Introduction to Online Learning Algorithms

Yoav Freund

December 27, 2024

Outline

Example trace for Halving Algorithm

Example trace for Halving Algorithm

expert1
expert2
expert3
expert4
expert5
expert6
expert7
expert8

alg.

Example trace for Halving Algorithm

expert1
expert2
expert3
expert4
expert5
expert6
expert7
expert8

alg.
outcome

Example trace for Halving Algorithm

	$t = 1$
expert1	1
expert2	1
expert3	0
expert4	1
expert5	1
expert6	0
expert7	1
expert8	1
alg.	
outcome	

Example trace for Halving Algorithm

	$t = 1$
expert1	1
expert2	1
expert3	0
expert4	1
expert5	1
expert6	0
expert7	1
expert8	1
alg.	1
outcome	

Example trace for Halving Algorithm

	$t = 1$
expert1	1
expert2	1
expert3	0
expert4	1
expert5	1
expert6	0
expert7	1
expert8	1
alg.	1
outcome	1

Example trace for Halving Algorithm

	$t = 1$	$t = 2$
expert1	1	1
expert2	1	0
expert3	0	-
expert4	1	0
expert5	1	0
expert6	0	-
expert7	1	1
expert8	1	1
alg.	1	
outcome	1	

Example trace for Halving Algorithm

	$t = 1$	$t = 2$
expert1	1	1
expert2	1	0
expert3	0	-
expert4	1	0
expert5	1	0
expert6	0	-
expert7	1	1
expert8	1	1
alg.	1	0
outcome	1	

Example trace for Halving Algorithm

	$t = 1$	$t = 2$
expert1	1	1
expert2	1	0
expert3	0	-
expert4	1	0
expert5	1	0
expert6	0	-
expert7	1	1
expert8	1	1
alg.	1	0
outcome	1	1

Example trace for Halving Algorithm

	$t = 1$	$t = 2$	$t = 3$
expert1	1	1	1
expert2	1	0	-
expert3	0	-	-
expert4	1	0	-
expert5	1	0	-
expert6	0	-	-
expert7	1	1	1
expert8	1	1	1
alg.	1	0	
outcome	1	1	

Example trace for Halving Algorithm

	$t = 1$	$t = 2$	$t = 3$
expert1	1	1	1
expert2	1	0	-
expert3	0	-	-
expert4	1	0	-
expert5	1	0	-
expert6	0	-	-
expert7	1	1	1
expert8	1	1	1
alg.	1	0	1
outcome	1	1	

Example trace for Halving Algorithm

	$t = 1$	$t = 2$	$t = 3$
expert1	1	1	1
expert2	1	0	-
expert3	0	-	-
expert4	1	0	-
expert5	1	0	-
expert6	0	-	-
expert7	1	1	1
expert8	1	1	1
alg.	1	0	1
outcome	1	1	1

Example trace for Halving Algorithm

	$t = 1$	$t = 2$	$t = 3$	$t = 4$
expert1	1	1	1	1
expert2	1	0	-	-
expert3	0	-	-	-
expert4	1	0	-	-
expert5	1	0	-	-
expert6	0	-	-	-
expert7	1	1	1	1
expert8	1	1	1	0
alg.	1	0	1	
outcome	1	1	1	

Example trace for Halving Algorithm

	$t = 1$	$t = 2$	$t = 3$	$t = 4$
expert1	1	1	1	1
expert2	1	0	-	-
expert3	0	-	-	-
expert4	1	0	-	-
expert5	1	0	-	-
expert6	0	-	-	-
expert7	1	1	1	1
expert8	1	1	1	0
alg.	1	0	1	1
outcome	1	1	1	

Example trace for Halving Algorithm

	$t = 1$	$t = 2$	$t = 3$	$t = 4$
expert1	1	1	1	1
expert2	1	0	-	-
expert3	0	-	-	-
expert4	1	0	-	-
expert5	1	0	-	-
expert6	0	-	-	-
expert7	1	1	1	1
expert8	1	1	1	0
alg.	1	0	1	1
outcome	1	1	1	0

