Online Learning

Piyush Rai

Machine Learning (CS771A)

Nov 9, 2016

Batch vs Online Learning

Batch vs Online Learning

- The Batch Learning setting
  - Train using some training data, apply it on some test data

Batch vs Online Learning

- The Batch Learning setting
  - Train using some training data, apply it on some test data
  - Training and test distribution assumed to be the same

Batch vs Online Learning

- The Batch Learning setting
  - Train using some training data, apply it on some test data
  - Training and test distribution assumed to be the same
  - Performance measured in terms of generalization error (difference between training and test error)

Batch vs Online Learning

- The Batch Learning setting
  - Train using some training data, apply it on some test data
  - Training and test distribution assumed to be the same
  - Performance measured in terms of generalization error (difference between training and test error)
- The Online Learning setting
  - No separate training phase. All the data is test data

Batch vs Online Learning

- The Batch Learning setting
  - Train using some training data, apply it on some test data
  - Training and test distribution assumed to be the same
  - Performance measured in terms of generalization error (difference between training and test error)
- The Online Learning setting
  - No separate training phase. All the data is test data
  - Data arrives sequentially. Predictions are made one at a time

Batch vs Online Learning

- The Batch Learning setting
  - Train using some training data, apply it on some test data
  - Training and test distribution assumed to be the same
  - Performance measured in terms of generalization error (difference between training and test error)
- The Online Learning setting
  - No separate training phase. All the data is test data
  - Data arrives sequentially. Predictions are made one at a time
  - Data need not come from a fixed distribution. Can be chosen adversarially

Batch vs Online Learning

- The Batch Learning setting
  - Train using some training data, apply it on some test data
  - Training and test distribution assumed to be the same
  - Performance measured in terms of generalization error (difference between training and test error)
- The Online Learning setting
  - No separate training phase. All the data is test data
  - Data arrives sequentially. Predictions are made one at a time
  - Data need not come from a fixed distribution. Can be chosen adversarially
  - The goal is to minimize the total mistakes for the sequence

Batch vs Online Learning

- The Batch Learning setting
  - Train using some training data, apply it on some test data
  - Training and test distribution assumed to be the same
  - Performance measured in terms of generalization error (difference between training and test error)
- The Online Learning setting
  - No separate training phase. All the data is test data
  - Data arrives sequentially. Predictions are made one at a time
  - Data need not come from a fixed distribution. Can be chosen adversarially
  - The goal is to minimize the total mistakes for the sequence
- Note: As we have seen previously, batch models can be trained in an online fashion (e.g., using SGD or the Perceptron algorithm)

Learning with Expert Advice

Learning with Expert Advice

Suppose we want to learn a model that predicts a sequence of bits

Learning with Expert Advice

Suppose we want to learn a model that predicts a sequence of bits

Assume we also have D "experts", each of whom can make a prediction (0/1)

Learning with Expert Advice

Suppose we want to learn a model that predicts a sequence of bits

Assume we also have D "experts", each of whom can make a prediction (0/1)

- For round $t = 1, 2, \ldots, T$

Learning with Expert Advice

Suppose we want to learn a model that predicts a sequence of bits

Assume we also have D "experts", each of whom can make a prediction (0/1)

- For round $t = 1, 2, \ldots, T$
  - Each expert d predicts $\xi_{d,t} \in \{0, 1\}$

Learning with Expert Advice

Suppose we want to learn a model that predicts a sequence of bits

Assume we also have D "experts", each of whom can make a prediction (0/1)

- For round $t = 1, 2, \ldots, T$
  - Each expert d predicts $\xi_{d,t} \in \{0, 1\}$
  - The learner "combines" experts' predictions and predicts $\hat{y}_t \in \{0, 1\}$

Learning with Expert Advice

Suppose we want to learn a model that predicts a sequence of bits

Assume we also have D "experts", each of whom can make a prediction (0/1)

- For round $t = 1, 2, \ldots, T$
  - Each expert d predicts $\xi_{d,t} \in \{0, 1\}$
  - The learner "combines" experts' predictions and predicts $\hat{y}_t \in \{0, 1\}$
  - The true $y_t \in \{0, 1\}$ is revealed

Learning with Expert Advice

Suppose we want to learn a model that predicts a sequence of bits

Assume we also have D "experts", each of whom can make a prediction (0/1)

- For round $t = 1, 2, \ldots, T$
  - Each expert d predicts $\xi_{d,t} \in \{0, 1\}$
  - The learner "combines" experts' predictions and predicts $\hat{y}_t \in \{0, 1\}$
  - The true $y_t \in \{0, 1\}$ is revealed
  - The learner suffers a loss $\ell(y_t, \hat{y}_t) = \mathbb{1}[y_t \neq \hat{y}_t]$

Learning with Expert Advice

Suppose we want to learn a model that predicts a sequence of bits

Assume we also have D "experts", each of whom can make a prediction (0/1)

