{ mistakes})}$$

Mistake bound: $O(k\, 2^n)$

k-term DNF via Feature Expansion: Winnow

Winnow: $w_{t,i} = \exp\Big(-\eta \sum_{q \text{ mistake}} \alpha_q \, \Phi(\boldsymbol{x}_q)_i\Big)$

log of weights is linear combination of past examples

Mistake bound: $O(k \ln 2^n) = O(k\, n)$

prediction time: $\Omega(2^n \,\#$ mistakes$)$

No kernel trick with purely multiplicative updates!

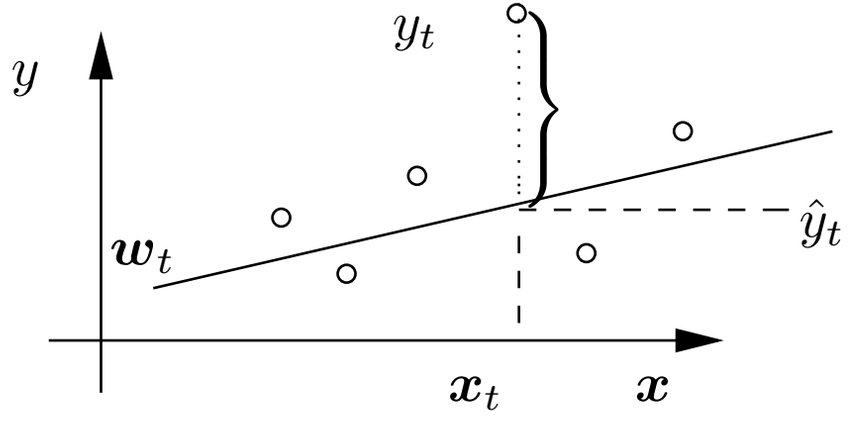
So far

- Learning relative to best expert
  and best disjunction
- Various loss functions
- Perceptron versus Winnow and expansion into feature space

On-line Linear Regression

For $t = 1, \ldots, T$ do

Get instance	$\boldsymbol{x}_t \in \mathbf{R}^n$
Predict	$\hat{y}_t = \boldsymbol{w}_t \cdot \boldsymbol{x}_t$
Get label	$y_t \in \mathbf{R}$
Incur loss	$L_t(\boldsymbol{w}_t) = (y_t - \hat{y}_t)^2$
Update	$\boldsymbol{w}_t$ to $\boldsymbol{w}_{t+1}$



Assume comparison class $\{\boldsymbol{u}\}$ is a set of linear predictors

$$\boldsymbol{u} \ : \ \boldsymbol{x} \rightarrow \boldsymbol{u} \cdot \boldsymbol{x}$$

Examples of Updates

Gradient descent

($\boldsymbol{w} \in \mathbf{R}^n$)

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \nabla L_t(\boldsymbol{w}_t)$$

$$= \boldsymbol{w}_t - \eta(\boldsymbol{w}_t \cdot \boldsymbol{x}_t - y_t)\,\boldsymbol{x}_t \qquad \text{[WH]}$$

Exponentiated Gradient Algorithm [KW]

($\boldsymbol{w}$ is probability vector)

$$\boldsymbol{w}_{t+1,i} = w_{t,i}\ \exp\left[-\eta \frac{\partial L_t(\boldsymbol{w}_t)}{\partial w_{t,i}}\right] / \text{ normalization}$$

More examples of Updates

Unnormalized Exponentiated Gradient Algorithm [KW]

($\boldsymbol{w} \geq \mathbf{0}$)

$$\boldsymbol{w}_{t+1,i} = w_{t,i} \ \exp\left[-\eta \frac{\partial L_t(\boldsymbol{w}_t)}{\partial w_{t,i}}\right]$$

Binary Exponentiated Gradient Algorithm [By]

($\boldsymbol{w} \in [0,1]^n$)

$$\boldsymbol{w}_{t+1,i} = \frac{w_{t,i} \exp\left[-\eta \frac{\partial Nt(\boldsymbol{w}_t)}{\partial w_{t,i}}\right]}{1 - w_{t,i} + w_{t,i} \exp\left[-\eta \frac{\partial L_t(\boldsymbol{w}_t)}{\partial w_{t,i}}\right]}$$

More examples of Updates (cont)

p-norm Algorithms [GLS,GL]

($\boldsymbol{w} \in \mathbf{R}^n$)

$$\boldsymbol{w}_{t+1} = f^{-1}\big(f(\boldsymbol{w}_t) - \eta\, \nabla L_t(\boldsymbol{w}_t)\big)$$

where

$$f(\boldsymbol{w}) = \nabla \frac{1}{2}||\boldsymbol{w}||_q^2 = \nabla \frac{1}{2}\left(\sum_i |w_i|^q\right)^{2/q}$$

and q dual to p (i.e., $\frac{1}{p} + \frac{1}{q} = 1$)

- $p = 2$ becomes gradient descent
- $p = O(\log n)$ becomes EG-like algs
- $2 < p < O(\log n)$ interpolates between the two extremes

Motivation of Updates [KW]

Gradient descent

$$\begin{aligned} \boldsymbol{w}_{t+1} &= \operatorname*{argmin}_{\boldsymbol{w}} \left(||\boldsymbol{w} - \boldsymbol{w}_t||_2^2/2 + \eta(y_t - \boldsymbol{w} \cdot \boldsymbol{x}_t)^2/2 \right) \\ &= \boldsymbol{w}_t - \eta(\underbrace{\boldsymbol{w}_{t+1} \cdot \boldsymbol{x}_t}_{\approx \boldsymbol{w}_t \cdot \boldsymbol{x}_t} - y_t)\, \boldsymbol{x}_t \end{aligned}$$

Exponentiated Gradient Algorithm

$$\begin{aligned} \boldsymbol{w}_{t+1} &= \operatorname*{argmin}_{\boldsymbol{w}} \left(\sum_{i=1}^{n} w_i \ln \frac{w_i}{w_{t,i}} + \eta(y_t - \boldsymbol{w} \cdot \boldsymbol{x}_t)^2/2 \right) \\ &= w_{t,i} \ \exp \left[-\eta \, (\underbrace{\boldsymbol{w}_{t+1} \cdot \boldsymbol{x}_t}_{\approx \boldsymbol{w}_t \cdot \boldsymbol{x}_t} - y_t)\, x_{t,i} \right] / \text{ normalization} \end{aligned}$$

Alternate Motivation: Continuous Updates [WJ]

Continuous time $\Rightarrow$ Ordinary differential equations

Gradient Descent

$\boldsymbol{w} \in \mathbf{R}^n$

$$\dot{\boldsymbol{w}}_t = -\eta \nabla_{\boldsymbol{w}} L_t(\boldsymbol{w}_t)$$

Unnormalized Exponentiated Gradient Alg.

