Leaving The Span

Manfred K. Warmuth and Vishy Vishwanathan

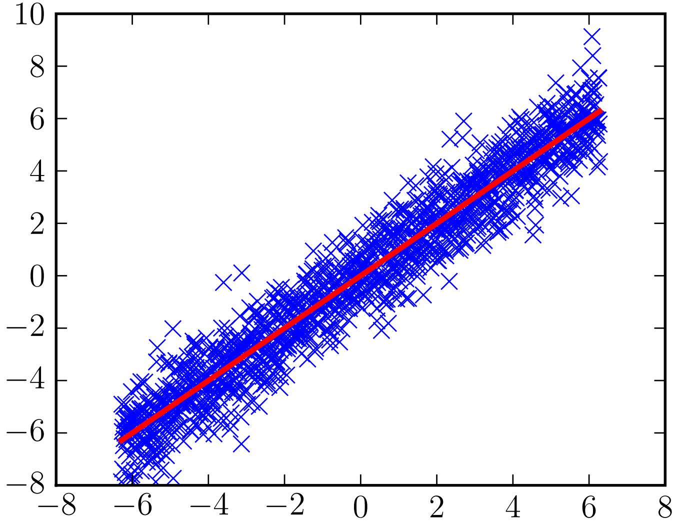
UCSC and NICTA

Talk at NYAS Conference, 10-27-06

Thanks to Dima and Jun

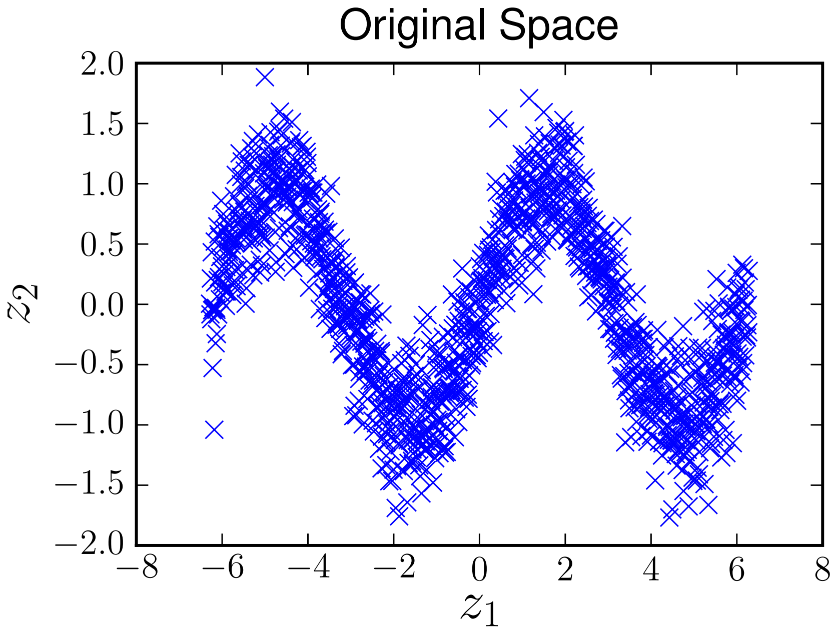
Let's keep it simple

Linear Regression



- Examples $(\mathbf{x}_t, y_t)$
- Linear hypothesis $\mathbf{w}$
- Predicts with $\hat{y}_t = \mathbf{w} \cdot \mathbf{x}_t$

