Introduction to Machine Learning (67577)
Lecture 1

Shai Shalev-Shwartz

School of CS and Engineering,
The Hebrew University of Jerusalem

Online Learning

Outline

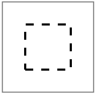
1. The Online Learning Framework
   - Need Prior Knowledge
   - Hypothesis class
2. Learning Finite Hypothesis Classes
   - The Consistent learner
   - The Halving learner
3. Structure over the hypothesis class
   - Halfspaces
   - The Ellipsoid Learner

Game Board

- Domain set, $\mathcal{X}$: This is the set of objects that we may wish to label.
- Label set, $\mathcal{Y}$: The set of possible labels.
- A prediction rule, $h : \mathcal{X} \to \mathcal{Y}$: used to label future examples. This function is called a *predictor*, a *hypothesis*, or a *classifier*.

Example

- $\mathcal{X} = \mathbb{R}^2$ representing color and shape of papayas.
- $\mathcal{Y} = \{\pm 1\}$ representing "tasty" or "non-tasty".
- $h(x) = 1$ if x is within the inner rectangle



The Online Game

For $t = 1, 2, \ldots$

- Environment presents an instance $x_t \in \mathcal{X}$
- Learner predicts label $\hat{y}_t \in \mathcal{Y}$
- Environment reveals true label $y_t \in \mathcal{Y}$
- Learner pays 1 if $\hat{y}_t \neq y_t$ and 0 otherwise

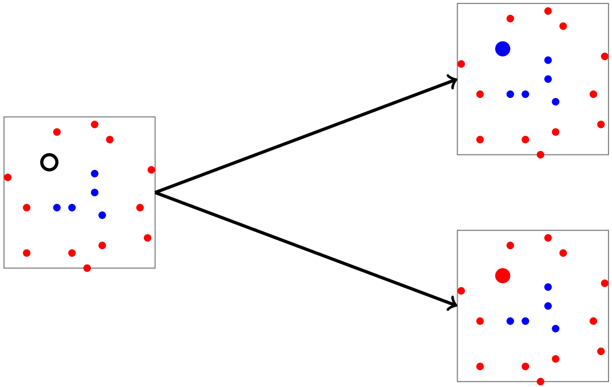
The Online Game

For $t = 1, 2, \ldots$

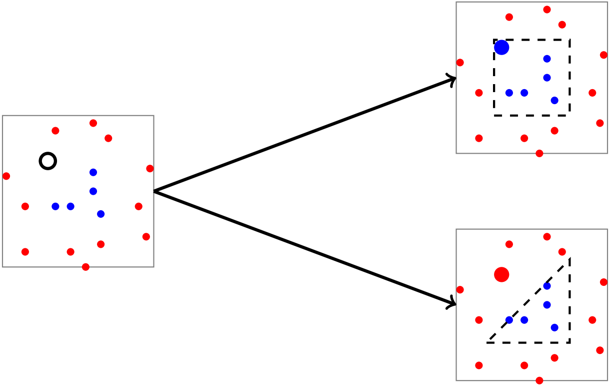
- Environment presents an instance $x_t \in \mathcal{X}$
- Learner predicts label $\hat{y}_t \in \mathcal{Y}$
- Environment reveals true label $y_t \in \mathcal{Y}$
- Learner pays 1 if $\hat{y}_t \neq y_t$ and 0 otherwise

Goal of the learner: Make few mistakes

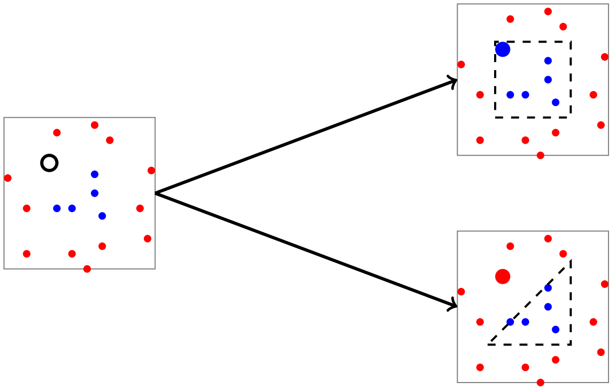
Mission impossible ?



