COL866: Foundations of Data Science

Ragesh Jaiswal, IITD

Machine Learning: Online learning and Perceptron

- The learning scenario that we have seen until now is called the batch learning scenario.
- We now discuss the online learning scenario where we remove the assumption that data is sampled from a fixed probability distribution (or from any probabilistic process at all).
- Here are some main ideas of online learning:
  - At each time $t = 1, 2, 3...$, the algorithm is presented with an arbitrary example $x_t \in \mathcal{X}$.
  - The algorithm is told the true label $c^\star(x_t)$ and is charged for a mistake, i.e., when $c^\star(x_t) \neq \ell_t$.
  - The goal of the algorithm is to make as few mistakes as possible.
- Online learning model is harder than the batch learning model. (In fact, we will show that an online algorithm can be converted to a batch learning algorithm)

- Here are some main ideas of online learning:
  - At each time $t = 1, 2, 3...$, the algorithm is presented with an arbitrary example $x_t \in \mathcal{X}$.
  - The algorithm is told the true label $c^{\star}(x_t)$ and is charged for a mistake, i.e., when $c^{\star}(x_t) \neq \ell_t$.
  - The goal of the algorithm is to make as few mistakes as possible.
- Case study:
  - Let $\mathcal{X} = \{0, 1\}^d$ and let the target hypothesis be a disjunction.
  - Question: Can you give an online algorithm that makes bounded number of mistakes?
  - Question: Argue that any deterministic algorithm makes at least r mistakes where r is the upper bound on the number of mistakes of the previous question.

- Here are some main ideas of online learning:
  - At each time $t = 1, 2, 3...$, the algorithm is presented with an arbitrary example $x_t \in \mathcal{X}$.
  - The algorithm is told the true label $c^\star(x_t)$ and is charged for a mistake, i.e., when $c^\star(x_t) \neq \ell_t$.
  - The goal of the algorithm is to make as few mistakes as possible.
- Case study:
  - Let $\mathcal{X} = \{0, 1\}^d$ and let the target hypothesis be a disjunction.
  - Question: Can you give an online algorithm that makes bounded number of mistakes?
  - Question: Argue that any deterministic algorithm makes at least r mistakes where r is the upper bound on the number of mistakes of the previous question.
  - Question: Show that there always exists an online algorithm that makes at most $\log_2 |\mathcal{H}|$ mistakes.

- Perceptron: An efficient algorithm for learning linear separator in $\mathbb{R}^d$ with mistake bound that depends on the margin of separation of the data.
- The assumption is that the target function can be described by a vector $\mathbf{w}^*$ such that for each positive example $\mathbf{x}$ we have $\mathbf{x}^T\mathbf{x}^* \geq 1$ and for each negative example we have $\mathbf{x}^T\mathbf{w}^* \leq -1$.
  - This essentially means there is a linear separator through the origin that separate the positive and negative data points such that all points are at a distance of at least $\gamma = \frac{1}{||\mathbf{x}^*||}$ from the separator.
  - γ is called the margin of separation.

- Perceptron: An efficient algorithm for learning linear separator in $\mathbb{R}^d$ with mistake bound that depends on the margin of separation of the data.
- The assumption is that the target function can be described by a vector $\mathbf{w}^*$ such that for each positive example $\mathbf{x}$ we have $\mathbf{x}^T\mathbf{x}^* \geq 1$ and for each negative example we have $\mathbf{x}^T\mathbf{w}^* \leq -1$.
  - This essentially means there is a linear separator through the origin that separate the positive and negative data points such that all points are at a distance of at least $\gamma = \frac{1}{||\mathbf{x}^*||}$ from the separator.
  - γ is called the margin of separation.
- The number of mistakes of the Perceptron algorithm is bounded by $(\frac{R}{\gamma})^2$ where $R = \max_t ||\mathbf{x}_t||$.

Algorithm

Perceptron($\mathbf{x}_1, ...$)

- $\mathbf{w} \leftarrow \mathbf{0}$
- For $t = 1, 2...$
  - Given $\mathbf{x}_t$, predict $sign(\mathbf{x}^T \mathbf{w})$
  - If the prediction was a mistake, then update:
    - If $\mathbf{x}_t$ was a positive data point, let $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}_t$
    - If $\mathbf{x}_t$ was a negative data point, let $\mathbf{w} \leftarrow \mathbf{w} - \mathbf{x}_t$

Theorem

On any sequence of examples $\mathbf{x}_1, \mathbf{x}_2, ...$, if there exists a vector $\mathbf{w}^*$ such that $\mathbf{x}_t^T \mathbf{w}^* \geq 1$ for positive examples and $\mathbf{x}_t^T \mathbf{w}^* \leq -1$ for negative examples, then the perceptron algorithm makes at most $R^2 \cdot ||\mathbf{w}^*||^2$ mistakes where $R = \max_t ||\mathbf{x}_t||$.