$\boldsymbol{w} \geq \mathbf{0}$

$$\log(\dot{\boldsymbol{w}}_t) = -\eta \nabla_{\boldsymbol{w}} L_t(\boldsymbol{w}_t)$$

Alternate Motivation: Continuous Updates (cont)

Discretization

$$\frac{\boldsymbol{f}(\boldsymbol{w}_{t+h}) - \boldsymbol{f}(w_t)}{h} = -\eta \nabla L_t(\boldsymbol{w}_t)$$

$$w_{t+h} = \boldsymbol{f}^{-1}\left(\boldsymbol{f}(w_t) - \eta\, h \nabla_{\boldsymbol{w}} L_t(\boldsymbol{w}_t)\right)$$

We use $h = 1$

$$\boldsymbol{w}_{t+1} = \boldsymbol{f}^{-1}\left(\boldsymbol{f}(\boldsymbol{w}_t) - \eta_t \nabla L_t(\boldsymbol{w}_t)\right)$$

Conjecture: Forward Euler better:

Replace $\nabla_{\boldsymbol{w}} L_t(\boldsymbol{w}_t)$ by $\nabla_{\boldsymbol{w}} L_t(\boldsymbol{w}_{t+h})$

Families of update algorithms

parameter "divergence"	name of family	update algorithms
$\|\|\boldsymbol{w} - \boldsymbol{w}_t\|\|_2^2$	Gradient Descent	Widrow Hoff (LMS) Linear Least Squares. Backpropagation Perceptron Algorithms kernel based algorithms,...
$\sum_{i=1}^{n} w_i \ln \frac{w_i}{w_{t,i}}$	Exponentiated Gradient Algorithm	expert algs Normalized Winnow "AdaBoost"

Families of update algorithms (cont)

parameter "divergence"	name of family	update algorithms
$\sum_{i=1}^{n} w_i \ln \frac{w_i}{w_{t,i}}$ $+w_{t,i} - w_i$	Unnormalized Exp. Grad. Alg.	Winnow
$\sum_{i=1}^{n} w_i \ln \frac{w_i}{w_{t,i}}$ $+(1-w_i) \ln \frac{1-w_i}{1-w_{t,i}}$	Binary Exp. Grad. Alg.	
any	-	-

"Bregman divergence"

Members of different families exhibit different behavior

Additive Gradient versus Exponentiated Gradient

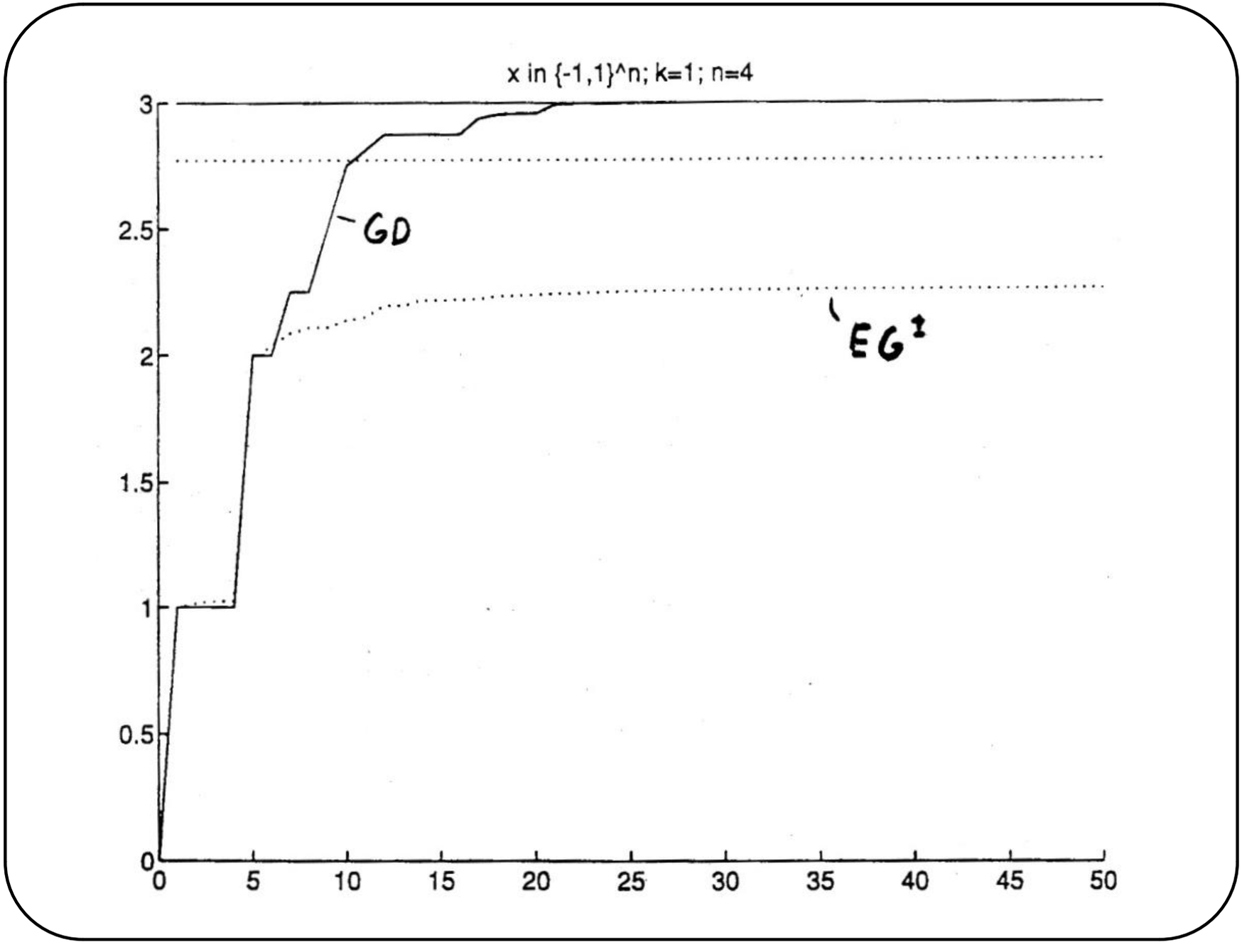
An experimental study: Part I

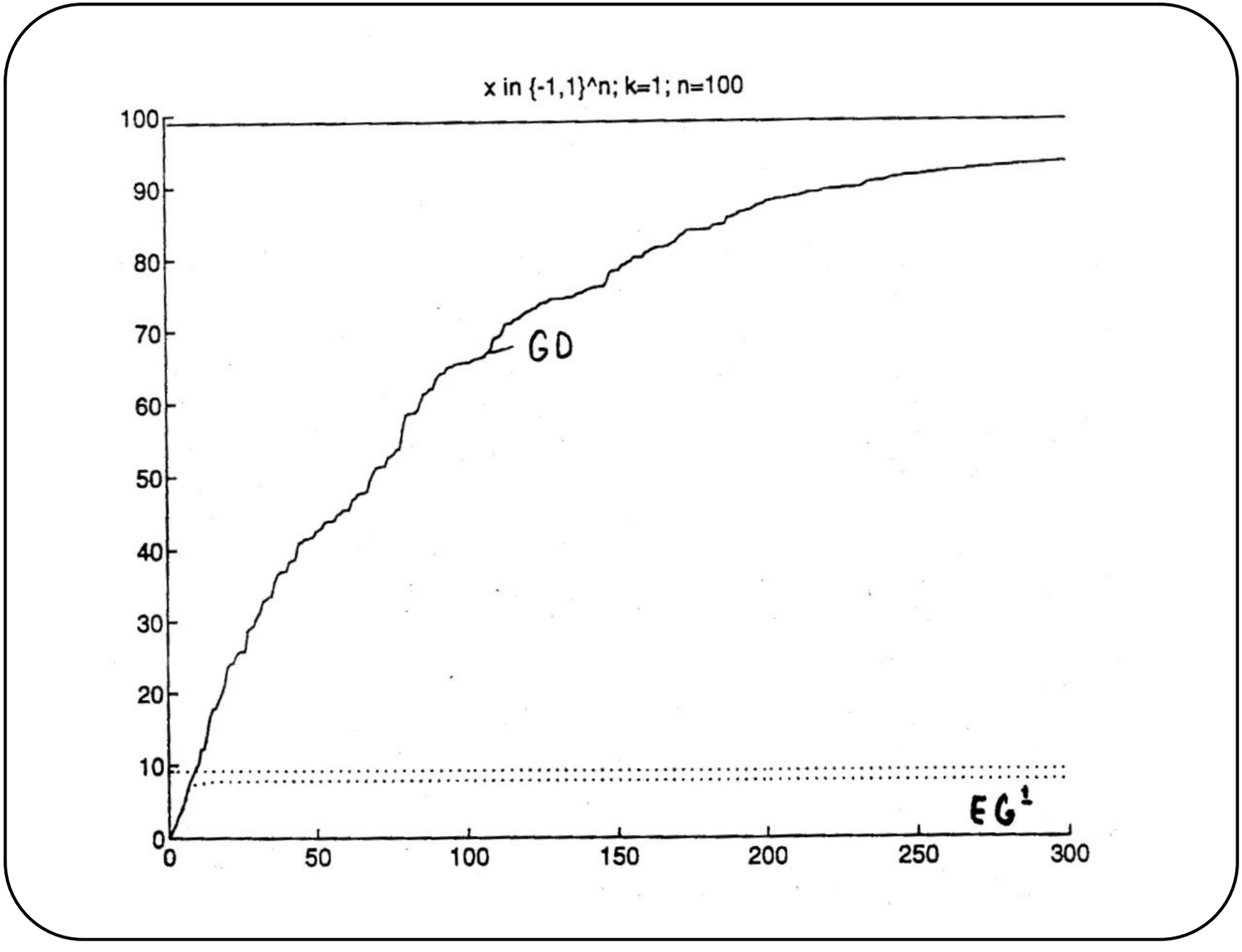
The instances are random vectors $\boldsymbol{x} \in \{-1, +1\}^n$