- For round $t = 1, 2, \ldots, T$
  - Each expert d predicts $\xi_{d,t} \in \{0, 1\}$
  - The learner "combines" experts' predictions and predicts $\hat{y}_t \in \{0, 1\}$
  - The true $y_t \in \{0, 1\}$ is revealed
  - The learner suffers a loss $\ell(y_t, \hat{y}_t) = \mathbb{1}[y_t \neq \hat{y}_t]$
- Our goal is to minimize the number of mistakes made by the learner, s.t.

$$\boxed{\underbrace{\sum_{t=1}^{T} \ell(y_t, \hat{y}_t)}_{\#\text{ mistakes by the learner}} \leq \underbrace{\min_{d} \sum_{t=1}^{T} \ell(y_t, \xi_{d,t})}_{\#\text{ mistakes by the best expert}} + (\text{a number called "regret"})}$$

Learning with Expert Advice

Suppose we want to learn a model that predicts a sequence of bits

Assume we also have D "experts", each of whom can make a prediction (0/1)

- For round $t = 1, 2, \ldots, T$
  - Each expert d predicts $\xi_{d,t} \in \{0, 1\}$
  - The learner "combines" experts' predictions and predicts $\hat{y}_t \in \{0, 1\}$
  - The true $y_t \in \{0, 1\}$ is revealed
  - The learner suffers a loss $\ell(y_t, \hat{y}_t) = \mathbb{1}[y_t \neq \hat{y}_t]$
- Our goal is to minimize the number of mistakes made by the learner, s.t.

$$\boxed{\underbrace{\sum_{t=1}^{T} \ell(y_t, \hat{y}_t)}_{\#\text{ mistakes by the learner}} \leq \underbrace{\min_d \sum_{t=1}^{T} \ell(y_t, \xi_{d,t})}_{\#\text{ mistakes by the best expert}} + \text{(a number called "regret")}}$$

- Also, don't want the learner to be too much worse than the best expert in hindsight (the difference known as "regret").

Learning with Expert Advice

Suppose we want to learn a model that predicts a sequence of bits

Assume we also have D "experts", each of whom can make a prediction (0/1)

- For round $t = 1, 2, \ldots, T$
  - Each expert d predicts $\xi_{d,t} \in \{0, 1\}$
  - The learner "combines" experts' predictions and predicts $\hat{y}_t \in \{0, 1\}$
  - The true $y_t \in \{0, 1\}$ is revealed
  - The learner suffers a loss $\ell(y_t, \hat{y}_t) = \mathbb{1}[y_t \neq \hat{y}_t]$
- Our goal is to minimize the number of mistakes made by the learner, s.t.

$$\boxed{\underbrace{\sum_{t=1}^{T} \ell(y_t, \hat{y}_t)}_{\#\text{ mistakes by the learner}} \leq \underbrace{\min_d \sum_{t=1}^{T} \ell(y_t, \xi_{d,t})}_{\#\text{ mistakes by the best expert}} + (\text{a number called "regret"})}$$

- Also, don't want the learner to be too much worse than the best expert in hindsight (the difference known as "regret"). Also want $\frac{\text{regret}}{T} \to 0$ as $T \to \infty$

An Example

- Consider predicting stock behavior on each day (will move up or down?)

	Experts 1	Experts 2	Experts 3	Experts 4	Learner (Master)	Outcome
day 1	$\uparrow$	$\uparrow$	$\downarrow$	$\uparrow$	$\uparrow$	$\uparrow$
day 2	$\downarrow$	$\uparrow$	$\uparrow$	$\downarrow$	$\downarrow$	$\uparrow$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
# of mistakes	37	12	67	50	18	

A Naïve Algorithm

Assume at least one of the D experts is perfect and always gives the right advice (of course, the learner doesn't know who that is)

A Naïve Algorithm

Assume at least one of the D experts is perfect and always gives the right advice (of course, the learner doesn't know who that is)

- For round $t = 1, 2, \ldots, T$

A Naïve Algorithm

Assume at least one of the D experts is perfect and always gives the right advice (of course, the learner doesn't know who that is)

- For round $t = 1, 2, \ldots, T$
  - Pick an expert d randomly and predict $\hat{y}_t = \xi_{d,t}$

A Naïve Algorithm

Assume at least one of the D experts is perfect and always gives the right advice (of course, the learner doesn't know who that is)

- For round $t = 1, 2, \ldots, T$
  - Pick an expert d randomly and predict $\hat{y}_t = \xi_{d,t}$
  - The true $y_t \in \{0, 1\}$ is revealed

A Naïve Algorithm

Assume at least one of the D experts is perfect and always gives the right advice (of course, the learner doesn't know who that is)

- For round $t = 1, 2, \ldots, T$
  - Pick an expert d randomly and predict $\hat{y}_t = \xi_{d,t}$
  - The true $y_t \in \{0, 1\}$ is revealed
  - If $y_t \neq \hat{y}_t$, remove expert d from the list of experts