Example trace for Halving Algorithm

	$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$
expert1	1	1	1	1	-
expert2	1	0	-	-	-
expert3	0	-	-	-	-
expert4	1	0	-	-	-
expert5	1	0	-	-	-
expert6	0	-	-	-	-
expert7	1	1	1	1	0
expert8	1	1	1	0	-
alg.	1	0	1	1	
outcome	1	1	1	0	

Example trace for Halving Algorithm

	$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$
expert1	1	1	1	1	-
expert2	1	0	-	-	-
expert3	0	-	-	-	-
expert4	1	0	-	-	-
expert5	1	0	-	-	-
expert6	0	-	-	-	-
expert7	1	1	1	1	0
expert8	1	1	1	0	-
alg.	1	0	1	1	0
outcome	1	1	1	0	

Example trace for Halving Algorithm

	$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$
expert1	1	1	1	1	-
expert2	1	0	-	-	-
expert3	0	-	-	-	-
expert4	1	0	-	-	-
expert5	1	0	-	-	-
expert6	0	-	-	-	-
expert7	1	1	1	1	0
expert8	1	1	1	0	-
alg.	1	0	1	1	0
outcome	1	1	1	0	0

Mistake bound for Halving algorithm

- Each time algorithm makes a mistakes, the pool of perfect experts is halved (at least).

Mistake bound for Halving algorithm

- Each time algorithm makes a mistakes, the pool of perfect experts is halved (at least).
- We assume that at least one expert is perfect.

Mistake bound for Halving algorithm

- Each time algorithm makes a mistakes, the pool of perfect experts is halved (at least).
- We assume that at least one expert is perfect.
- Number of mistakes is at most $\log_2 N$.

Mistake bound for Halving algorithm

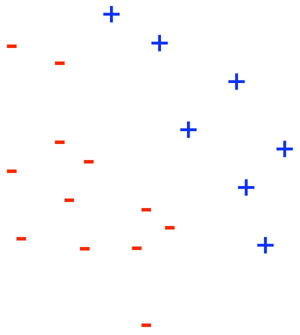
- Each time algorithm makes a mistakes, the pool of perfect experts is halved (at least).
- We assume that at least one expert is perfect.
- Number of mistakes is at most $\log_2 N$.
- No stochastic assumptions whatsoever.

Mistake bound for Halving algorithm

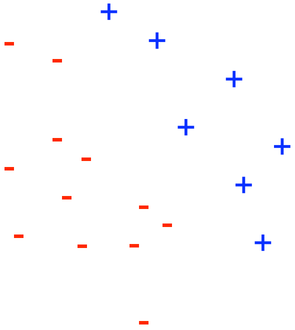
- Each time algorithm makes a mistakes, the pool of perfect experts is halved (at least).
- We assume that at least one expert is perfect.
- Number of mistakes is at most $\log_2 N$.
- No stochastic assumptions whatsoever.
- Proof is based on combining a lower and upper bounds on the number of perfect experts.