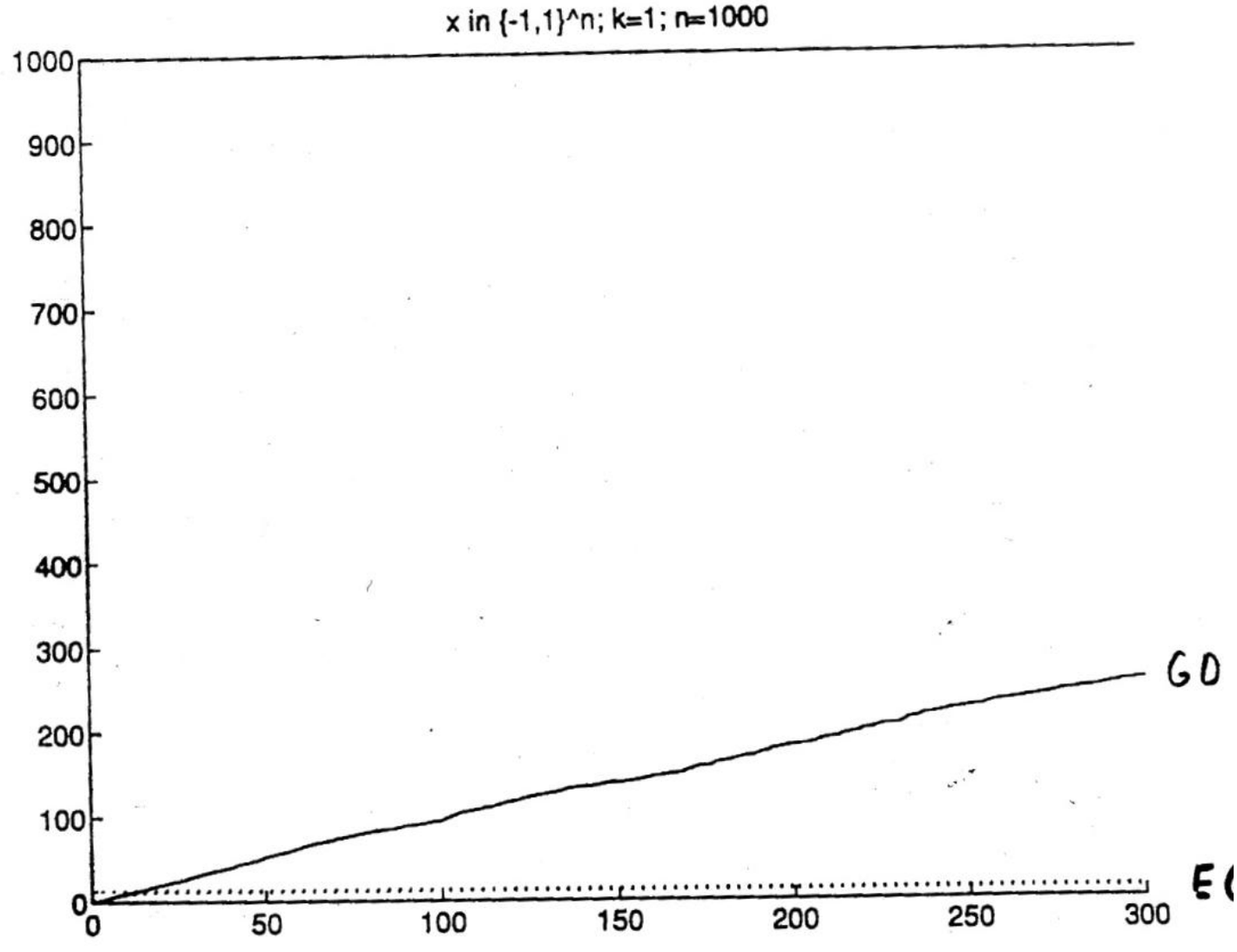
The target vector $\mathbf{u}$ is sparse, e.g. $\mathbf{u} = [\underbrace{1, 1}_{k \text{ times}}, 0, \ldots, 0]$

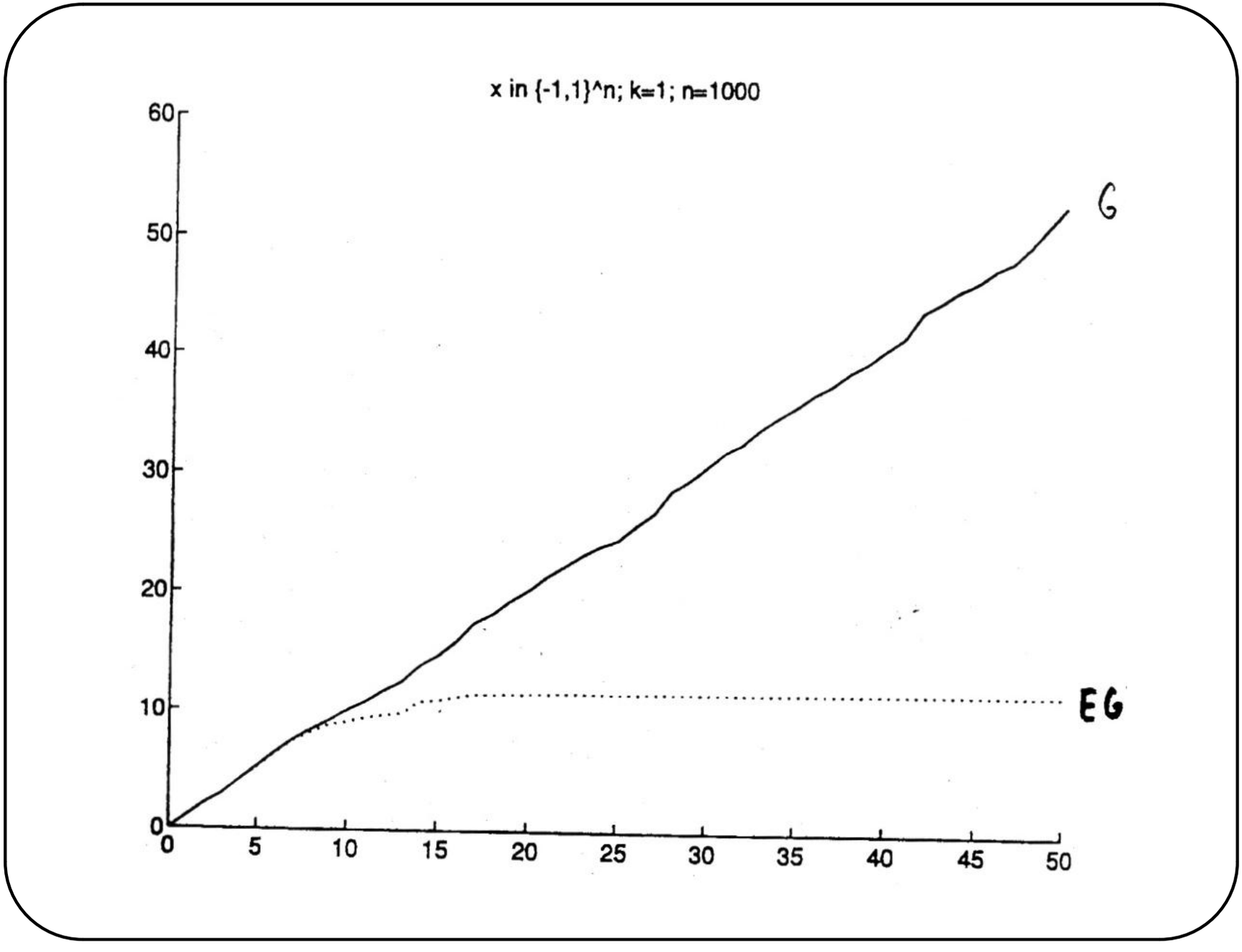
The labels are given $y_t = \mathbf{u} \cdot \boldsymbol{x}_t$