A Naïve Algorithm

Assume at least one of the D experts is perfect and always gives the right advice (of course, the learner doesn't know who that is)

- For round $t = 1, 2, \ldots, T$
  - Pick an expert d randomly and predict $\hat{y}_t = \xi_{d,t}$
  - The true $y_t \in \{0, 1\}$ is revealed
  - If $y_t \neq \hat{y}_t$, remove expert d from the list of experts
- The learner makes at most $D - 1$ mistakes

A Naïve Algorithm

Assume at least one of the D experts is perfect and always gives the right advice (of course, the learner doesn't know who that is)

- For round $t = 1, 2, \ldots, T$
  - Pick an expert d randomly and predict $\hat{y}_t = \xi_{d,t}$
  - The true $y_t \in \{0, 1\}$ is revealed
  - If $y_t \neq \hat{y}_t$, remove expert d from the list of experts
- The learner makes at most $D - 1$ mistakes
- This is a rather pessimistic bound. Can we do better?

Halving Algorithm

Same Setting, i.e., assume at least one of the D experts is perfect

Halving Algorithm

Same Setting, i.e., assume at least one of the D experts is perfect

- For round $t = 1, 2, \ldots, T$

Halving Algorithm

Same Setting, i.e., assume at least one of the D experts is perfect

- For round $t = 1, 2, \ldots, T$
  - $\#$ of experts predicting 1: $n_1 = \sum_n \mathbb{1}[\xi_{d,t} = 1]$

Halving Algorithm

Same Setting, i.e., assume at least one of the D experts is perfect

- For round $t = 1, 2, \ldots, T$
  - # of experts predicting 1: $n_1 = \sum_n \mathbb{1}[\xi_{d,t} = 1]$
  - # of experts predicting 0: $n_0 = \sum_n \mathbb{1}[\xi_{d,t} = 0]$

Halving Algorithm

Same Setting, i.e., assume at least one of the D experts is perfect

- For round $t = 1, 2, \ldots, T$
  - # of experts predicting 1: $n_1 = \sum_n \mathbb{1}[\xi_{d,t} = 1]$
  - # of experts predicting 0: $n_0 = \sum_n \mathbb{1}[\xi_{d,t} = 0]$
  - Learner predicts based on what the majority says, i.e., $\hat{y}_t = \mathbb{1}[n_1 > n_0]$

Halving Algorithm

Same Setting, i.e., assume at least one of the D experts is perfect

- For round $t = 1, 2, \ldots, T$
  - $\#$ of experts predicting 1: $n_1 = \sum_n \mathbb{1}[\xi_{d,t} = 1]$
  - $\#$ of experts predicting 0: $n_0 = \sum_n \mathbb{1}[\xi_{d,t} = 0]$
  - Learner predicts based on what the majority says, i.e., $\hat{y}_t = \mathbb{1}[n_1 > n_0]$
  - The true $y_t \in \{0, 1\}$ is revealed

Halving Algorithm

Same Setting, i.e., assume at least one of the D experts is perfect

- For round $t = 1, 2, \ldots, T$
  - # of experts predicting 1: $n_1 = \sum_n \mathbb{1}[\xi_{d,t} = 1]$
  - # of experts predicting 0: $n_0 = \sum_n \mathbb{1}[\xi_{d,t} = 0]$
  - Learner predicts based on what the majority says, i.e., $\hat{y}_t = \mathbb{1}[n_1 > n_0]$
  - The true $y_t \in \{0, 1\}$ is revealed
  - If $y_t \neq \hat{y}_t$, remove experts who predicted incorrectly (this gets rid of at least half of them, hence the name)

Mistake Bound for Halving Algorithm

How many mistakes does the learner make?

Mistake Bound for Halving Algorithm

How many mistakes does the learner make?

- Let W denotes the number of surviving experts at any round

Mistake Bound for Halving Algorithm

How many mistakes does the learner make?

- Let W denotes the number of surviving experts at any round

$$\text{After 1 mistake: } W \leq \frac{1}{2}D$$

Mistake Bound for Halving Algorithm

How many mistakes does the learner make?

- Let W denotes the number of surviving experts at any round

$$\text{After 1 mistake: } W \leq \frac{1}{2}D$$

$$\text{After 2 mistakes: } W \leq \frac{1}{4}D$$

Mistake Bound for Halving Algorithm

How many mistakes does the learner make?

- Let W denotes the number of surviving experts at any round

$$\text{After 1 mistake: } W \leq \frac{1}{2}D$$

$$\text{After 2 mistakes: } W \leq \frac{1}{4}D$$

$$\vdots$$

$$\text{After } M \text{ mistakes: } W \leq 2^{-M}D$$

Mistake Bound for Halving Algorithm

How many mistakes does the learner make?