The Perceptron Problem

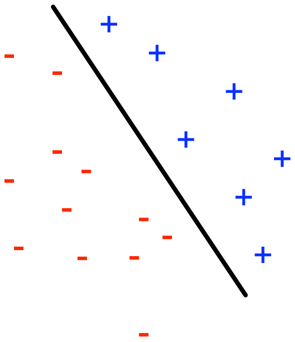
The Perceptron Problem



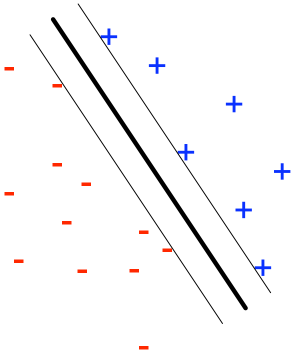
The Perceptron Problem



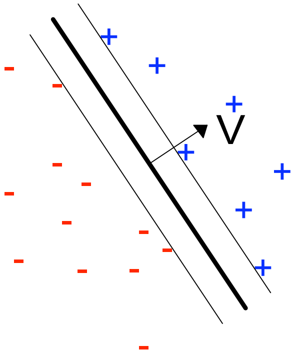
The Perceptron Problem



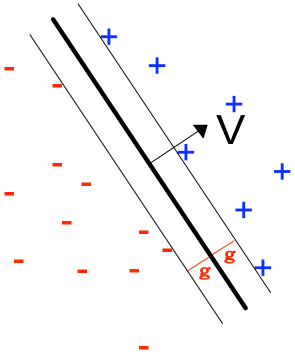
The Perceptron Problem



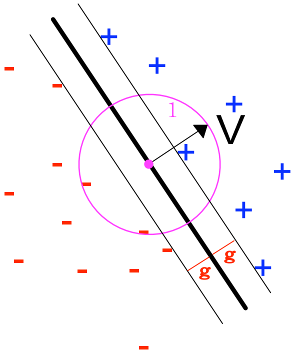
The Perceptron Problem



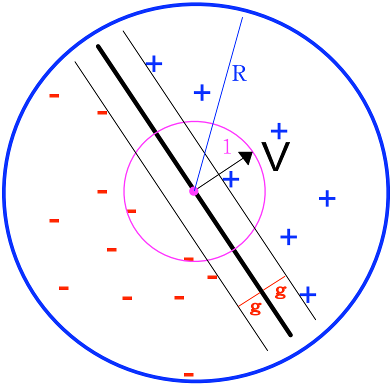
The Perceptron Problem



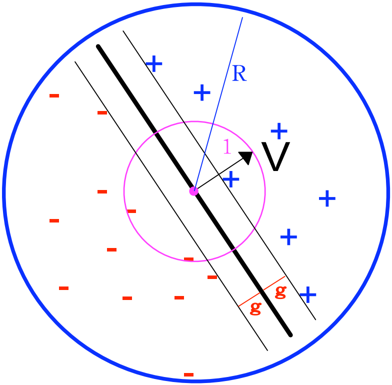
The Perceptron Problem



The Perceptron Problem

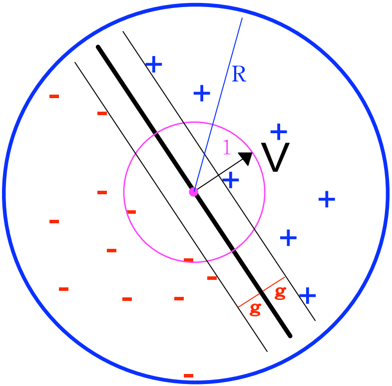


The Perceptron Problem



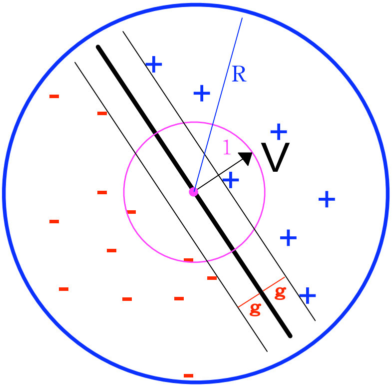
- $\|\vec{V}\| = 1$

The Perceptron Problem



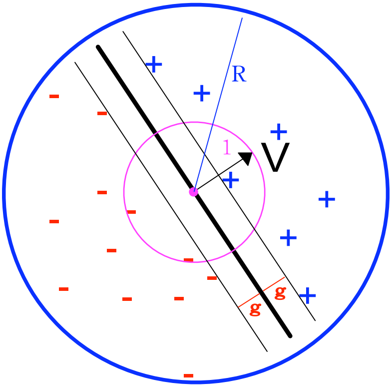
- $\|\vec{V}\| = 1$
- Example = $(\vec{X}, y)$, $y \in \{-1, +1\}$.