What if data not close to linear

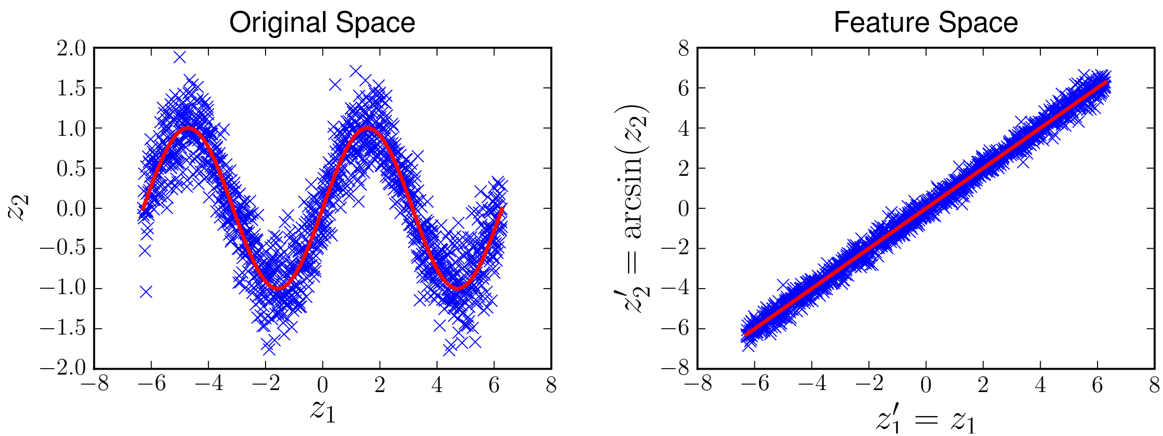


- Simply invent new variables/features :-)

Close to linear in feature space

Embed instances into a feature space

$$\phi : \mathbb{R}^n \rightarrow \mathbb{R}^m$$



- $\phi(x_1, x_2) = (\underbrace{\arcsin(x_1)}_{x'_1}, \underbrace{x_2}_{x'_2})$

Does the expansion always work

- Can you always improve things by inventing new features
- Fitting the data may be - But is this learning?

The Kernel Trick [BGV92]

If $\mathbf{w}$ linear combination of expanded instances, then

$$\hat{y} = \underbrace{\sum_t \alpha_t\, \phi(\mathbf{x}_t)}_{\mathbf{w}} \cdot \phi(\mathbf{x}) = \sum_t \alpha_t\, \underbrace{\phi(\mathbf{x}_t) \cdot \phi(\mathbf{x})}_{K(\mathbf{x}_t, \mathbf{x})}$$

Kernel function $K(\mathbf{x}_t, \mathbf{x})$ often efficient to compute

$$\phi(\ \underbrace{(x_1, \ldots, x_n)}_{n}\ ) = \underbrace{(1, \ldots, x_i, \ldots, x_i x_j \ldots, x_i x_j x_k \ldots)}_{2^n \text{ products}}$$

Kernel magic

$$K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{z}) = \underbrace{\sum_{I \subseteq 1..n} \prod_{i \in I} x_i \prod_{i \in I} z_i}_{O(2^n) \text{ time}} = \underbrace{\prod_{i=1}^{n} (1 + x_i z_i)}_{O(n) \text{ time}}$$

Good news

Many of our favorite algorithms can be "kernelized":

Linear Least Squares, Widrow-Hoff, Support Vector Machines, PCA, Simplex Algorithm, ...

Kernel Trick:

- Weight vector linear combination of embedded instances
- Individual features never accessed

Linear combinations ?

Representer Theorem: [KW71]

$$\mathbf{w} = \operatorname*{arginf}_{\mathbf{w}'} \left(||\,\mathbf{w}'\,||^2 + \eta \sum_t (\mathbf{w}' \cdot \phi(\mathbf{x}_t) - y_t)^2 \right)$$

Solution $\mathbf{w}$ linear combination of the $\phi(\mathbf{x}_t)$

Rotation invariance: [KWA97]

Any algorithm whose predictions are not affected by rotating the instances in feature space
must predict with linear combination of embedded instances

Sufficient conditions!

Linear or non-linear ?

- We give a problem for which kernel algorithms behave like linear algorithms
- Embeddings don't help

:-(

A hard problem

Hadamard Matrix:

$$\begin{array}{rcrrrr} & \rightarrow & +1 & +1 & +1 & +1 \\ instances & \rightarrow & +1 & -1 & +1 & -1 \\ & \rightarrow & +1 & +1 & -1 & -1 \\ & \rightarrow & +1 & -1 & -1 & +1 \\ & & \uparrow & \uparrow & \uparrow & \uparrow \\ & & \multicolumn{4}{c}{targets} \end{array}$$

- n instances and n targets
- Instances are orthogonal
- Target weight vectors are units