- Let W denotes the number of surviving experts at any round

$$\text{After 1 mistake: } W \leq \frac{1}{2}D$$

$$\text{After 2 mistakes: } W \leq \frac{1}{4}D$$

$$\vdots$$

$$\text{After } M \text{ mistakes: } W \leq 2^{-M}D$$

- One of the experts is perfect and will never be thrown out, so $W \geq 1$

Mistake Bound for Halving Algorithm

How many mistakes does the learner make?

- Let W denotes the number of surviving experts at any round

$$\text{After 1 mistake: } W \leq \frac{1}{2}D$$
$$\text{After 2 mistakes: } W \leq \frac{1}{4}D$$
$$\vdots$$
$$\text{After } M \text{ mistakes: } W \leq 2^{-M}D$$

- One of the experts is perfect and will never be thrown out, so $W \geq 1$

$$1 \leq W \leq 2^{-M}D$$

Mistake Bound for Halving Algorithm

How many mistakes does the learner make?

- Let W denotes the number of surviving experts at any round

$$\text{After 1 mistake: } W \leq \frac{1}{2}D$$
$$\text{After 2 mistakes: } W \leq \frac{1}{4}D$$
$$\vdots$$
$$\text{After } M \text{ mistakes: } W \leq 2^{-M}D$$

- One of the experts is perfect and will never be thrown out, so $W \geq 1$

$$1 \leq W \leq 2^{-M}D$$

- Therefore the learner makes $M \leq \log_2 D$ mistakes

An Aside: Experts as Hypotheses

- Suppose we have a hypothesis space $\mathcal{H} = \{h_1, \ldots, h_D\}$

An Aside: Experts as Hypotheses

- Suppose we have a hypothesis space $\mathcal{H} = \{h_1, \ldots, h_D\}$
- Assume each hypothesis $h_d : h(\boldsymbol{x}) \rightarrow \{0, 1\}$

An Aside: Experts as Hypotheses

- Suppose we have a hypothesis space $\mathcal{H} = \{h_1, \ldots, h_D\}$
- Assume each hypothesis $h_d : h(\boldsymbol{x}) \rightarrow \{0, 1\}$
- Assume the "true" concept $c \in \mathcal{H}$

An Aside: Experts as Hypotheses

- Suppose we have a hypothesis space $\mathcal{H} = \{h_1, \ldots, h_D\}$
- Assume each hypothesis $h_d : h(\boldsymbol{x}) \rightarrow \{0, 1\}$
- Assume the "true" concept $c \in \mathcal{H}$
- Like the previous case, the halving algorithm will have the following bound

$$M \leq \log_2 D = \log_2 |\mathcal{H}|$$

An Aside: Experts as Hypotheses

- Suppose we have a hypothesis space $\mathcal{H} = \{h_1, \ldots, h_D\}$
- Assume each hypothesis $h_d : h(\boldsymbol{x}) \to \{0, 1\}$
- Assume the "true" concept $c \in \mathcal{H}$
- Like the previous case, the halving algorithm will have the following bound

$$M \leq \log_2 D = \log_2 |\mathcal{H}|$$

- Note that this is related to PAC learning where the generalization error depends on $\log |\mathcal{H}|$

An Aside: Experts as Hypotheses

- Suppose we have a hypothesis space $\mathcal{H} = \{h_1, \ldots, h_D\}$
- Assume each hypothesis $h_d : h(\boldsymbol{x}) \to \{0, 1\}$
- Assume the "true" concept $c \in \mathcal{H}$
- Like the previous case, the halving algorithm will have the following bound

$$M \leq \log_2 D = \log_2 |\mathcal{H}|$$

- Note that this is related to PAC learning where the generalization error depends on $\log |\mathcal{H}|$
- Special case: Experts as features (expert d is the value of feature d)

Weighted Majority Algorithm

- What if no expert is perfect?

Weighted Majority Algorithm

- What if no expert is perfect?
- Let's maintain a weight w_d for each expert d. Initialized to $w_d = 1, \forall d$

Weighted Majority Algorithm

- What if no expert is perfect?
- Let's maintain a weight w_d for each expert d. Initialized to $w_d = 1, \forall d$
- Let $W = \sum_{d=1}^{D} w_d$ (Initially $W = D$), and $\beta \in [0, 1)$

Weighted Majority Algorithm

- What if no expert is perfect?
- Let's maintain a weight w_d for each expert d. Initialized to $w_d = 1, \forall d$
- Let $W = \sum_{d=1}^{D} w_d$ (Initially $W = D$), and $\beta \in [0, 1)$
- For round $t = 1, 2, \ldots, T$

Weighted Majority Algorithm

- What if no expert is perfect?
- Let's maintain a weight w_d for each expert d. Initialized to $w_d = 1, \forall d$
- Let $W = \sum_{d=1}^{D} w_d$ (Initially $W = D$), and $\beta \in [0, 1)$
- For round $t = 1, 2, \ldots, T$
  - Total weight of experts predicting 1: $n_1 = \sum_{d:\xi_{d,t}=1} w_d$