The Perceptron Problem



- $\|\vec{V}\| = 1$
- Example = $(\vec{X}, y)$, $y \in \{-1, +1\}$.
- $\forall \vec{X},\ \ \|\vec{X}\| \leq R$.

The Perceptron Problem



- $\|\vec{V}\| = 1$
- Example = $(\vec{X}, y)$, $y \in \{-1, +1\}$.
- $\forall \vec{X},\ \ \|\vec{X}\| \leq R$.
- $\forall (\vec{X}, y)$, $y(\vec{X} \cdot \vec{V}) \geq g$

The Perceptron learning algorithm

- An online algorithm. Examples presented one by one.

The Perceptron learning algorithm

- An online algorithm. Examples presented one by one.
- start with $\vec{W}_0 = \vec{0}$.

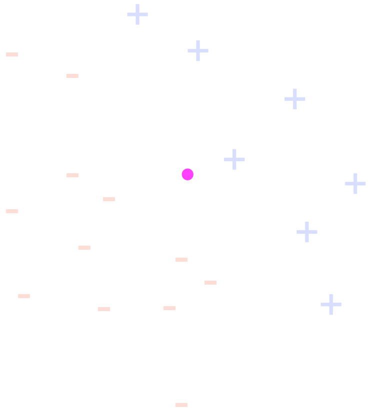
The Perceptron learning algorithm

- An online algorithm. Examples presented one by one.
- start with $\vec{W}_0 = \vec{0}$.
- If mistake: $(\vec{W}_i \cdot \vec{X}_i) y_i \leq 0$

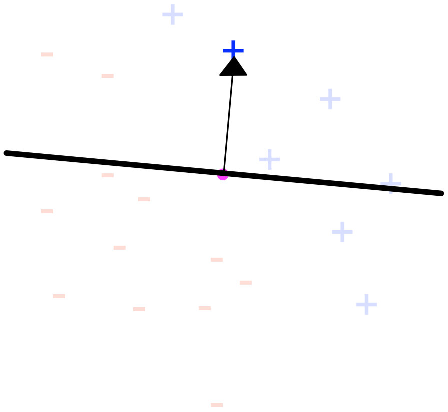
The Perceptron learning algorithm

- An online algorithm. Examples presented one by one.
- start with $\vec{W}_0 = \vec{0}$.
- If mistake: $(\vec{W}_i \cdot \vec{X}_i)y_i \leq 0$
    - Update $\vec{W}_{i+1} = \vec{W}_i + y_i X_i$.

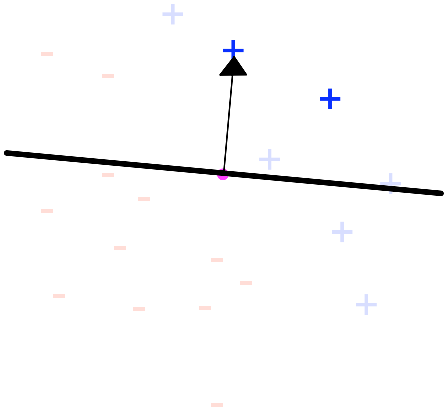
Example trace for the perceptron algorithm



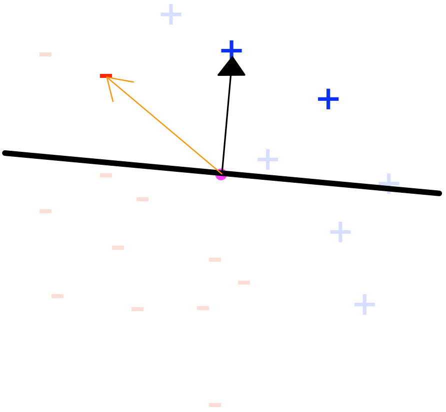
Example trace for the perceptron algorithm