Mission impossible ?



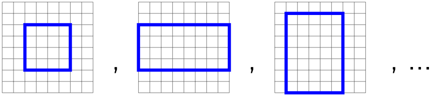
Mission impossible ?



- If $|\mathcal{X}| = \infty$ and on each day environment shows a new x_t, then the learner can't know its label and might always err
- If $|\mathcal{X}| < \infty$, the learner can memorize all labels, but this doesn't feel like learning ...

Prior Knowledge

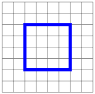
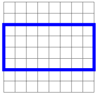
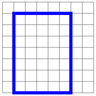
- Give more knowledge to the learner:
- The environment produces labels by applying a target f that comes from some hypothesis class, $\mathcal{H} \subset \mathcal{Y}^{\mathcal{X}}$.
  That is, $\mathcal{H}$ is a pre-defined set of classifiers
- E.g. $\mathcal{H}$ is the set of all axis-aligned rectangles over some grid



- The learner knows $\mathcal{H}$ (but of course doesn't know f)
- How should we learn ?

Prior Knowledge

- Give more knowledge to the learner:
- The environment produces labels by applying a target f that comes from some hypothesis class, $\mathcal{H} \subset \mathcal{Y}^{\mathcal{X}}$.
  That is, $\mathcal{H}$ is a pre-defined set of classifiers
- E.g. $\mathcal{H}$ is the set of all axis-aligned rectangles over some grid

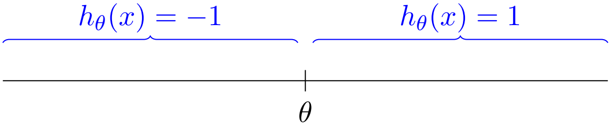
 ,  ,  , ...

- The learner knows $\mathcal{H}$ (but of course doesn't know f)
- How should we learn ?

Remark: What if our prior knowledge is wrong ?
We'll get back to this question later

Not always helps

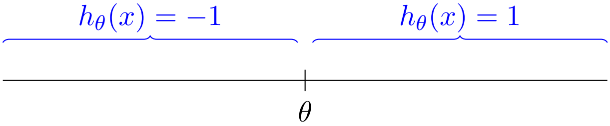
- Let $\mathcal{X} = \mathbb{R}$, and $\mathcal{H}$ be thresholds:
- $\mathcal{H} = \{h_\theta : \theta \in \mathbb{R}\}$, where $h_\theta(x) = \mathrm{sign}(x - \theta)$



- Theorem: for every learner, exists sequence of examples which is consistent with some $f \in \mathcal{H}$ but on which the learner will always err
- Proof idea: environment will follow the bisection method

Not always helps

- Let $\mathcal{X} = \mathbb{R}$, and $\mathcal{H}$ be thresholds:
- $\mathcal{H} = \{h_\theta : \theta \in \mathbb{R}\}$, where $h_\theta(x) = \mathrm{sign}(x - \theta)$



- Theorem: for every learner, exists sequence of examples which is consistent with some $f \in \mathcal{H}$ but on which the learner will always err
- Proof idea: environment will follow the bisection method
- Exercise: show that it's impossible to learn the class of axis-aligned rectangles over the reals

Outline

Learning Finite Classes

- Assume that $\mathcal{H}$ is of finite size
  - E.g.: $\mathcal{H}$ is all the functions from $\mathcal{X}$ to $\mathcal{Y}$ that can be implemented using a Python program of length at most b
  - E.g.: $\mathcal{H}$ is thresholds over a grid $\mathcal{X} = \{0, \frac{1}{n}, \frac{2}{n}, \ldots, 1\}$

Learning Finite Classes

The consistent learner

- Initialize $V_1 = \mathcal{H}$
- For $t = 1, 2, \ldots$
  - Get x_t
  - Pick some $h \in V_t$ and predict $\hat{y}_t = h(x_t)$
  - Get y_t and update $V_{t+1} = \{h \in V_t : h(x_t) = y_t\}$

Analysis

Theorem

The consistent learner will make at most $|\mathcal{H}| - 1$ *mistakes*

Analysis

Theorem

The consistent learner will make at most $|\mathcal{H}| - 1$ *mistakes*

Proof.