Weighted Majority Algorithm

- What if no expert is perfect?
- Let's maintain a weight w_d for each expert d. Initialized to $w_d = 1, \forall d$
- Let $W = \sum_{d=1}^{D} w_d$ (Initially $W = D$), and $\beta \in [0, 1)$
- For round $t = 1, 2, \ldots, T$
  - Total weight of experts predicting 1: $n_1 = \sum_{d:\xi_{d,t}=1} w_d$
  - Total weight of experts predicting 0: $n_0 = \sum_{d:\xi_{d,t}=0} w_d = W - n_1$

Weighted Majority Algorithm

- What if no expert is perfect?
- Let's maintain a weight w_d for each expert d. Initialized to $w_d = 1, \forall d$
- Let $W = \sum_{d=1}^{D} w_d$ (Initially $W = D$), and $\beta \in [0, 1)$
- For round $t = 1, 2, \ldots, T$
  - Total weight of experts predicting 1: $n_1 = \sum_{d:\xi_{d,t}=1} w_d$
  - Total weight of experts predicting 0: $n_0 = \sum_{d:\xi_{d,t}=0} w_d = W - n_1$
  - Learner predicts as $\hat{y}_t = \mathbb{1}[n_1 > n_0]$, i.e., based on "weighted" majority

Weighted Majority Algorithm

- What if no expert is perfect?
- Let's maintain a weight w_d for each expert d. Initialized to $w_d = 1, \forall d$
- Let $W = \sum_{d=1}^{D} w_d$ (Initially $W = D$), and $\beta \in [0, 1)$
- For round $t = 1, 2, \ldots, T$
  - Total weight of experts predicting 1: $n_1 = \sum_{d:\xi_{d,t}=1} w_d$
  - Total weight of experts predicting 0: $n_0 = \sum_{d:\xi_{d,t}=0} w_d = W - n_1$
  - Learner predicts as $\hat{y}_t = \mathbb{1}[n_1 > n_0]$, i.e., based on "weighted" majority
  - Downweight each expert d who predicted incorrectly

$$w_d = \beta w_d \quad \text{if } \xi_{d,t} \neq y_t$$

Weighted Majority Algorithm

- What if no expert is perfect?
- Let's maintain a weight w_d for each expert d. Initialized to $w_d = 1, \forall d$
- Let $W = \sum_{d=1}^{D} w_d$ (Initially $W = D$), and $\beta \in [0, 1)$
- For round $t = 1, 2, \ldots, T$
  - Total weight of experts predicting 1: $n_1 = \sum_{d:\xi_{d,t}=1} w_d$
  - Total weight of experts predicting 0: $n_0 = \sum_{d:\xi_{d,t}=0} w_d = W - n_1$
  - Learner predicts as $\hat{y}_t = \mathbb{1}[n_1 > n_0]$, i.e., based on "weighted" majority
  - Downweight each expert d who predicted incorrectly

$$w_d = \beta w_d \quad \text{if } \xi_{d,t} \neq y_t$$

- For $\beta = 0$, this is equivalent to the halving algorithm

Mistake Bound for Weighted Majority

- Let $W = \sum_d w_d$

Mistake Bound for Weighted Majority

- Let $W = \sum_d w_d$
- Suppose, at the current round, the true $y = 0$

Mistake Bound for Weighted Majority

- Let $W = \sum_d w_d$
- Suppose, at the current round, the true $y = 0$

$$W_{new} = n_1\beta + n_0$$

Mistake Bound for Weighted Majority

- Let $W = \sum_d w_d$
- Suppose, at the current round, the true $y = 0$

$$\begin{aligned} W_{new} &= n_1\beta + n_0 \\ &= n_1\beta + W - n_1 \end{aligned}$$

Mistake Bound for Weighted Majority

- Let $W = \sum_d w_d$
- Suppose, at the current round, the true $y = 0$

$$\begin{aligned} W_{new} &= n_1\beta + n_0 \\ &= n_1\beta + W - n_1 \\ &= W - (1-\beta)n_1 \end{aligned}$$

Mistake Bound for Weighted Majority

- Let $W = \sum_d w_d$
- Suppose, at the current round, the true $y = 0$

$$
\begin{aligned}
W_{new} &= n_1\beta + n_0 \\
&= n_1\beta + W - n_1 \\
&= W - (1-\beta)n_1
\end{aligned}
$$

- Since the learner made a mistake, we must have $n_1 \geq W/2$

Mistake Bound for Weighted Majority

- Let $W = \sum_d w_d$
- Suppose, at the current round, the true $y = 0$

$$\begin{aligned} W_{new} &= n_1\beta + n_0 \\ &= n_1\beta + W - n_1 \\ &= W - (1-\beta)n_1 \end{aligned}$$