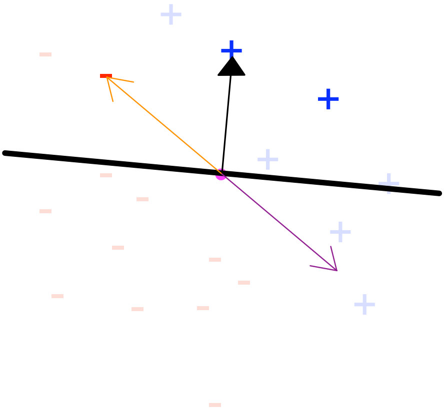
Example trace for the perceptron algorithm



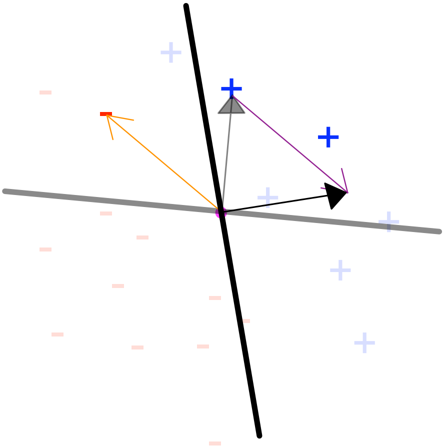
Example trace for the perceptron algorithm



Example trace for the perceptron algorithm



Example trace for the perceptron algorithm



Bound on number of mistakes

- The number of mistakes that the perceptron algorithm can make is at most $\left(\frac{R}{g}\right)^2$.

Bound on number of mistakes

- The number of mistakes that the perceptron algorithm can make is at most $\left(\frac{R}{g}\right)^2$.
- Proof by combining upper and lower bounds on $\|\vec{W}\|$.

Pythagorian Lemma

If $(\vec{W}_i \cdot X_i)y < 0$ then

Pythagorian Lemma

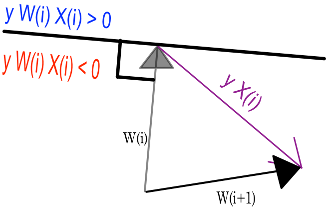
If $(\vec{W}_i \cdot X_i)y < 0$ then

$$\|\vec{W}_{i+1}\|^2 = \|\vec{W}_i + y_i\vec{X}_i\|^2 \leq \|\vec{W}_i\|^2 + \|\vec{X}_i\|^2$$

Pythagorian Lemma

If $(\vec{W}_i \cdot X_i)y < 0$ then

$$\|\vec{W}_{i+1}\|^2 = \|\vec{W}_i + y_i\vec{X}_i\|^2 \leq \|\vec{W}_i\|^2 + \|\vec{X}_i\|^2$$



Upper bound on $\|\vec{W}_i\|$

Upper bound on $\|\vec{W}_i\|$

Proof by induction

- Claim: $\|\vec{W}_i\|^2 \leq iR^2$

Upper bound on $\|\vec{W}_i\|$

Proof by induction

- Claim: $\|\vec{W}_i\|^2 \leq iR^2$
- Base: $i = 0$, $\|\vec{W}_0\|^2 = 0$

Upper bound on $\|\vec{W}_i\|$

Proof by induction

- Claim: $\|\vec{W}_i\|^2 \leq iR^2$
- Base: $i = 0$, $\|\vec{W}_0\|^2 = 0$
- Induction step (assume for i and prove for $i + 1$):
  $\|\vec{W}_{i+1}\|^2 \leq \|\vec{W}_i\|^2 + \|\vec{X}_i\|^2$
  $\leq \|\vec{W}_i\|^2 + R^2 \leq (i + 1)R^2$

Lower bound on $\|\vec{W}_i\|$

Lower bound on $\|\vec{W}_i\|$

$\|\vec{W}_i\| \geq \vec{W}_i \cdot \vec{V}$ because $\|\vec{V}\| = 1$.