If we err at round t, then the $h \in V_t$ we used for prediction will not be in V_{t+1}. Therefore, $|V_{t+1}| \leq |V_t| - 1$. □

Analysis

Theorem

The consistent learner will make at most $|\mathcal{H}| - 1$ mistakes

Proof.

If we err at round t, then the $h \in V_t$ we used for prediction will not be in V_{t+1}. Therefore, $|V_{t+1}| \le |V_t| - 1$. $\square$

Can we do better ?

The Halving learner

The Halving learner

- Initialize $V_1 = \mathcal{H}$
- For $t = 1, 2, \ldots$
  - Get x_t
  - Predict $\mathrm{Majority}(h(x_t) : h \in V_t)$
  - Get y_t and update $V_{t+1} = \{h \in V_t : h(x_t) = y_t\}$

Analysis

Theorem

The Halving learner will make at most $\log_2(|\mathcal{H}|)$ *mistakes*

Analysis

Theorem

The Halving learner will make at most $\log_2(|\mathcal{H}|)$ *mistakes*

Proof.

If we err at round t, then at least half of the functions in V_t will not be in V_{t+1}. Therefore, $|V_{t+1}| \leq |V_t|/2$. □

Analysis

Theorem

The Halving learner will make at most $\log_2(|\mathcal{H}|)$ mistakes

Proof.

If we err at round t, then at least half of the functions in V_t will not be in V_{t+1}. Therefore, $|V_{t+1}| \leq |V_t|/2$. $\square$

Corollary

The Halving learner can learn the class $\mathcal{H}$ of all python programs of length $< b$ bits while making at most b mistakes.

Powerful, but ...

1. What if the environment is not consistent with any $f \in \mathcal{H}$?
   - We'll deal with this later

Powerful, but ...

1. What if the environment is not consistent with any $f \in \mathcal{H}$?
   - We'll deal with this later
2. While the mistake bound of Halving grows with $\log_2(|\mathcal{H}|)$, the runtime of Halving grows with $|\mathcal{H}|$
   - This is the main reason why the course doesn't end now ...
   - Learning must take computational considerations into account

Outline

Efficient learning with structured $\mathcal{H}$

Example:

- Recall again the class $\mathcal{H}$ of thresholds over a grid $\mathcal{X} = \{0, \frac{1}{n}, \ldots, 1\}$ for some integer $n \gg 1$
- Halving mistake bound is $\log(n+1)$
- A naive implementation of Halving takes $\Omega(n)$ time
- How to implement Halving efficiently?

Efficient learning with structured $\mathcal{H}$

Efficient Halving for discrete thresholds

- Initialize $l_1 = -0.5/n, r_1 = 1 + 0.5/n$
- For $t = 1, 2, \ldots$
  - Get $x_t \in \{0, \frac{1}{n}, \ldots, 1\}$
  - Predict $\mathrm{sign}((x_t - l_t) - (r_t - x_t))$
  - Get y_t and if $x_t \in [l_t, r_t]$ update:
    - if $y_t = 1$ then $l_{t+1} = l_t, r_{t+1} = x_t - 0.5/n$
    - if $y_t = -1$ then $l_{t+1} = x_t + 0.5/n, r_{t+1} = r_t$

Efficient learning with structured $\mathcal{H}$

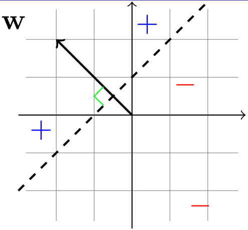
Efficient Halving for discrete thresholds

- Initialize $l_1 = -0.5/n, r_1 = 1 + 0.5/n$
- For $t = 1, 2, \ldots$
  - Get $x_t \in \{0, \frac{1}{n}, \ldots, 1\}$
  - Predict $\mathrm{sign}((x_t - l_t) - (r_t - x_t))$
  - Get y_t and if $x_t \in [l_t, r_t]$ update:
    - if $y_t = 1$ then $l_{t+1} = l_t, r_{t+1} = x_t - 0.5/n$
    - if $y_t = -1$ then $l_{t+1} = x_t + 0.5/n, r_{t+1} = r_t$