- Since the learner made a mistake, we must have $n_1 \geq W/2$. Therefore

$$W_{new} \leq W - (1-\beta)\frac{1}{2}W = \left(\frac{1+\beta}{2}\right)W$$

Mistake Bound for Weighted Majority

- Let $W = \sum_d w_d$
- Suppose, at the current round, the true $y = 0$

$$\begin{aligned} W_{new} &= n_1\beta + n_0 \\ &= n_1\beta + W - n_1 \\ &= W - (1-\beta)n_1 \end{aligned}$$

- Since the learner made a mistake, we must have $n_1 \geq W/2$. Therefore

$$W_{new} \leq W - (1-\beta)\frac{1}{2}W = \left(\frac{1+\beta}{2}\right)W$$

- Note: Same would be true if $y = 1$ and the learner had made a mistake

Mistake Bound for Weighted Majority

- Let $W = \sum_d w_d$
- Suppose, at the current round, the true $y = 0$

$$\begin{aligned} W_{new} &= n_1\beta + n_0 \\ &= n_1\beta + W - n_1 \\ &= W - (1-\beta)n_1 \end{aligned}$$

- Since the learner made a mistake, we must have $n_1 \geq W/2$. Therefore

$$W_{new} \leq W - (1-\beta)\frac{1}{2}W = \left(\frac{1+\beta}{2}\right)W$$

- Note: Same would be true if $y = 1$ and the learner had made a mistake
- After a total of M mistakes, we will have

$$W_{new} \leq \left(\frac{1+\beta}{2}\right)^M D \quad \text{(since } W = D \text{ initially)}$$

Mistake Bound for Weighted Majority

- Suppose M_d be the # of mistakes of expert d. Then $w_d = \beta^{M_d}$

Mistake Bound for Weighted Majority

- Suppose M_d be the # of mistakes of expert d. Then $w_d = \beta^{M_d}$. Therefore

$$\beta^{M_d} = w_d \leq W \leq \left(\frac{1+\beta}{2}\right)^M D$$

Mistake Bound for Weighted Majority

- Suppose M_d be the # of mistakes of expert d. Then $w_d = \beta^{M_d}$. Therefore

$$\beta^{M_d} = w_d \leq W \leq \left(\frac{1+\beta}{2}\right)^M D$$

- Solve for M gives, for each expert d

$$M \leq \frac{M_d \log \frac{1}{\beta} + \log D}{\log\left(\frac{2}{1+\beta}\right)}$$

Mistake Bound for Weighted Majority

- Suppose M_d be the # of mistakes of expert d. Then $w_d = \beta^{M_d}$. Therefore

$$\beta^{M_d} = w_d \leq W \leq \left(\frac{1+\beta}{2}\right)^M D$$

- Solve for M gives, for each expert d

$$M \leq \frac{M_d \log \frac{1}{\beta} + \log D}{\log\left(\frac{2}{1+\beta}\right)}$$

- This holds for all experts, including the best expert. Thus

$$M \leq \min_d \frac{M_d \log \frac{1}{\beta} + \log D}{\log\left(\frac{2}{1+\beta}\right)} = \min_d [a_\beta M_d + c_\beta \log D]$$

Mistake Bound for Weighted Majority

- Suppose M_d be the # of mistakes of expert d. Then $w_d = \beta^{M_d}$. Therefore

$$\beta^{M_d} = w_d \leq W \leq \left(\frac{1+\beta}{2}\right)^M D$$

- Solve for M gives, for each expert d

$$M \leq \frac{M_d \log \frac{1}{\beta} + \log D}{\log\left(\frac{2}{1+\beta}\right)}$$

- This holds for all experts, including the best expert. Thus

$$M \leq \min_d \frac{M_d \log \frac{1}{\beta} + \log D}{\log\left(\frac{2}{1+\beta}\right)} = \min_d [a_\beta M_d + c_\beta \log D]$$

- Dividing by T gives the rate at which the learner makes a mistake

$$\text{Learner's Mistake Rate} \leq a_\beta (\text{Best expert's rate}) + \frac{c_\beta \log D}{T}$$

Mistake Bound for Weighted Majority

- Suppose M_d be the # of mistakes of expert d. Then $w_d = \beta^{M_d}$. Therefore

$$\beta^{M_d} = w_d \leq W \leq \left(\frac{1+\beta}{2}\right)^M D$$

- Solve for M gives, for each expert d

$$M \leq \frac{M_d \log \frac{1}{\beta} + \log D}{\log\left(\frac{2}{1+\beta}\right)}$$

- This holds for all experts, including the best expert. Thus

$$M \leq \min_d \frac{M_d \log \frac{1}{\beta} + \log D}{\log\left(\frac{2}{1+\beta}\right)} = \min_d [a_\beta M_d + c_\beta \log D]$$

- Dividing by T gives the rate at which the learner makes a mistake

$$\text{Learner's Mistake Rate} \leq a_\beta(\text{Best expert's rate}) + \frac{c_\beta \log D}{T}$$

- $a_\beta \leq 2$, so the learner can at best be twice as bad as the best expert!