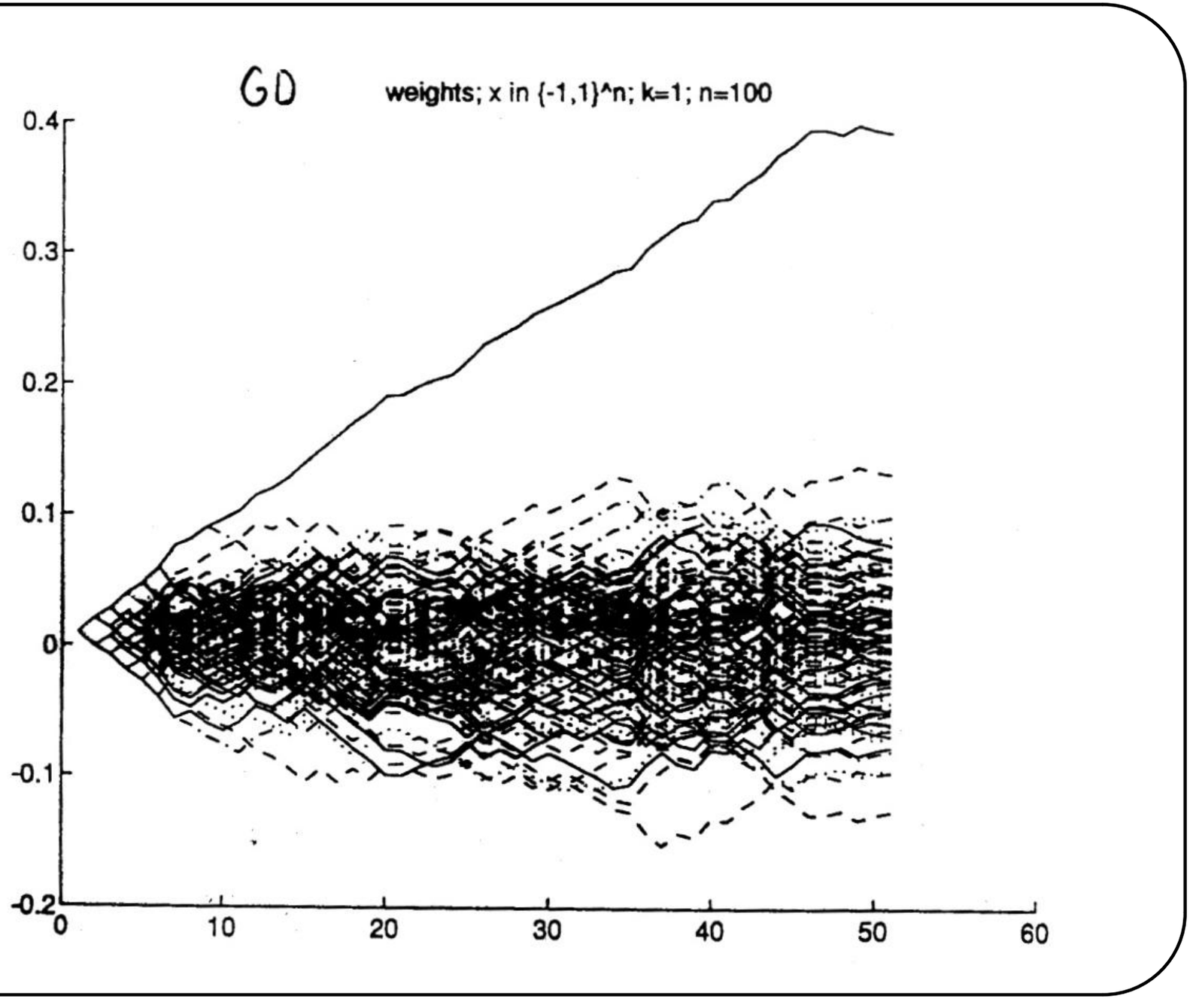
Loss function is the quadratic loss $L(y, \hat{y}) = (y - \hat{y})^2$