- Exercise: show that the above is indeed an implementation of Halving and that the runtime of each iteration is $O(\log(n))$

Outline

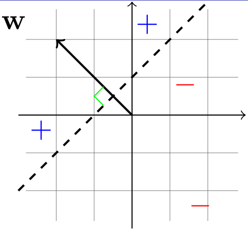
Halfspaces



$$\mathcal{H} = \{\mathbf{x} \mapsto \text{sign}(\langle \mathbf{w}, \mathbf{x} \rangle + b) : \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}\}$$

- Inner product: $\langle \mathbf{w}, \mathbf{x} \rangle = \mathbf{w}^\top \mathbf{x} = \sum_{i=1}^d w_i x_i$
- $\mathbf{w}$ is called a *weight vector* and b a *bias*

Halfspaces



$$\mathcal{H} = \{\mathbf{x} \mapsto \mathrm{sign}(\langle \mathbf{w}, \mathbf{x} \rangle + b) : \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}\}$$

- Inner product: $\langle \mathbf{w}, \mathbf{x} \rangle = \mathbf{w}^\top \mathbf{x} = \sum_{i=1}^d w_i x_i$
- $\mathbf{w}$ is called a *weight vector* and b a *bias*
- For $d = 1$, the class of Halfspaces is the class of thresholds
- W.l.o.g., assume that $x_d = 1$ for all examples, and then we can treat w_d as the bias and forget about b

Using halving to learn halfspaces on a grid

- Let us represent all numbers on the grid $G = \{-1, -1 + 1/n, \ldots, 1 - 1/n, 1\}$
- Then, $|\mathcal{H}| = |G|^d = (2n + 1)^d$
- Therefore, Halving's bound is at most $d \log(2n + 1)$
- We will show an algorithm with a slightly worse mistake bound but that can be implemented efficiently

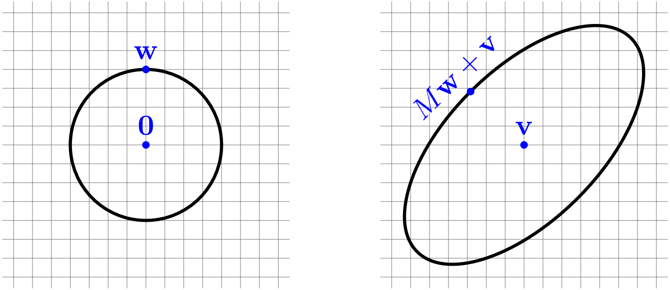
The Ellipsoid Learner

- Recall that Halving maintains the "Version Space", V_t, containing all hypotheses in $\mathcal{H}$ which are consistent with the examples observed so far
- Each halfspace hypothesis corresponds to a vector in G^d
- Instead of maintaining V_t, we will maintain an ellipsoid, $\mathcal{E}_t$, that contains V_t
- We will show that every time we make a mistake the volume of $\mathcal{E}_t$ shrinks by a factor of $e^{-1/(2n+2)}$
- On the other hand, we will show that the volume of $\mathcal{E}_t$ cannot be made too small (this is where we use the grid assumption)

Background: Balls and Ellipsoids

- Let $B = \{\mathbf{w} \in \mathbb{R}^d : \|\mathbf{w}\|^2 \le 1\}$ be the unit ball of $\mathbb{R}^d$
- Recall: $\|\mathbf{w}\|^2 = \langle \mathbf{w}, \mathbf{w} \rangle = \mathbf{w}^\top \mathbf{w} = \sum_{i=1}^d w_i^2$
- An ellipsoid is the image of a ball under an affine mapping: given a matrix M and a vector $\mathbf{v}$,

$$\mathcal{E}(M, \mathbf{v}) = \{M\mathbf{w} + \mathbf{v} : \|\mathbf{w}\|^2 \le 1\}$$