The n data sets

$$
\begin{array}{ll}
((+1,+1,+1,+1),+1) & ((+1,+1,+1,+1),+1) \\
((+1,-1,+1,-1),+1) & ((+1,-1,+1,-1),-1) \\
((+1,+1,-1,-1),+1) & ((+1,+1,-1,-1),+1) \\
((+1,-1,-1,+1,)+1) & ((+1,-1,-1,+1),-1)
\end{array}
$$

$$
\begin{array}{ll}
((+1,+1,+1,+1),+1) & ((+1,+1,+1,+1),+1) \\
((+1,-1,+1,-1),+1) & ((+1,-1,+1,-1),-1) \\
((+1,+1,-1,-1),-1) & ((+1,+1,-1,-1),-1) \\
((+1,-1,-1,+1),-1) & ((+1,-1,-1,+1),+1)
\end{array}
$$

For each of the n data sets

- Subset of labeled examples is received
- Labels of remaining examples must be predicted
- Loss is averaged over all n examples

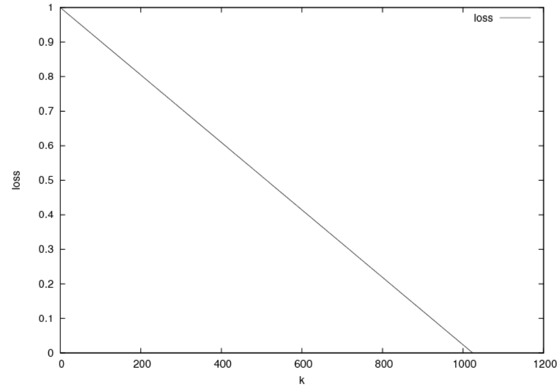
Without embeddings I

Any linear combination of k training instances predicts zero on all $n - k$ test instances [LLW95,KWA97]

So loss 1 on $n - k$ of the n instances

Average square loss over all n instances is

$$\geq 1 - \frac{k}{n}$$



Without embeddings II

Theorem

For any linear combination of k rows of the n-dimensional Hadamard matrix and any of the n targets
the average square loss over all n instances is

$$\geq 1 - \frac{k}{n}$$

Theorem

Any linear combination of k rows of n dimensional Hadamard matrix has

$$\text{distance } \geq \sqrt{1 - \frac{k}{n}}$$

from each of the n unit vectors

With embeddings

$$\phi : \underbrace{\begin{matrix} +1 & +1 & +1 & +1 \\ +1 & -1 & +1 & -1 \\ +1 & +1 & -1 & -1 \\ +1 & -1 & -1 & +1 \end{matrix}}_{H} \quad \rightarrow \quad \underbrace{\begin{matrix} +1 \\ -1 \\ +1 \\ -1 \end{matrix}}_{Z}$$

So after one example you learned one target

Caveat: this embedding does poorly on the other targets

$$\phi : \underbrace{\begin{matrix} +1 & +1 & +1 & +1 \\ +1 & -1 & +1 & -1 \\ +1 & +1 & -1 & -1 \\ +1 & -1 & -1 & +1 \end{matrix}}_{H} \quad \rightarrow \quad \underbrace{\begin{matrix} +1 & +1 & +1 \\ +1 & -1 & +1 \\ +1 & +1 & -1 \\ +1 & -1 & -1 \end{matrix}}_{k \text{ rows}}$$

With k independent examples you can learn first k targets

Summary

$$
\begin{array}{ccccc}
\rightarrow & +1 & +1 & +1 & +1 \\
\rightarrow & +1 & -1 & +1 & -1 \\
\rightarrow & +1 & +1 & -1 & -1 \\
\rightarrow & +1 & -1 & -1 & +1 \\
 & \uparrow & \uparrow & \uparrow & \uparrow
\end{array}
$$

Memorize labels of first k instances

$$
\begin{array}{ccccc}
\rightarrow & +1 & +1 & +1 & +1 \\
\rightarrow & +1 & -1 & +1 & -1 \\
\rightarrow & +1 & +1 & -1 & -1 \\
\rightarrow & +1 & -1 & -1 & +1 \\
 & \uparrow & \uparrow & \uparrow & \uparrow
\end{array}
$$

Correct on k targets

No improvement possible

Main Result

Theorem

- No matter how the instances are embedded
- No matter what k training instances chosen by the learner
- No matter what linear combination used

For one of the targets
average square loss on all n instances is $1 - \frac{k}{n}$