Lower bound on $\|\vec{W}_i\|$

$\|\vec{W}_i\| \geq \vec{W}_i \cdot \vec{V}$ because $\|\vec{V}\| = 1$.
We prove a lower bound on $\vec{W}_i \cdot \vec{V}$ using induction over i

- Claim: $\vec{W}_i \cdot \vec{V} \geq ig$

Lower bound on $\|\vec{W}_i\|$

$\|\vec{W}_i\| \geq \vec{W}_i \cdot \vec{V}$ because $\|\vec{V}\| = 1$.
We prove a lower bound on $\vec{W}_i \cdot \vec{V}$ using induction over i

- Claim: $\vec{W}_i \cdot \vec{V} \geq ig$
- Base: $i = 0$, $\vec{W}_0 \cdot \vec{V} = 0$

Lower bound on $\|\vec{W}_i\|$

$\|\vec{W}_i\| \geq \vec{W}_i \cdot \vec{V}$ because $\|\vec{V}\| = 1$.
We prove a lower bound on $\vec{W}_i \cdot \vec{V}$ using induction over i

- Claim: $\vec{W}_i \cdot \vec{V} \geq ig$
- Base: $i = 0$, $\vec{W}_0 \cdot \vec{V} = 0$
- Induction step (assume for i and prove for $i + 1$):
  $\vec{W}_{i+1} \cdot \vec{V} = \left(\vec{W}_i + \vec{X}_i y_i\right) \vec{V}$

Lower bound on $\|\vec{W}_i\|$

$\|\vec{W}_i\| \geq \vec{W}_i \cdot \vec{V}$ because $\|\vec{V}\| = 1$.
We prove a lower bound on $\vec{W}_i \cdot \vec{V}$ using induction over i

- Claim: $\vec{W}_i \cdot \vec{V} \geq ig$
- Base: $i = 0$, $\vec{W}_0 \cdot \vec{V} = 0$
- Induction step (assume for i and prove for $i + 1$):
  $\vec{W}_{i+1} \cdot \vec{V} = \left(\vec{W}_i + \vec{X}_i y_i\right) \vec{V}$

Lower bound on $\|\vec{W}_i\|$

$\|\vec{W}_i\| \geq \vec{W}_i \cdot \vec{V}$ because $\|\vec{V}\| = 1$.
We prove a lower bound on $\vec{W}_i \cdot \vec{V}$ using induction over i

- Claim: $\vec{W}_i \cdot \vec{V} \geq ig$
- Base: $i = 0$, $\vec{W}_0 \cdot \vec{V} = 0$
- Induction step (assume for i and prove for $i + 1$):
  $\vec{W}_{i+1} \cdot \vec{V} = \left(\vec{W}_i + \vec{X}_i y_i\right) \vec{V} = \vec{W}_i \cdot \vec{V} + y_i \vec{X}_i \cdot \vec{V}$
  $\geq ig + g = (i + 1)g$

Combining the upper and lower bounds

Combining the upper and lower bounds

$$(ig)^2 \leq \|\vec{W}_i\|^2 \leq iR^2$$

Combining the upper and lower bounds

$$(ig)^2 \leq \|\vec{W}_i\|^2 \leq iR^2$$

Thus:

$$i \leq \left(\frac{R}{g}\right)^2$$

The mean estimation game

- An adversary choses a real number $y_t in [0, 1]$ and keeps it secret.