The Ellipsoid Learner

- We implicitly maintain an ellipsoid: $\mathcal{E}_t = \mathcal{E}(A_t^{1/2}, \mathbf{w}_t)$
- Start with $\mathbf{w}_1 = \mathbf{0}$, $A_1 = I$
- For $t = 1, 2, \ldots$
  - Get $\mathbf{x}_t$
  - Predict $\hat{y}_t = \mathrm{sign}(\mathbf{w}_t^\top \mathbf{x}_t)$
  - Get y_t
  - If $\hat{y}_t \neq y_t$ update:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \frac{y_t}{d+1} \frac{A_t \mathbf{x}_t}{\sqrt{\mathbf{x}_t^\top A_t \mathbf{x}_t}}$$

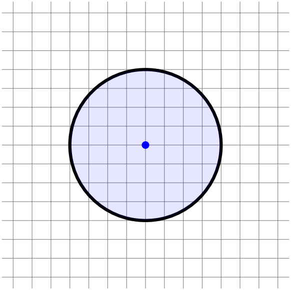
$$A_{t+1} = \frac{d^2}{d^2-1}\left(A_t - \frac{2}{d+1}\frac{A_t \mathbf{x}_t \mathbf{x}_t^\top A_t}{\mathbf{x}_t^\top A_t \mathbf{x}_t}\right)$$

  - If $\hat{y}_t = y_t$ keep $\mathbf{w}_{t+1} = \mathbf{w}_t$ and $A_{t+1} = A_t$

Python code

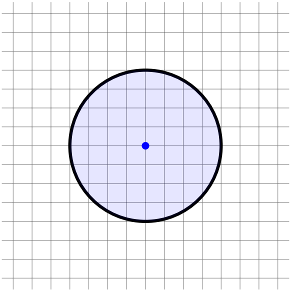
```
w = zeros((d,));
A = eye(d);
M = 0; # counts mistakes
eta = d*d/(d*d-1.0);
for t in range(0,n):
    yhat = sign(dot(w,X[:,t]));
    if Y[t] != yhat:
        M = M+1;
        Ax = dot(A , X[:,t]);
        xAx = dot(X[:,t] , Ax);
        w = w + Y[t]/((d+1)*sqrt(xAx)) * Ax;
        A = eta*( A - (2.0/((d+1.0)*xAx)) * outer(Ax,Ax) );
```

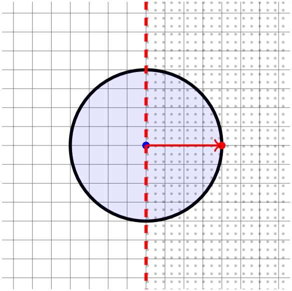
Intuition



Intuition

Suppose $\mathbf{x}_1 = (1,0)^\top, y_1 = 1$.

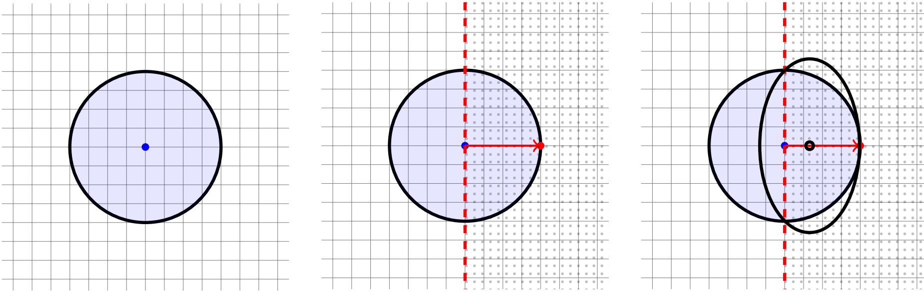




Intuition

Suppose $\mathbf{x}_1 = (1,0)^\top, y_1 = 1$. Then:

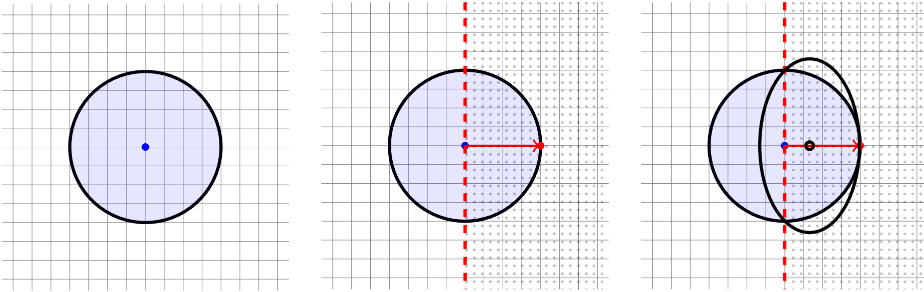
$$\mathbf{w}_2 = \begin{pmatrix} 1/3 \\ 0 \end{pmatrix} \quad , \quad A_2 = \begin{pmatrix} 4/3 & 0 \\ 0 & 4/9 \end{pmatrix}$$



Intuition

Suppose $\mathbf{x}_1 = (1,0)^\top, y_1 = 1$. Then:

$$\mathbf{w}_2 = \begin{pmatrix} 1/3 \\ 0 \end{pmatrix} \quad , \quad A_2 = \begin{pmatrix} 4/3 & 0 \\ 0 & 4/9 \end{pmatrix}$$



- $\mathcal{E}_2$ is Ellipsoid of minimum volume that contains $\mathcal{E}_1 \cap \{\mathbf{w} : y_1 \langle \mathbf{w}, \mathbf{x}_1 \rangle > 0\}$

Analysis

Theorem

The Ellipsoid learner makes at most $2d(2d+2)\log(n)$ *mistakes.*

Analysis

Theorem

The Ellipsoid learner makes at most $2d(2d+2)\log(n)$ *mistakes.*

Proof is based on two lemmas:

Lemma (Volume Reduction)

Whenever we make a mistake, $\mathrm{Vol}(\mathcal{E}_{t+1}) \le \mathrm{Vol}(\mathcal{E}_t)\, e^{-\frac{1}{2d+2}}$.

Lemma (Volume can't be too small)

For every t, $\mathrm{Vol}(\mathcal{E}_t) \ge \mathrm{Vol}(B)\,(1/n)^{2d}$

- Therefore, after M mistakes:

$$\mathrm{Vol}(B)\,(1/n)^{2d} \le \mathrm{Vol}(\mathcal{E}_t) \le \mathrm{Vol}(B)\, e^{-M\,\frac{1}{2d+2}}$$

Summary

- The Online Learning model
- Need prior knowledge
- Learning finite hypothesis classes using Halving
- The runtime problem
- The Ellipsoid efficiently learns halfspaces (over a grid)

Summary

- The Online Learning model
- Need prior knowledge
- Learning finite hypothesis classes using Halving
- The runtime problem
- The Ellipsoid efficiently learns halfspaces (over a grid)

- How did we derive the update equations?
- How to prove the lemmas?
- You need math for this !
- details in the next slides

Background: Balls and Ellipsoids

- Recall: $\mathcal{E}(M, \mathbf{v}) = \{M\mathbf{w} + \mathbf{v} : \|\mathbf{w}\|^2 \le 1\}$
- We deal with non-degenerative ellipsoids, i.e., M is invertible
- SVD theorem: Every real invertible matrix M can be decomposed as $M = UDV^\top$ where U, V orthonormal and D diagonal with $D_{i,i} > 0$.
- Exercise: Show that $\mathcal{E}(M, \mathbf{v}) = \mathcal{E}(UD, \mathbf{v}) = \mathcal{E}(UDU^\top, \mathbf{v})$
- Therefore, we can assume w.l.o.g. that $M = UDU^\top$ (i.e., it is symmetric positive definite)
- Exercise: Show that for such M

$$\mathcal{E}(M, \mathbf{v}) = \{\mathbf{x} : (\mathbf{x} - \mathbf{v})^\top M^{-2} (\mathbf{x} - \mathbf{v}) \le 1\}$$

where $M^{-2} = UD^{-2}U^\top$ with $(D^{-2})_{i,i} = D_{i,i}^{-2}$

Volume Calculations

- Let $\mathrm{Vol}(B)$ be the volume of the unit ball
- Lemma: If $M = UDU^\top$ is positive definite, then

$$\mathrm{Vol}(\mathcal{E}(M, \mathbf{v})) = \det(M)\mathrm{Vol}(B) = \left(\prod_{i=1}^{m} D_{i,i}\right)\mathrm{Vol}(B)$$