Probabilistic model I

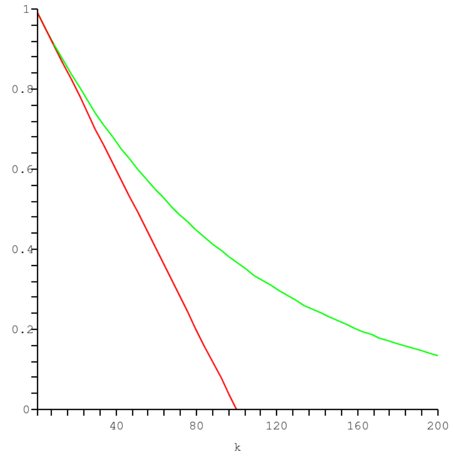
- Uniform distribution on the n rows of Hadamard matrix
- Algorithm first embeds the n rows and then draws k rows without replacement $-$ all labeled by one of the n targets.
- Chooses hypothesis as linear combination of the k embedded instances
- Average square loss for at least one of the targets is

$$\geq 1 - \frac{k}{n}$$

Probabilistic model II

- As above but k examples are drawn with replacement
- Average square loss for at least one of the targets is

$$\geq (1 - \frac{1}{n})^k$$

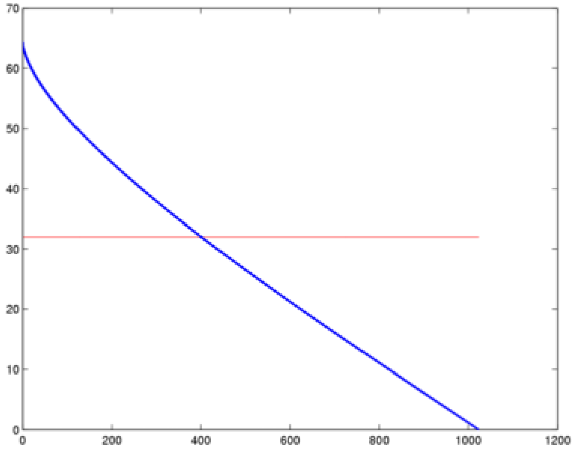


Without replacement With replacement

$n = 100$

Our Approach

- Use the SVD spectrum instead



Hadamard Random

$$\text{Average square loss} \geq \frac{1}{n^2} \sum_{i=k+1}^{n} s_i^2$$

$$= 1 - \frac{k}{n}$$

Proof

$\underset{n \times n}{\mathbf{H}}$	mapped to $\underset{n \times m}{\mathbf{Z}}$
$\underset{k \times m}{\widehat{\mathbf{Z}}}$	first k rows
$\widehat{\mathbf{Z}}^T \underset{k \times 1}{\mathbf{a}}$	weight vector
$\mathbf{Z}\,\widehat{\mathbf{Z}}^T\,\mathbf{a} - \mathbf{h}$	residuals for one target
$\mathbf{Z}\,\widehat{\mathbf{Z}}^T \underset{k \times n}{\mathbf{A}} - \mathbf{H}$	all n^2 residuals
$\frac{1}{n^2} \|\| \underbrace{\mathbf{Z}\,\widehat{\mathbf{Z}}^T\,\mathbf{A}}_{\text{rank } k} - \mathbf{H} \|\|_F^2$	average squared error
$\geq \frac{1}{n^2} \sum_{i=k+1}^{n} s_i^2$	

Proof for non-square $\mathbb{H}$

$\underset{n\times q}{\mathbf{H}}$	mapped to $\underset{n\times m}{\mathbf{Z}}$
$\underset{k\times m}{\widehat{\mathbf{Z}}}$	first k rows
$\widehat{\mathbf{Z}}^T \underset{k\times 1}{\mathbf{a}}$	weight vector
$\mathbf{Z}\,\widehat{\mathbf{Z}}^T\,\mathbf{a}-\mathbf{h}$	residuals for one target
$\mathbf{Z}\,\widehat{\mathbf{Z}}^T \underset{k\times q}{\mathbf{A}} - \mathbf{H}$	all n^2 residuals
$\frac{1}{nq}\lvert\lvert \underbrace{\mathbf{Z}\,\widehat{\mathbf{Z}}^T\,\mathbf{A}}_{\text{rank } k} - \mathbf{H}\,\rvert\rvert_F^2$	average squared error
$\geq \frac{1}{nq}\sum_{i=k+1}^{\min(n,q)} s_i^2$	