The mean estimation game

- An adversary choses a real number $y_t in [0, 1]$ and keeps it secret.
- You make a guess of the secret number x_t

The mean estimation game

- An adversary choses a real number $y_t in [0,1]$ and keeps it secret.
- You make a guess of the secret number x_t
- The adversary reveals the secret and you pay $(x_t - y_t)^2$

The mean estimation game

- An adversary choses a real number $y_t in [0, 1]$ and keeps it secret.
- You make a guess of the secret number x_t
- The adversary reveals the secret and you pay $(x_t - y_t)^2$
- You want to minimize $\frac{1}{T} \sum_{t=1}^{T} (x_t - y_t)^2$

The mean estimation game

- An adversary choses a real number $y_t in [0, 1]$ and keeps it secret.
- You make a guess of the secret number x_t
- The adversary reveals the secret and you pay $(x_t - y_t)^2$
- You want to minimize $\frac{1}{T} \sum_{t=1}^{T} (x_t - y_t)^2$
- Impossible without additional constraints.

Adversary is a fixed distribution

- Suppose that the adversary draws $y_1, y_2, \ldots, y_T$ IID from a fixed distribution over $[0, 1]$ with mean μ and std σ.

Adversary is a fixed distribution

- Suppose that the adversary draws $y_1, y_2, \ldots, y_T$ IID from a fixed distribution over $[0, 1]$ with mean μ and std σ.
- Optimal prediction $x_t = \mu$

Adversary is a fixed distribution

- Suppose that the adversary draws $y_1, y_2, \ldots, y_T$ IID from a fixed distribution over $[0, 1]$ with mean μ and std σ.
- Optimal prediction $x_t = \mu$
- $\mathrm{E}_Y\left[(\mu - Y)^2\right] = \sigma^2$

Adversary is a fixed distribution

- Suppose that the adversary draws $y_1, y_2, \ldots, y_T$ IID from a fixed distribution over $[0, 1]$ with mean μ and std σ.
- Optimal prediction $x_t = \mu$
- $\mathrm{E}_Y \left[(\mu - Y)^2 \right] = \sigma^2$
- Online prediction: predict x_{t+1} from $Y^t = \langle Y_1, Y_2, \ldots, Y_t \rangle$.

Adversary is a fixed distribution

- Suppose that the adversary draws $y_1, y_2, \ldots, y_T$ IID from a fixed distribution over $[0, 1]$ with mean μ and std σ.
- Optimal prediction $x_t = \mu$
- $\mathrm{E}_Y \left[(\mu - Y)^2 \right] = \sigma^2$
- Online prediction: predict x_{t+1} from $Y^t = \langle Y_1, Y_2, \ldots, Y_t \rangle$.
- **Expected regret**: compare performance of algorithm to Regret $= \mathrm{E}_{Y^T} \left[(x_t - Y_t)^2 \right] - \sigma^2$

Individual sequence bounds

- Make no assumption about how the sequence is generated.

Individual sequence bounds

- Make no assumption about how the sequence is generated.
- The best constant value for x in hind-sight:

$$x_T^* \doteq \operatorname*{argmin}_{x \in [0,1]} \sum_{t=1}^{T} (x - y_t)^2, \quad x_t^* = \frac{1}{T} \sum_{t=1}^{T} X_t$$

Individual sequence bounds

- Make no assumption about how the sequence is generated.
- The best constant value for x in hind-sight:

$$x_T^* \doteq \operatorname*{argmin}_{x \in [0,1]} \sum_{t=1}^{T} (x - y_t)^2, \quad x_t^* = \frac{1}{T} \sum_{t=1}^{T} X_t$$

- Regret: the loss over and above the loss of x_T^*. **for the worst-case sequence**
  $\text{Regret}_T = \sum_{t=1}^{T} (x_t - y_t)^2 - \sum_{t=1}^{T} (x_T^* - y_t)^2$

Individual sequence bounds

- Make no assumption about how the sequence is generated.
- The best constant value for x in hind-sight:

$$x_T^* \doteq \operatorname*{argmin}_{x \in [0,1]} \sum_{t=1}^{T} (x - y_t)^2, \quad x_t^* = \frac{1}{T} \sum_{t=1}^{T} X_t$$