Why volume shrinks

- Suppose $A_t = UD^2U^\top$. Define $\tilde{\mathbf{x}}_t = DU^\top \mathbf{x}_t$. Then:

$$A_{t+1} = \frac{d^2}{d^2-1}\left(A_t - \frac{2}{d+1}\frac{A_t\mathbf{x}_t\mathbf{x}_t^\top A_t}{\mathbf{x}_t^\top A_t \mathbf{x}_t}\right)$$
$$= \frac{d^2}{d^2-1} UD\left(I - \frac{2}{d+1}\frac{\tilde{\mathbf{x}}_t\tilde{\mathbf{x}}_t^\top}{\|\tilde{\mathbf{x}}_t\|^2}\right)DU^\top$$

- By Sylvester's determinant theorem, $\det(I + \mathbf{u}\mathbf{v}^\top) = 1 + \langle \mathbf{u}, \mathbf{v}\rangle$. Therefore,

$$\det(A_{t+1}) = \left(\frac{d^2}{d^2-1}\right)^d \det(D)\det\left(I - \frac{2}{d+1}\frac{\tilde{\mathbf{x}}_t\tilde{\mathbf{x}}_t^\top}{\|\tilde{\mathbf{x}}_t\|^2}\right)\det(D)$$
$$= \det(A_t)\left(\frac{d^2}{d^2-1}\right)^d\left(1 - \frac{2}{d+1}\right)$$

Why volume shrinks

We obtain:

$$
\begin{aligned}
\frac{\mathrm{Vol}(\mathcal{E}_{t+1})}{\mathrm{Vol}(\mathcal{E}_t)} &= \left(\frac{d^2}{d^2-1}\right)^{d/2}\left(1-\frac{2}{d+1}\right)^{1/2} \\
&= \left(\frac{d^2}{d^2-1}\right)^{\frac{d-1}{2}} \cdot \frac{d}{\sqrt{(d-1)(d+1)}} \cdot \frac{\sqrt{d-1}}{\sqrt{d+1}} \\
&= \left(1+\frac{1}{d^2-1}\right)^{\frac{d-1}{2}} \cdot \left(1-\frac{1}{d+1}\right) \\
&\le e^{\frac{d-1}{2(d^2-1)}} \cdot e^{-\frac{1}{d+1}} = e^{-\frac{1}{2(d+1)}}
\end{aligned}
$$

where we used $1+a \le e^a$ which holds for all $a \in \mathbb{R}$.

Why volume can't be too small

- Recall, $y_t\langle \mathbf{w}^\star, \mathbf{x}_t\rangle > 0$ for every t.
- Since $\mathbf{w}^\star, \mathbf{x}_t$ are on the grid G, it follows that $y_t\langle \mathbf{w}^\star, \mathbf{x}_t\rangle \ge 1/n^2$.
- Therefore, if $\|\mathbf{w} - \mathbf{w}^\star\| < 1/n^2$ then

$$y_t\langle \mathbf{w}, \mathbf{x}_t\rangle = y_t\langle \mathbf{w}-\mathbf{w}^\star, \mathbf{x}_t\rangle + y_t\langle \mathbf{w}^\star, \mathbf{x}_t\rangle \ge -\|\mathbf{w}-\mathbf{w}^\star\|\|\mathbf{x}_t\| + 1/n^2 > 0$$

- Convince yourself (by induction) that $\mathcal{E}_t$ contains the ball of radius $1/n^2$ centered around $\mathbf{w}^\star$. It follows that

$$\mathrm{Vol}(B)\,(1/n^2)^d = \mathrm{Vol}(\mathcal{E}(\tfrac{1}{n^2}I, \mathbf{w}^\star)) \le \mathrm{Vol}(\mathcal{E}_t)$$