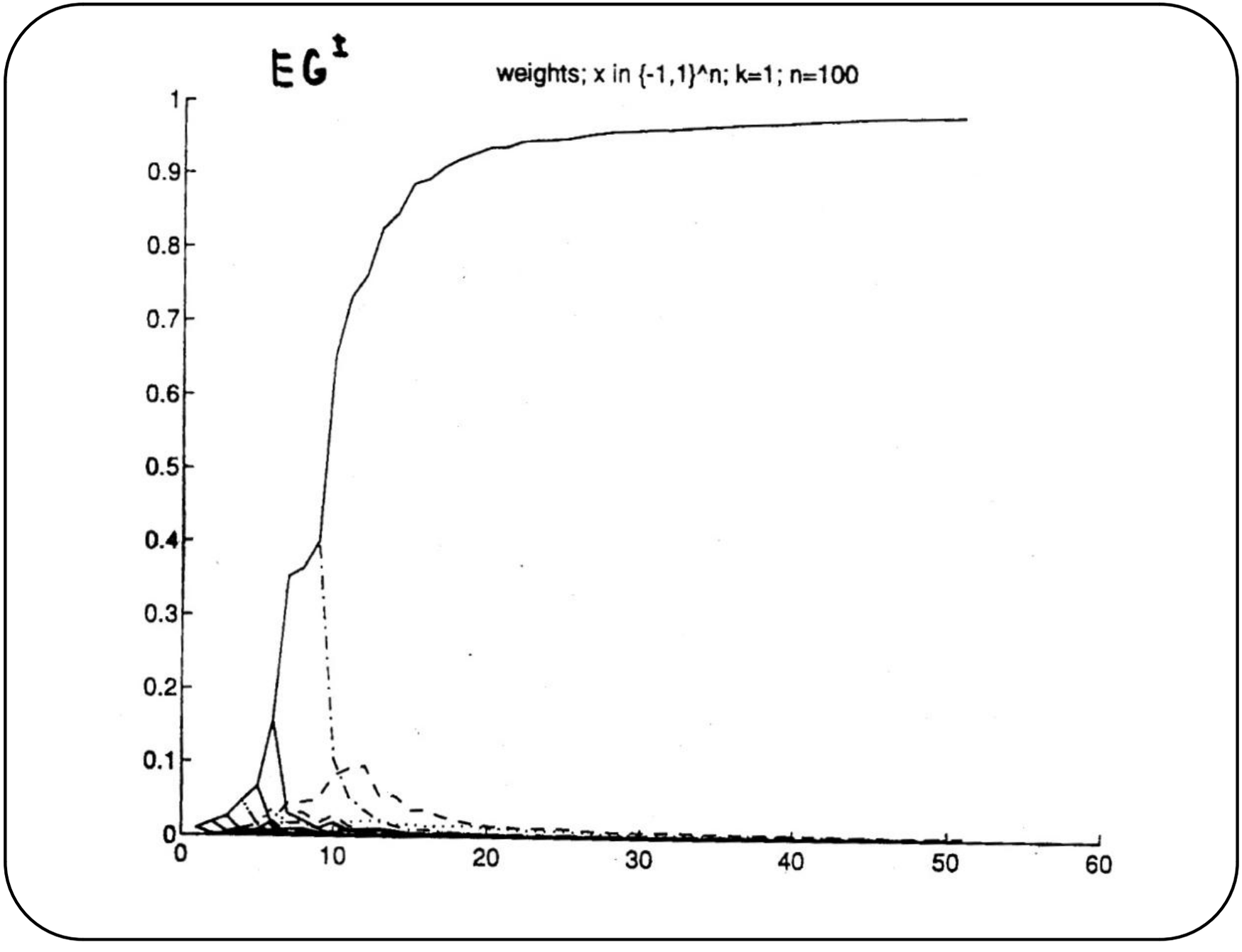

x in {-1,1}^n; k=1; n=4
3
2.5
2
1.5
1
0.5
0
0
5
10
15
20
25
30
35
40
45
50
GD
EG±


x in {-1,1}^n; k=1; n=100
100
90
80
70
60
50
40
30
20
10
0
0
50
100
150
200
250
300
GD
EG±


x in {-1,1}^n; k=1; n=1000
1000
900
800
700
600
500
400
300
200
100
0
0
50
100
150
200
250
300
GD
E(


x in {-1,1}^n; k=1; n=1000
60
50
40
30
20
10
0
0
5
10
15
20
25
30
35
40
45
50
G
EG


GD
weights; x in {-1,1}^n; k=1; n=100
0.4
0.3
0.2
0.1
0
-0.1
-0.2
0
10
20
30
40
50
60


EG±
weights; x in {-1,1}^n; k=1; n=100
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
10
20
30
40
50
60

Loss bounds

Assume Example Sequence is

$$(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_t, y_t), \ldots \text{ where } y_t = \mathbf{u} \cdot \boldsymbol{x}_t$$

(the zero-error case)

Gradient Descent:

$$L_{GD}(S) \leq \left(\|\mathbf{u}\|_2 \max_t \|\boldsymbol{x}_t\|_2 \right)^2$$

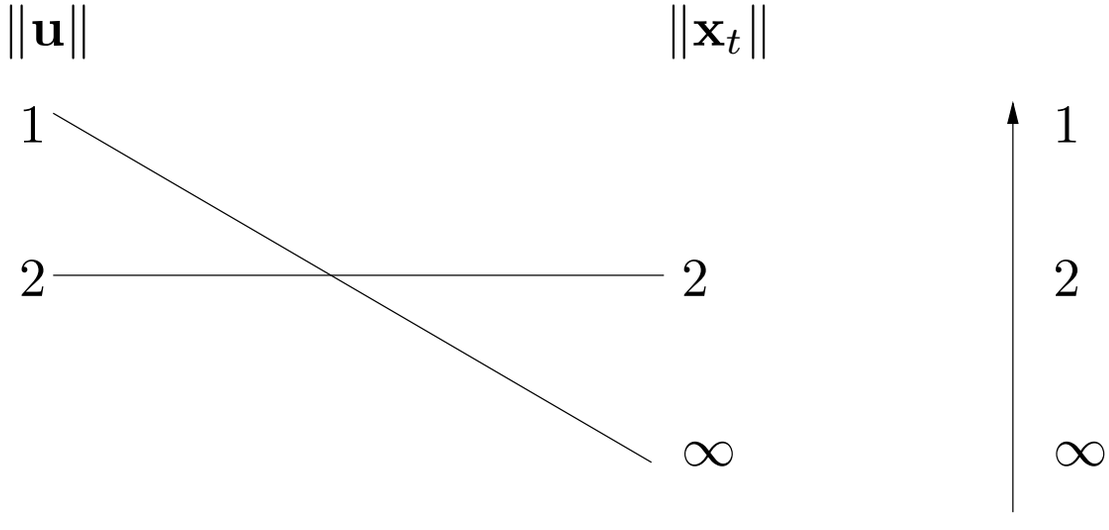
Exponentiated Gradients:

$$L_{EG\pm}(S) \leq \left(\|\mathbf{u}\|_1 \max_t \|\boldsymbol{x}_t\|_\infty \right)^2 \log(2n)$$

Incomparable Loss bounds

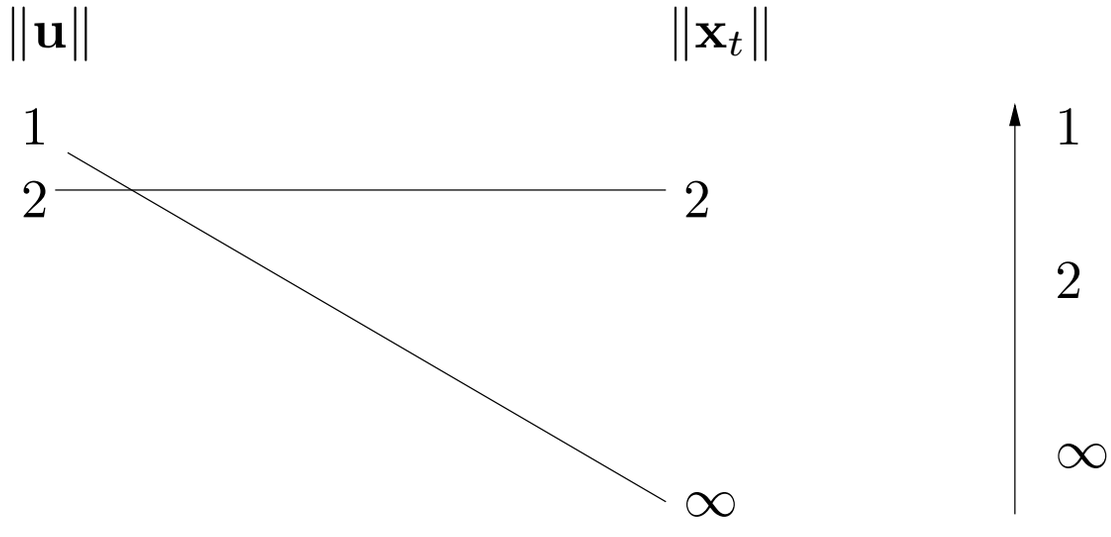
$$
\begin{aligned}
L_{GD}(S) &\leq \left(\|\mathbf{u}\|_2 \max_t \|\boldsymbol{x}_t\|_2\right)^2 \\
L_{EG\pm}(S) &\leq \left(\|\mathbf{u}\|_1 \max_t \|\boldsymbol{x}_t\|_\infty\right)^2 \log(2n)
\end{aligned}
$$

Products of two norms:



Incomparable Loss bounds (cont)

Experiments so far:



Widest gap between ℓ_2 and ℓ_∞ norms if $\boldsymbol{x} \in \{-1, +1\}^N$

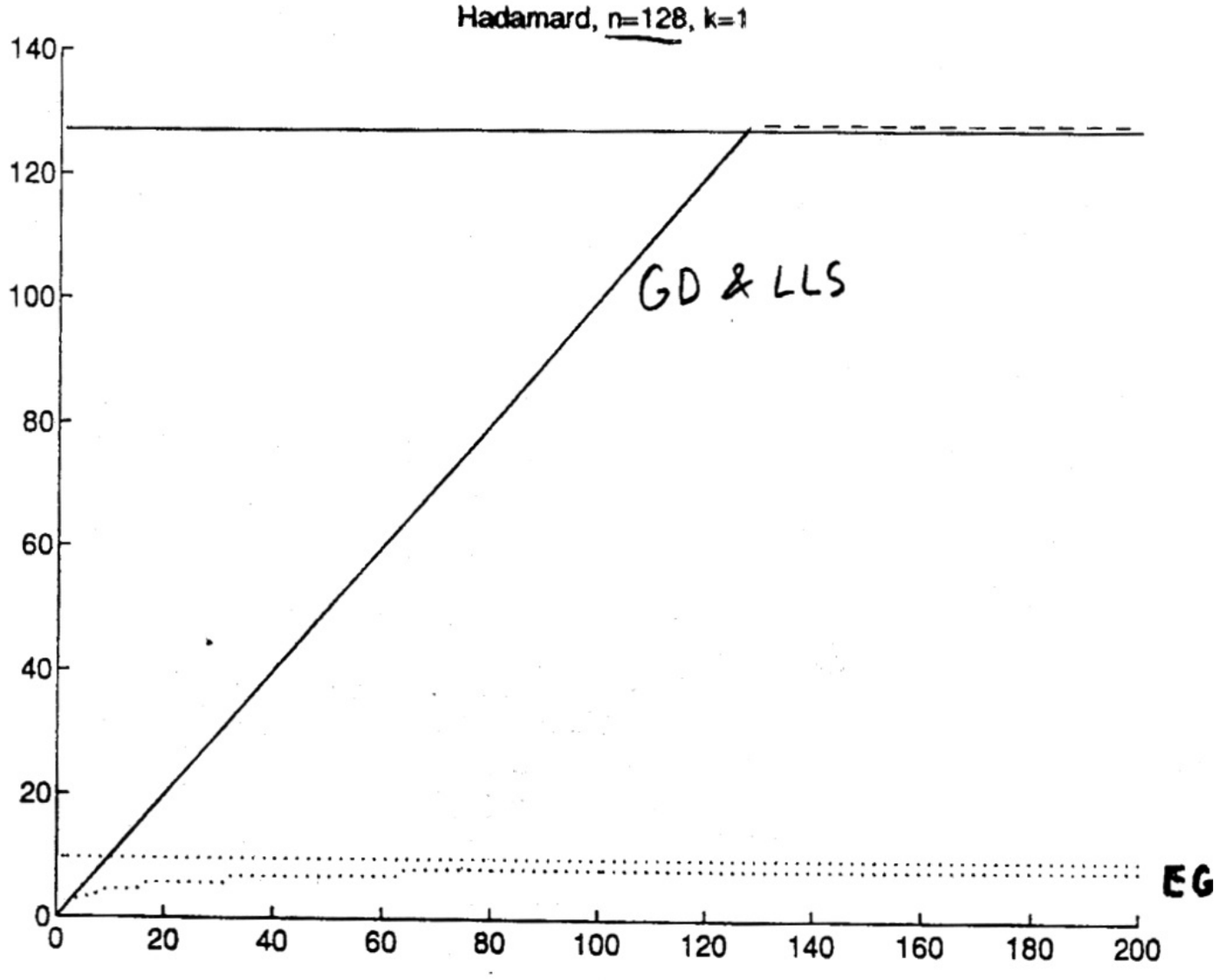
The Worst Case: Hadamard Matrices

$$H_1 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad H_2 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}, \quad \cdots$$

$$H_{t+1} = \begin{pmatrix} H_t & H_t \\ H_t & -H_t \end{pmatrix}$$

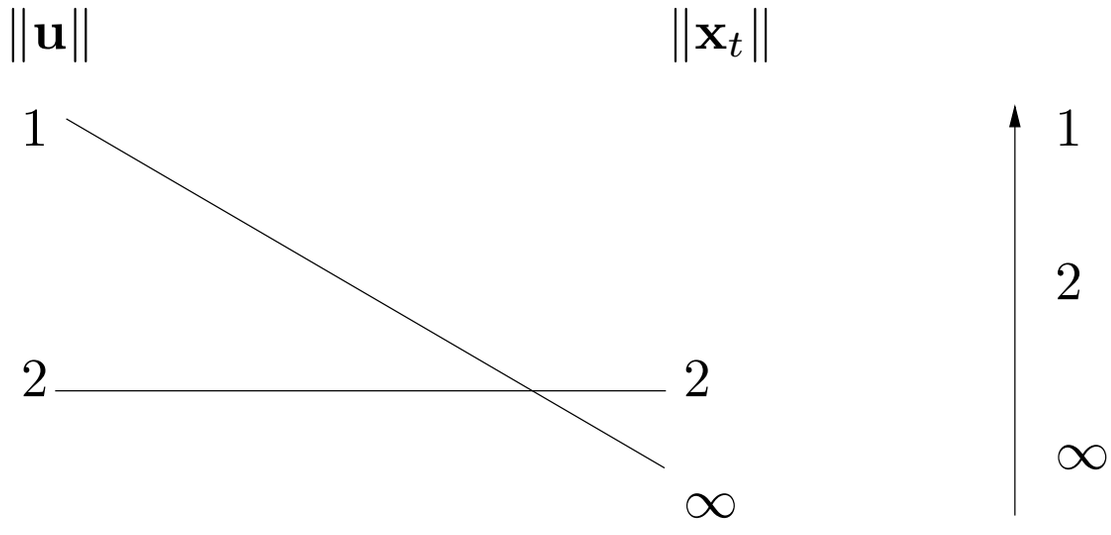
All Rows are Orthogonal!

Use Rows as Instances!


Hadamard, n=128, k=1
140
120
100
80
60
40
20
0
0
20
40
60
80
100
120
140
160
180
200
GD & LLS
EG