Randomized Weighted Majority

- We can actually do better

Randomized Weighted Majority

- We can actually do better
- Don't predict by simply using weighted majority vote

Randomized Weighted Majority

- We can actually do better
- Don't predict by simply using weighted majority vote
- Introduce randomness in this step

Randomized Weighted Majority

- We can actually do better
- Don't predict by simply using weighted majority vote
- Introduce randomness in this step
- Predict 1 with prob. n_1/W and predict 0 with prob. n_0/W (recall $n_1 = \sum_{d:\xi_{d,t}=1} w_d$ and $n_0 = \sum_{d:\xi_{d,t}=0} w_d$)

Randomized Weighted Majority

- We can actually do better
- Don't predict by simply using weighted majority vote
- Introduce randomness in this step
- Predict 1 with prob. n_1/W and predict 0 with prob. n_0/W (recall $n_1 = \sum_{d:\xi_{d,t}=1} w_d$ and $n_0 = \sum_{d:\xi_{d,t}=0} w_d$)
- Downweight experts that predicted incorrectly, i.e., $w_d = w_d\beta$

Randomized Weighted Majority

- We can actually do better
- Don't predict by simply using weighted majority vote
- Introduce randomness in this step
- Predict 1 with prob. n_1/W and predict 0 with prob. n_0/W (recall $n_1 = \sum_{d:\xi_{d,t}=1} w_d$ and $n_0 = \sum_{d:\xi_{d,t}=0} w_d$)
- Downweight experts that predicted incorrectly, i.e., $w_d = w_d\beta$
- In this case, the expectation of the no. of mistakes

$$\mathbb{E}[M] \leq \min_d [a_\beta M_d + c_\beta \log D]$$

where $a_\beta = \frac{\log(1/\beta)}{1-\beta}$ and $c_\beta = \frac{1}{1-\beta}$.

Winnow Algorithm

Winnow Algorithm

- Very similar to Perceptron, except the weight updates are multiplicative

Winnow Algorithm

- Very similar to Perceptron, except the weight updates are multiplicative
- Works very well when lots of features are irrelevant (it "winnows" out irrelevant features quickly)

Winnow Algorithm

- Very similar to Perceptron, except the weight updates are multiplicative
- Works very well when lots of features are irrelevant (it "winnows" out irrelevant features quickly)
- Assumes binary features $\{-1, +1\}$

Winnow Algorithm

- Very similar to Perceptron, except the weight updates are multiplicative
- Works very well when lots of features are irrelevant (it “winnows” out irrelevant features quickly)
- Assumes binary features $\{-1, +1\}$
- The basic idea of Winnow is as follows:

Winnow Algorithm

- Very similar to Perceptron, except the weight updates are multiplicative
- Works very well when lots of features are irrelevant (it "winnows" out irrelevant features quickly)
- Assumes binary features $\{-1, +1\}$
- The basic idea of Winnow is as follows:
  - Initialize all weights equally as $w_d = \frac{1}{D}$

Winnow Algorithm

- Very similar to Perceptron, except the weight updates are multiplicative
- Works very well when lots of features are irrelevant (it "winnows" out irrelevant features quickly)
- Assumes binary features $\{-1, +1\}$
- The basic idea of Winnow is as follows:
  - Initialize all weights equally as $w_d = \frac{1}{D}$
  - For round $t = 1, 2, \ldots, T$

Winnow Algorithm

- Very similar to Perceptron, except the weight updates are multiplicative
- Works very well when lots of features are irrelevant (it "winnows" out irrelevant features quickly)
- Assumes binary features $\{-1, +1\}$
- The basic idea of Winnow is as follows:
  - Initialize all weights equally as $w_d = \frac{1}{D}$
  - For round $t = 1, 2, \ldots, T$
    - Predict label as $\hat{y}_t = \text{sign}(\boldsymbol{w}_t^\top \boldsymbol{x}_t)$

Winnow Algorithm

- Very similar to Perceptron, except the weight updates are multiplicative
- Works very well when lots of features are irrelevant (it “winnows” out irrelevant features quickly)
- Assumes binary features $\{-1, +1\}$
- The basic idea of Winnow is as follows:
  - Initialize all weights equally as $w_d = \frac{1}{D}$
  - For round $t = 1, 2, \ldots, T$
    - Predict label as $\hat{y}_t = \text{sign}(\boldsymbol{w}_t^\top \boldsymbol{x}_t)$
    - Observe the true label $y_t \in \{-1, +1\}$