- Regret: the loss over and above the loss of x_T^*. **for the worst-case sequence**
  $\text{Regret}_T = \sum_{t=1}^{T} (x_t - y_t)^2 - \sum_{t=1}^{T} (x_T^* - y_t)^2$
- **Goal:** sublinear regret $\lim_{T \to \infty} \frac{\text{Regret}_T}{T} = 0$

Follow the Leader

- Idea: set x_{t+1} to be the best constant prediction on $y_1, \ldots, y_t$

Follow the Leader

- Idea: set x_{t+1} to be the best constant prediction on $y_1, \ldots, y_t$
- $x_{t+1} = \operatorname{argmin}_{x \in [0,1]} \sum_{i=1}^{t} (x - y_i)^2$

Follow the Leader

- Idea: set x_{t+1} to be the best constant prediction on $y_1, \ldots, y_t$
- $x_{t+1} = \operatorname{argmin}_{x \in [0,1]} \sum_{i=1}^{t} (x - y_i)^2$
- We will prove that the regret of this algorithm is upper bound by $4 + 4 \ln T$

A more general setup

- General euclidean space: $\mathbf{x}, \mathbf{y}$ are elements in $V \subset \mathbf{R}^d$

A more general setup

- General euclidean space: $\mathbf{x}, \mathbf{y}$ are elements in $V \subset \mathbf{R}^d$
- The loss function for time step t maps $\mathbf{x}$ to $\mathbf{R}$:
  $\ell_t : V \to \mathbf{R}$

A more general setup

- General euclidean space: $\mathbf{x}, \mathbf{y}$ are elements in $V \subset \mathbf{R}^d$
- The loss function for time step t maps $\mathbf{x}$ to $\mathbf{R}$: $\ell_t : V \to \mathbf{R}$
- For square loss: $\ell_t(\mathbf{x}) = (\mathbf{x} - \mathbf{y}_t)^2$

A more general setup

- General euclidean space: $\mathbf{x}, \mathbf{y}$ are elements in $V \subset \mathbf{R}^d$
- The loss function for time step t maps $\mathbf{x}$ to $\mathbf{R}$: $\ell_t : V \to \mathbf{R}$
- For square loss: $\ell_t(\mathbf{x}) = (\mathbf{x} - \mathbf{y}_t)^2$
- Regret relative to $\mathbf{u} \in V$: $\text{Regret}_T = \sum_{t=1}^{T} \ell_t(\mathbf{x}_t) - \sum_{t=1}^{T} \ell_t(\mathbf{u})$

Technical Lemma

Lemma

Let $\mathbf{x}_t^*$ be the minimizer of $\sum_{i=1}^{t} \ell_i(\mathbf{x})$. Then
$\sum_{t=1}^{T} \ell_t(\mathbf{x}_t^*) \leq \sum_{t=1}^{T} \ell_t\left(\mathbf{x}_T^*\right)$

regret bound

Theorem

Let $y_t \in [0, 1]$ *for* $t = 1, \ldots T$ *an arbitrary sequence of numbers. Let the algorithm output be* $x_t = x^*_{t-1} = \frac{1}{t-1}\sum_{i=1}^{t-1} y_i$*, then*

$$Regret_T = \sum_{t=1}^{T}(x_t - y_t)^2 - \sum_{t=1}^{T}(x^*_T - y_t)^2 \leq 4 + 4 \ln T$$

regret bound

Theorem

Let $y_t \in [0,1]$ *for* $t = 1, \ldots T$ *an arbitrary sequence of numbers. Let the algorithm output be* $x_t = x^*_{t-1} = \frac{1}{t-1}\sum_{i=1}^{t-1} y_i$*, then*

$$Regret_T = \sum_{t=1}^{T}(x_t - y_t)^2 - \sum_{t=1}^{T}(x^*_T - y_t)^2 \leq 4 + 4\ln T$$

- HW for next monday: prove the theorem.