Additional Constraints

$$w_i \geq 0 \text{ and } \sum_{i=1}^{n} w_i = 1$$

$$\widehat{\mathrm{Z}} = \begin{matrix} -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 \\ -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 \\ -1 & -1 & -1 & -1 & +1 & +1 & +1 & +1 \end{matrix}$$

- For above k instances, labeled by one of the 2^k columns, only consistent weight vector is unit identifying that column
- With constraints all 2^k units can be obtained
- Weight space can has rank 2^k
- With linear combinations of k rows at most rank many units (i.e. k) can be expressed

Additional Constraints - Part 2

$$\begin{array}{cccccccc} -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 \\ -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 \\ -1 & -1 & -1 & -1 & +1 & +1 & +1 & +1 \end{array}$$

- The above k rows appear as rows in the $2^k \times 2^k$ Hadamard
- Therefore any linear combination of the k rows of the sub matrix is distance at least $\sqrt{1 - \frac{k}{2^k}}$ from each of the 2^k unit vectors

- Every linear combination has average square loss at least $1 - \frac{k}{2^k}$ on the full Hadamard matrix
- You need the additional constraints to bring up the span?
- Constraints and consistency = unique solution

Maintain additional constraints?

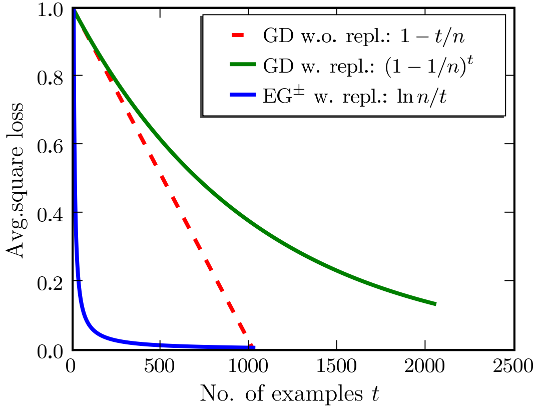
Use Exponentiated Gradient Algorithm [KW97]

Kernel methods $w_i = \sum_{t=1}^{k} \widehat{Z}_{t,i} a_t$

EG $w_i = \exp^{\sum_{t=1}^{k} \widehat{Z}_{t,i} a_t} / const$

Now log weights linear combination of expanded instances

Average Squared Error



EG	Kernel algs
$\frac{\ln(n)}{t}$	$1 - \frac{t}{n}$ and $(1 - \frac{1}{n})^t$

How does EG realize units?

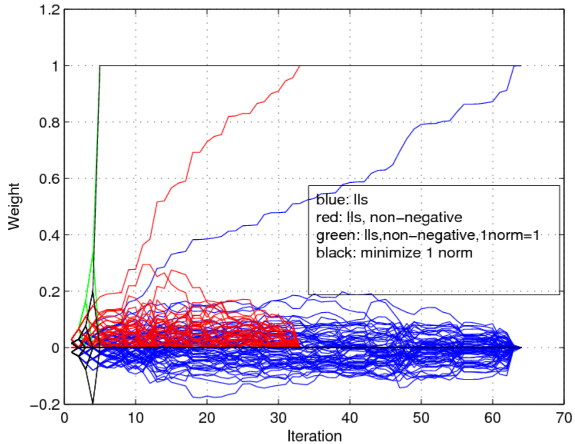
$$\widehat{\mathrm{Z}} = \begin{array}{cccccccc} -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 \\ -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 \\ -1 & -1 & -1 & -1 & +1 & +1 & +1 & +1 \end{array}$$

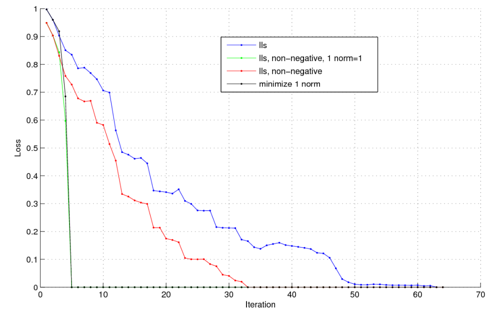
EG $w_i = \exp^{\sum_{t=1}^{k} \widehat{Z}_{t,i} a_t} / const$