Winnow Algorithm

- Very similar to Perceptron, except the weight updates are multiplicative
- Works very well when lots of features are irrelevant (it "winnows" out irrelevant features quickly)
- Assumes binary features $\{-1, +1\}$
- The basic idea of Winnow is as follows:
  - Initialize all weights equally as $w_d = \frac{1}{D}$
  - For round $t = 1, 2, \ldots, T$
    - Predict label as $\hat{y}_t = \text{sign}(\boldsymbol{w}_t^\top \boldsymbol{x}_t)$
    - Observe the true label $y_t \in \{-1, +1\}$
    - If $y_t = \hat{y}_t$, $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t$, else $w_{t+1,d} \propto w_{t,d} \exp(\eta y_t x_{t,d})$

Winnow Algorithm

- Very similar to Perceptron, except the weight updates are multiplicative
- Works very well when lots of features are irrelevant (it "winnows" out irrelevant features quickly)
- Assumes binary features $\{-1, +1\}$
- The basic idea of Winnow is as follows:
  - Initialize all weights equally as $w_d = \frac{1}{D}$
  - For round $t = 1, 2, \ldots, T$
    - Predict label as $\hat{y}_t = \text{sign}(\boldsymbol{w}_t^\top \boldsymbol{x}_t)$
    - Observe the true label $y_t \in \{-1, +1\}$
    - If $y_t = \hat{y}_t$, $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t$, else $w_{t+1,d} \propto w_{t,d} \exp(\eta y_t x_{t,d})$
- Number of mistakes $\propto \frac{\log(D)}{\gamma_W^2}$ where $\gamma_W \propto 1/||\boldsymbol{w}_*||_1$ and $\boldsymbol{w}_*$ is optimal sol.

Winnow Algorithm

- Very similar to Perceptron, except the weight updates are multiplicative
- Works very well when lots of features are irrelevant (it "winnows" out irrelevant features quickly)
- Assumes binary features $\{-1, +1\}$
- The basic idea of Winnow is as follows:
  - Initialize all weights equally as $w_d = \frac{1}{D}$
  - For round $t = 1, 2, \ldots, T$
    - Predict label as $\hat{y}_t = \text{sign}(\boldsymbol{w}_t^\top \boldsymbol{x}_t)$
    - Observe the true label $y_t \in \{-1, +1\}$
    - If $y_t = \hat{y}_t$, $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t$, else $w_{t+1,d} \propto w_{t,d} \exp(\eta y_t x_{t,d})$
- Number of mistakes $\propto \frac{\log(D)}{\gamma_W^2}$ where $\gamma_W \propto 1/||\boldsymbol{w}_*||_1$ and $\boldsymbol{w}_*$ is optimal sol.
- For sparse $\boldsymbol{w}_*$ with small ℓ_1 norm, the bound is therefore better than Perceptron which assumes a non-sparse $\boldsymbol{w}_*$.

Online Learning: Some Comments

- Online Learning is fundamentally different from Batch Learning

Online Learning: Some Comments

- Online Learning is fundamentally different from Batch Learning
- No distinction between training and test data (no training data)

Online Learning: Some Comments

- Online Learning is fundamentally different from Batch Learning
- No distinction between training and test data (no training data)
- Looks at the notion of "regret" instead of generalization error

Online Learning: Some Comments

- Online Learning is fundamentally different from Batch Learning
- No distinction between training and test data (no training data)
- Looks at the notion of "regret" instead of generalization error
- Widely applicable and simple to implement algorithms

Online Learning: Some Comments

- Online Learning is fundamentally different from Batch Learning
- No distinction between training and test data (no training data)
- Looks at the notion of "regret" instead of generalization error
- Widely applicable and simple to implement algorithms
- Highly scalable and can also be used to solve batch learning problems

Online Learning: Some Comments

- Online Learning is fundamentally different from Batch Learning
- No distinction between training and test data (no training data)
- Looks at the notion of "regret" instead of generalization error
- Widely applicable and simple to implement algorithms
- Highly scalable and can also be used to solve batch learning problems
- Can also be used for learning from partial information ("Bandit Problems")

Online Learning: Some Comments

- Online Learning is fundamentally different from Batch Learning
- No distinction between training and test data (no training data)
- Looks at the notion of "regret" instead of generalization error
- Widely applicable and simple to implement algorithms
- Highly scalable and can also be used to solve batch learning problems
- Can also be used for learning from partial information ("Bandit Problems")
  - You make one of K choices and get information only about that choice (e.g., whether it's right/wrong or how much you will be rewarded for making that choice). Don't get any information about the outcomes of alternative choices

Online Learning: Some Comments

- Online Learning is fundamentally different from Batch Learning
- No distinction between training and test data (no training data)
- Looks at the notion of "regret" instead of generalization error
- Widely applicable and simple to implement algorithms
- Highly scalable and can also be used to solve batch learning problems
- Can also be used for learning from partial information ("Bandit Problems")
  - You make one of K choices and get information only about that choice (e.g., whether it's right/wrong or how much you will be rewarded for making that choice). Don't get any information about the outcomes of alternative choices
  - These are essentially "explore and exploit" style problems

Next (and Final) Class:
Survey of Some Topics We Didn't Cover