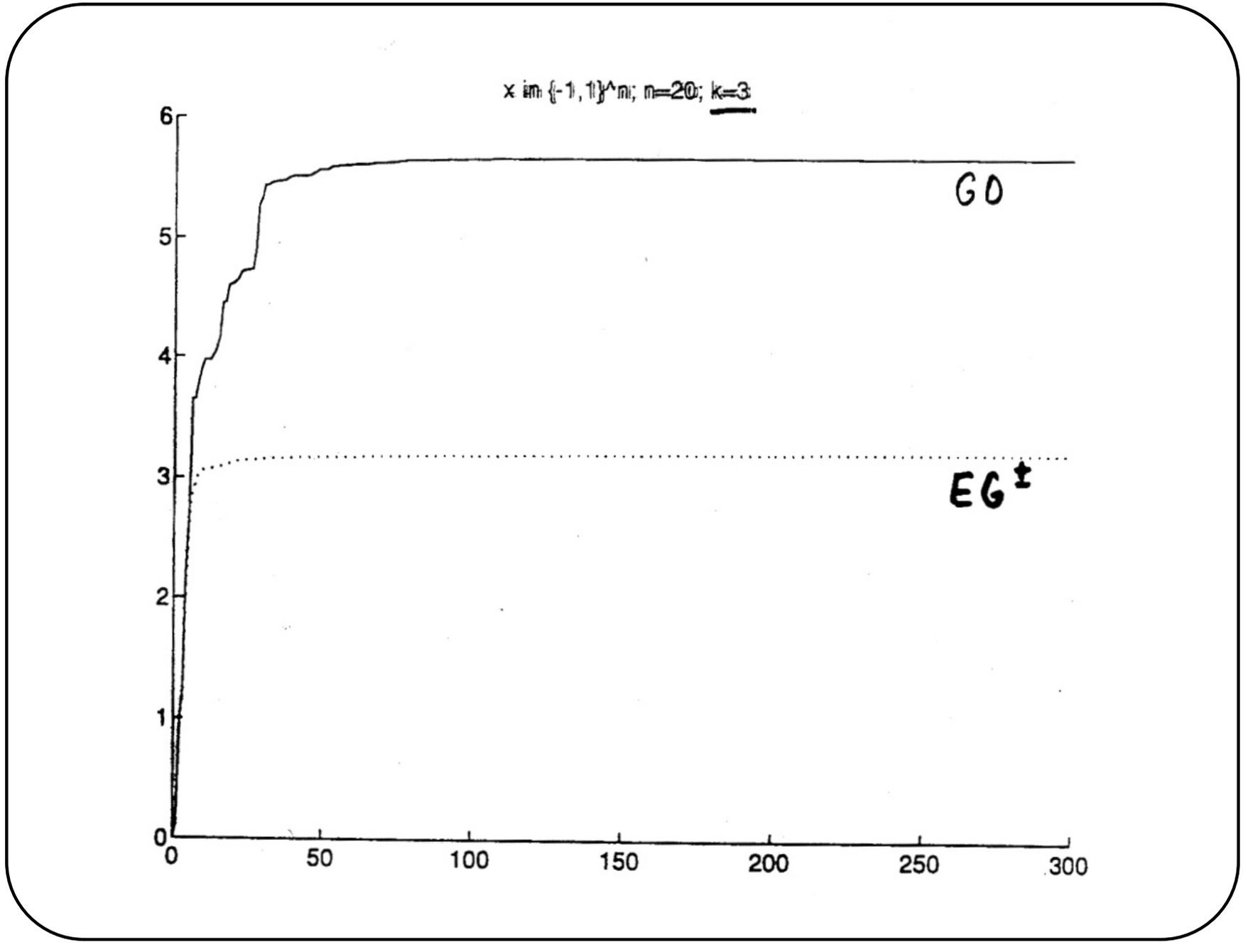
Incomparable Loss bounds (cont)

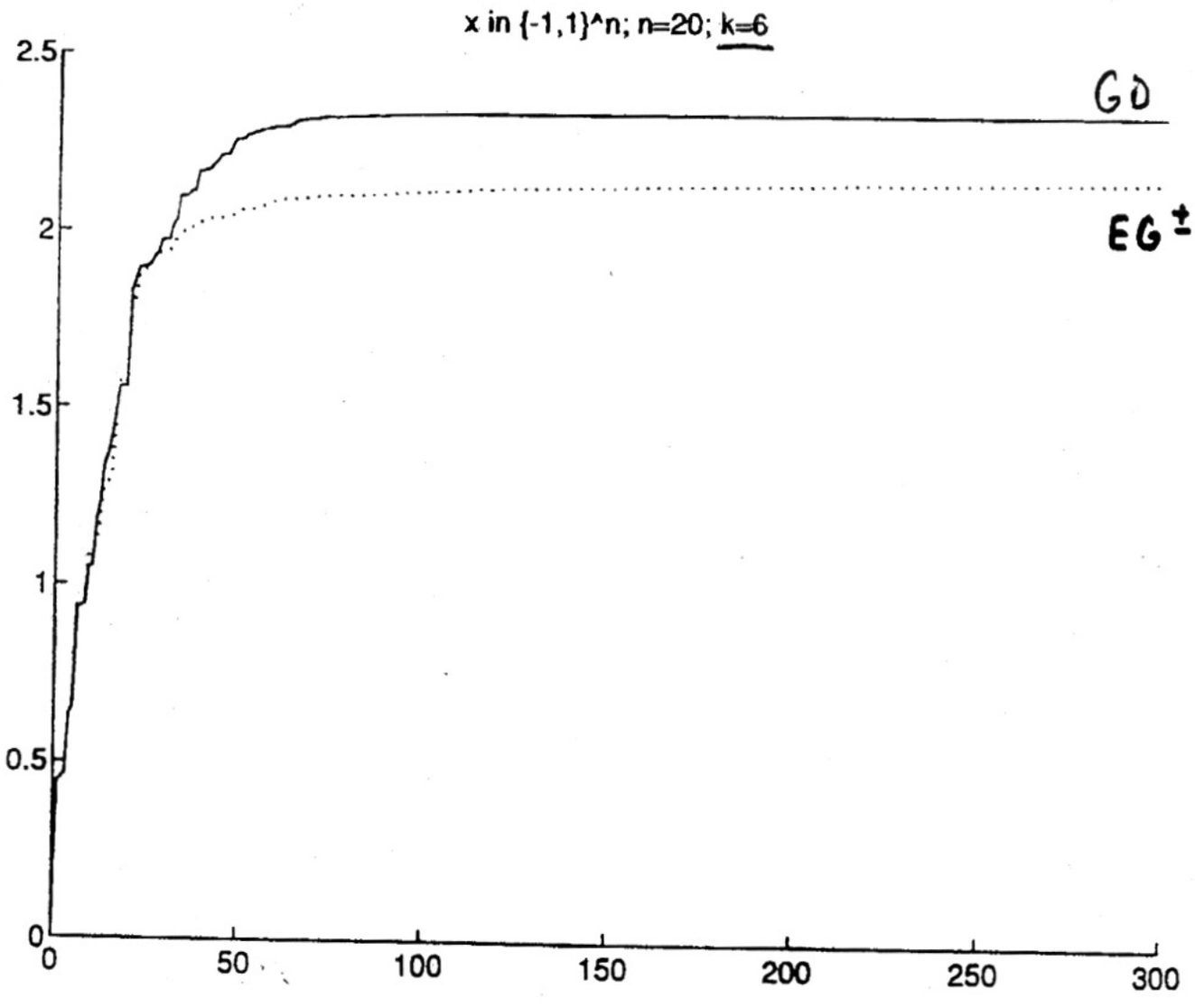
Gradient Descent has advantage:

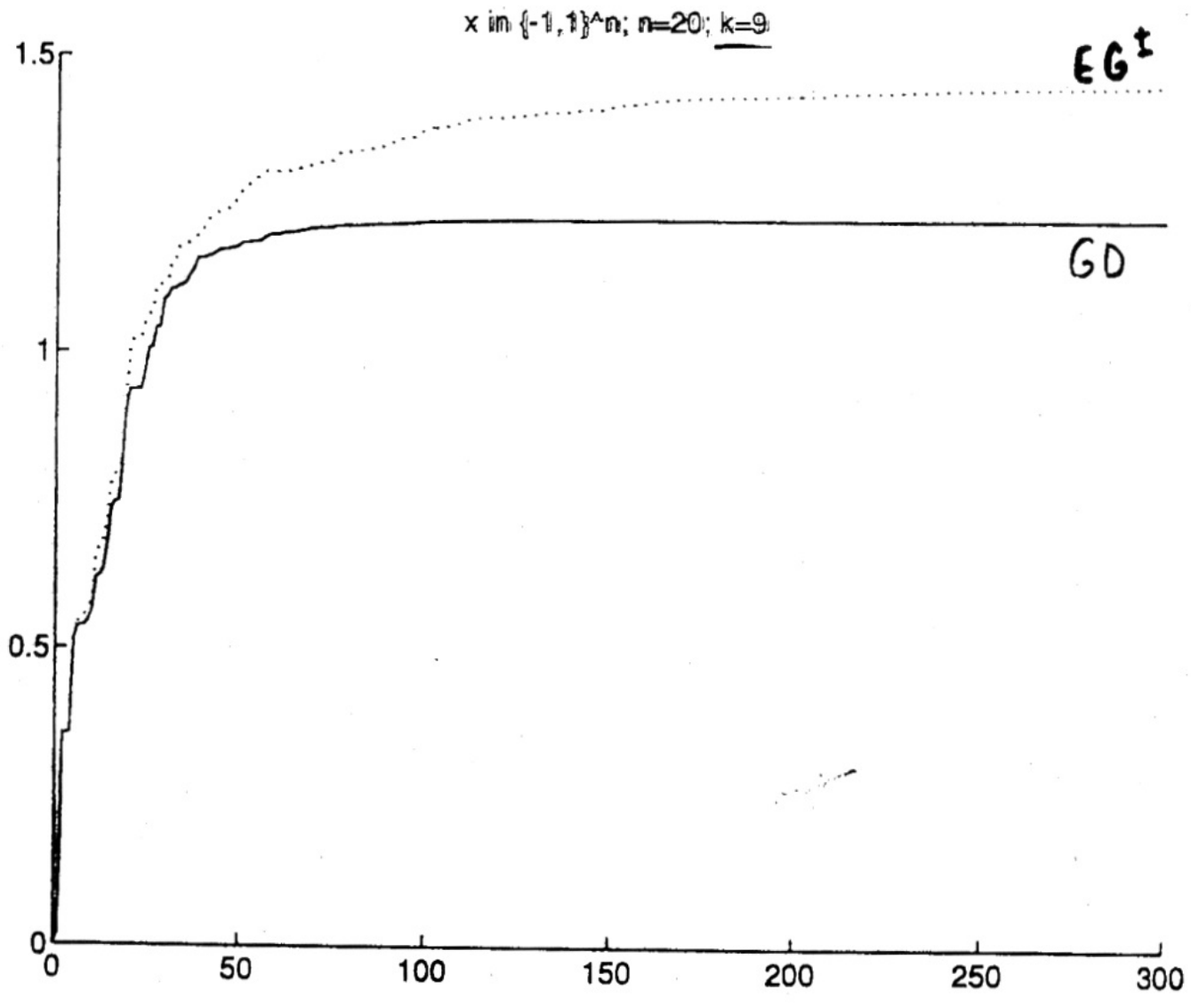


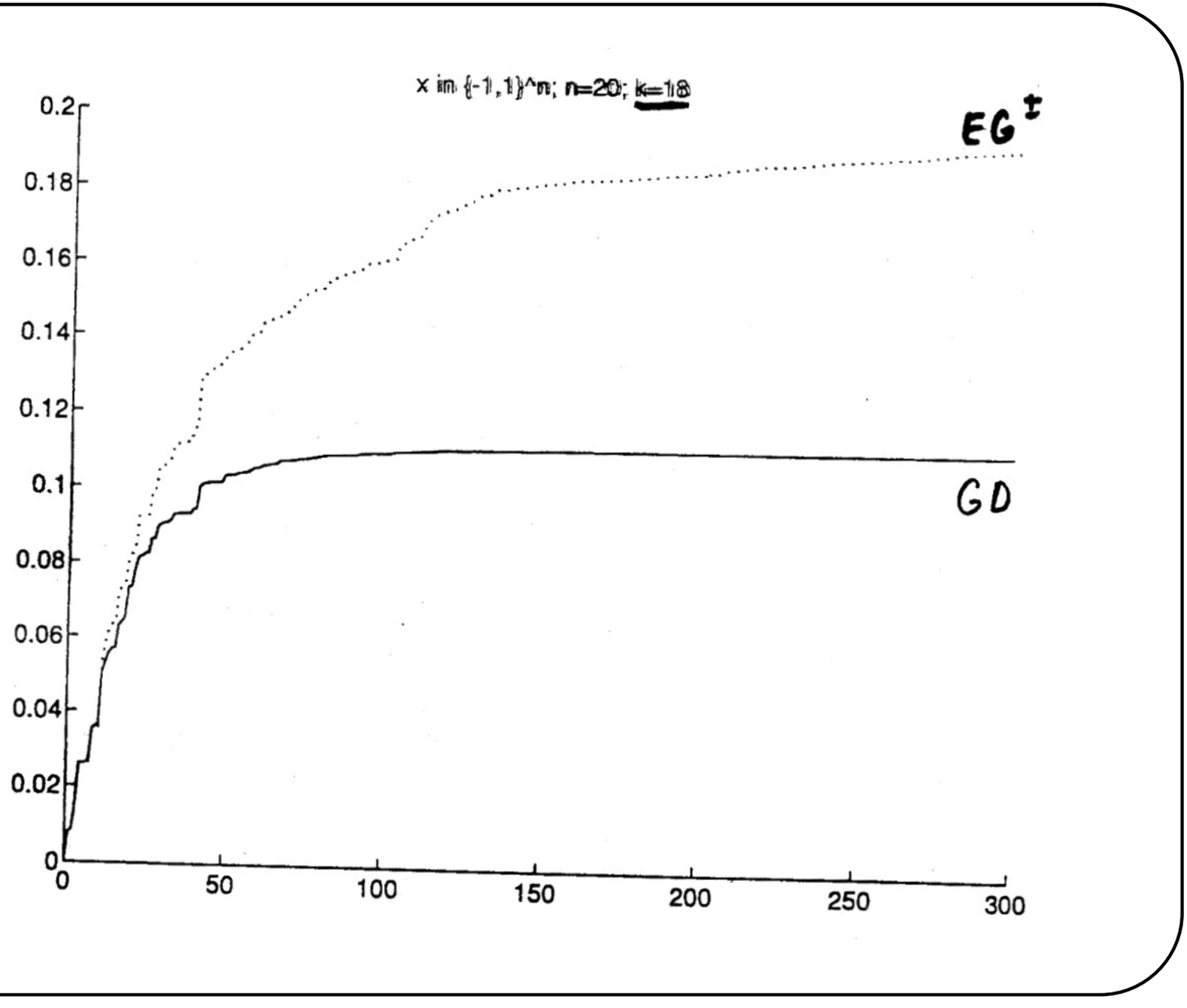
Widest gap between ℓ_1 and ℓ_2 norms if comparator $\mathbf{u}$ is dense

. . . and there is the additional factor $\log N$ for EG


x in {-1,1}^n; n=20; k=3
GD
EG±
0
1
2
3
4
5
6
0
50
100
150
200
250
300


x in {-1,1}^n; n=20; k=6
GD
EG±
2.5
2
1.5
1
0.5
0
0
50
100
150
200
250
300


x in {-1,1}^n; n=20; k=9
EG±
GD
1.5
1
0.5
0
0
50
100
150
200
250
300


x in {-1,1}^n; n=20; k=18
EG±
GD
0.2
0.18
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
0
50
100
150
200
250
300

Summary of Comparison

- EG better when:

- Instances $\boldsymbol{x}_t$ are dense ($\|\boldsymbol{x}_t\|_\infty \ll \|\boldsymbol{x}_t\|_2$)

- best weight vector is sparse ($\|\boldsymbol{u}_t\|_1 \approx \|\boldsymbol{u}_t\|_2$)

- GD better when:

- instances are sparse ($\|\boldsymbol{x}_t\|_\infty \approx \|\boldsymbol{x}_t\|_2$)

- best weight vector is dense ($\|\boldsymbol{u}_t\|_2 \ll \|\boldsymbol{u}_t\|_2$)

GD can be exponentially worse than EG