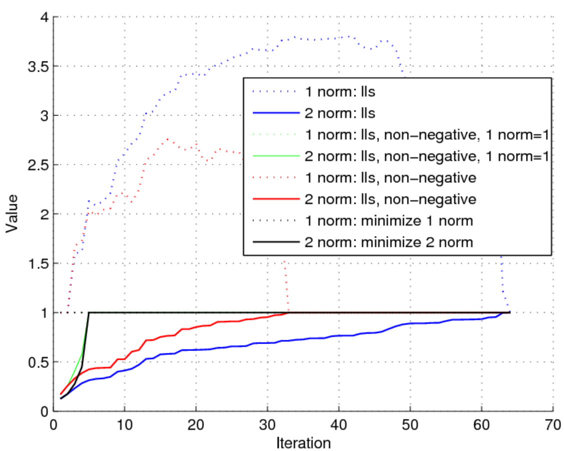
Set coefficient $a_t = \pm\eta$ and let η go to infinity

Each sign pattern corresponds to a different column

What constraints?







Good algs for sparsity?

- EG with loss loss$(\mathbf{w} \cdot \mathbf{x}_t, y_t)$ Santa Cruz way
- GD with loss loss$(\mathbf{w} \cdot \mathbf{x}_t, y_t)$ + sparsity regularizer such as $|\,\mathbf{w}\,|_1$ or entropy of $\sum_i -w_i \log w_i - w_i$

  What neural net community does
- Open:
  - Can above handle worst case example sequences
  - Regret bounds?

For the random bit matrix case

[DPH]

- Consistency + minimizing $|\,\mathbf{w}\,|_1$
  puts all weight on consistent components
- Minimizing $\sum_i (\mathbf{w}\cdot\mathbf{x}_i - y_i)^2 + \eta|\,\mathbf{w}\,|_1$ as $\eta \to 0$
  puts all weight on consistent components
- Minimizing $\sum_i |\,\mathbf{w}\cdot\mathbf{x}_i - y_i|_1 + \eta|\,\mathbf{w}\,|_1$
  puts all weight on consistent components.
  Is $\eta \to 0$ required?

Optimization versus ML

Problem: Noise-free linear regression
I.e. solve a system of linear equations

Optimization: any solution is good

- Time, space, accuracy

Machine Learning:

- How well does solution generalize

Incorporating side info

Kernel algorithms: none

$$\geq 1 - k/n$$

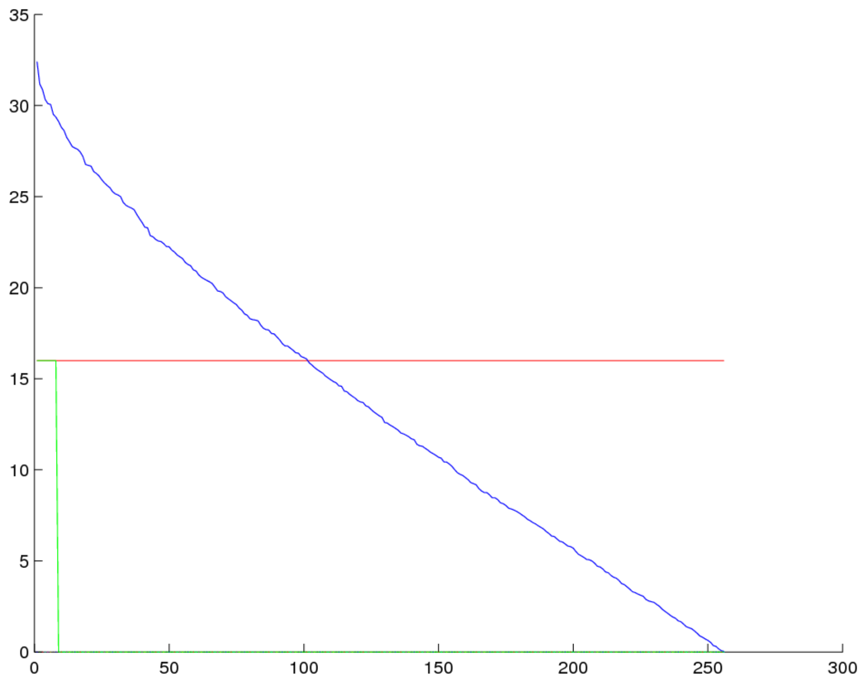
EG: $w_i \geq 0$ and $\sum_i w_i = 1$

$$O(\frac{\log n}{k})$$

$$\begin{array}{rcrrrr}
 & \rightarrow & +1.001 & +1.002 & +1.003 & +1.004 \\
instances & \rightarrow & +1.001 & -1.002 & +1.003 & -1.004 \\
 & \rightarrow & +1.001 & +1.002 & -1.003 & -1.004 \\
 & \rightarrow & +1.001 & -1.002 & -1.003 & +1.004 \\
 & & \uparrow & \uparrow & \uparrow & \uparrow \\
 & & & & targets &
\end{array}$$

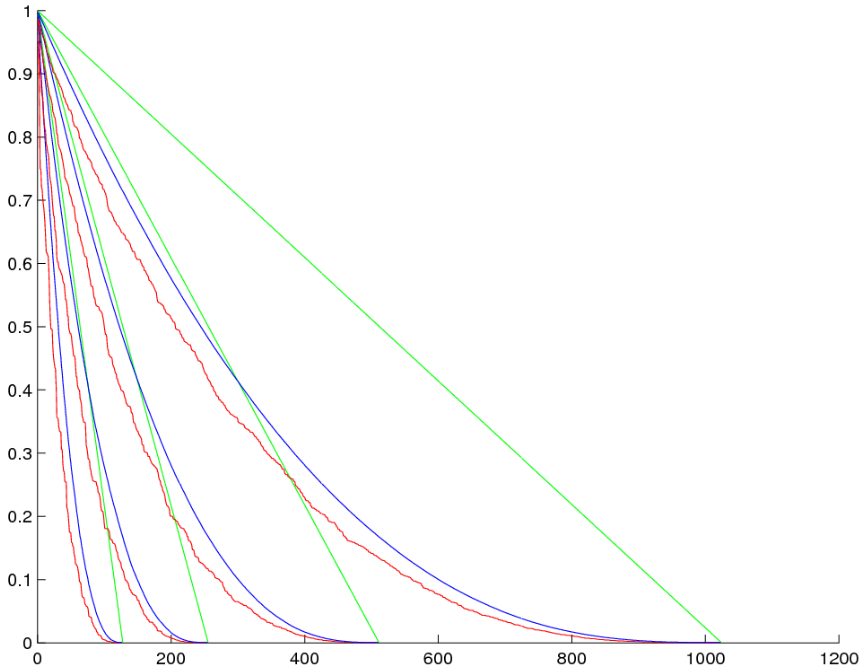
Now target determined by any single example
Trivial algorithm beats EG

Making it worse



—Spectrum of $n \times \log n$ matrix - all $2^{\log n}$ sign patterns
—Spectrum of $n \times n$ matrix produces by expanding the $\log n$ features to all $2^{\log n}$ products
—Adding $n - \log n$ random features instead

Random features cost



—LLS error w.r.t. any single feature in Hadamard matrix
—Average error w.r.t. all single features in random matrix
—Minimum error w.r.t. all single features
$O(1)$ examples needed per random feature

Which matrix?

- If eigen-spectrum of kernel matrix has heavy tail
  then kernel not useful
  - Picked wrong kernel
  - Problem too hard
- If svd-spectrum of problem matrix has heavy tail
  then problem not learnable

Kernel matrix	dot products of instances
Problem matrix	instances as rows - targets as columns

We showed:

- Hadamard problem matrix has heavy tail
- Adding random features makes tail of kernel matrix heavy

Questions?

Gave problem that cannot be learned well by kernel algs

- Similar bounds for classification ?
- Linear neurons with sigmoided output ?
- What is the optimal kernel for a given problem ?
- Simpler generalization bounds for probabilistic settings ?
- Feature selection
- Is there a similar story for learning matrix parameters
- Your questions :-)