Algorithm

`Perceptron(`$\mathbf{x}_1, ...$`)`

- $\mathbf{w} \leftarrow \mathbf{0}$
- For $t = 1, 2...$
  - Given $\mathbf{x}_t$, predict $sign(\mathbf{x}^T \mathbf{w})$
  - If the prediction was a mistake, then update:
    - If $\mathbf{x}_t$ was a positive data point, let $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}_t$
    - If $\mathbf{x}_t$ was a negative data point, let $\mathbf{w} \leftarrow \mathbf{w} - \mathbf{x}_t$

Theorem

On any sequence of examples $\mathbf{x}_1, \mathbf{x}_2, ...,$ *if there exists a vector* $\mathbf{w}^*$ *such that* $\mathbf{x}_t^T \mathbf{w}^* \geq 1$ *for positive examples and* $\mathbf{x}_t^T \mathbf{w}^* \leq -1$ *for negative examples, then the perceptron algorithm makes at most* $R^2 \cdot ||\mathbf{w}^*||^2$ *mistakes where* $R = \max_t ||\mathbf{x}_t||$.

Proof sketch

- Claim 1: Each time the algorithm makes.a mistake, $\mathbf{w}^T \mathbf{w}^*$ increases by at least 1.

Algorithm

`Perceptron(`$\mathbf{x}_1, ...$`)`

- $\mathbf{w} \leftarrow \mathbf{0}$
- For $t = 1, 2...$
    - Given $\mathbf{x}_t$, predict $sign(\mathbf{x}^T\mathbf{w})$
    - If the prediction was a mistake, then update:
        - If $\mathbf{x}_t$ was a positive data point, let $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}_t$
        - If $\mathbf{x}_t$ was a negative data point, let $\mathbf{w} \leftarrow \mathbf{w} - \mathbf{x}_t$

Theorem

On any sequence of examples $\mathbf{x}_1, \mathbf{x}_2, ...$, if there exists a vector $\mathbf{w}^*$ such that $\mathbf{x}_t^T\mathbf{w}^* \geq 1$ for positive examples and $\mathbf{x}_t^T\mathbf{w}^* \leq -1$ for negative examples, then the perceptron algorithm makes at most $R^2 \cdot ||\mathbf{w}^*||^2$ mistakes where $R = \max_t ||\mathbf{x}_t||$.

Proof sketch

- Claim 1: Each time the algorithm makes.a mistake, $\mathbf{w}^T\mathbf{w}^*$ increases by at least 1.
- Claim 2: Each time the algorithm makes a mistake, $||\mathbf{w}||^2$ increases by at most R^2.

Algorithm

`Perceptron`$(\mathbf{x}_1, ...)$

- $\mathbf{w} \leftarrow \mathbf{0}$
- For $t = 1, 2...$
  - Given $\mathbf{x}_t$, predict $sign(\mathbf{x}^T\mathbf{w})$
  - If the prediction was a mistake, then update:
    - If $\mathbf{x}_t$ was a positive data point, let $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}_t$
    - If $\mathbf{x}_t$ was a negative data point, let $\mathbf{w} \leftarrow \mathbf{w} - \mathbf{x}_t$

Theorem

On any sequence of examples $\mathbf{x}_1, \mathbf{x}_2, ...$, *if there exists a vector* $\mathbf{w}^*$ *such that* $\mathbf{x}_t^T\mathbf{w}^* \geq 1$ *for positive examples and* $\mathbf{x}_t^T\mathbf{w}^* \leq -1$ *for negative examples, then the perceptron algorithm makes at most* $R^2 \cdot ||\mathbf{w}^*||^2$ *mistakes where* $R = \max_t ||\mathbf{x}_t||$.

Proof sketch

- <u>Claim 1</u>: Each time the algorithm makes.a mistake, $\mathbf{w}^T\mathbf{w}^*$ increases by at least 1.
- <u>Claim 2</u>: Each time the algorithm makes a mistake, $||\mathbf{w}||^2$ increases by at most R^2.
- So, if we make M mistakes, then $\mathbf{w}^T\mathbf{w}^* \geq M$ and $||\mathbf{w}|| \leq \sqrt{M}R$.
- We obtain the result using the fact that $\frac{\mathbf{w}^T\mathbf{w}^*}{||\mathbf{w}^*||} \leq ||\mathbf{w}||$. $\square$

Algorithm

`Perceptron`$(\mathbf{x}_1, ...)$

- $\mathbf{w} \leftarrow \mathbf{0}$
- For $t = 1, 2...$
  - Given $\mathbf{x}_t$, predict $sign(\mathbf{x}^T \mathbf{w})$
  - If the prediction was a mistake, then update:
    - If $\mathbf{x}_t$ was a positive data point, let $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}_t$
    - If $\mathbf{x}_t$ was a negative data point, let $\mathbf{w} \leftarrow \mathbf{w} - \mathbf{x}_t$

Theorem

On any sequence of examples $\mathbf{x}_1, \mathbf{x}_2, ...$, if there exists a vector $\mathbf{w}^*$ such that $\mathbf{x}_t^T \mathbf{w}^* \geq 1$ for positive examples and $\mathbf{x}_t^T \mathbf{w}^* \leq -1$ for negative examples, then the perceptron algorithm makes at most $R^2 \cdot ||\mathbf{w}^*||^2$ mistakes where $R = \max_t ||\mathbf{x}_t||$.

- A perfect linear separator may be too strong an assumption to be practical.
- A more reasonable assumption is that there exists a linear separator that is a "little bit wrong".
- We can do analysis in terms of the margin and hinge loss defined as follows:
  - For positive examples, hinge-loss is $\max(0, 1 - \mathbf{x}^T \mathbf{w}^*)$.
  - For negative examples, hinge-loss is $\max(0, 1 + \mathbf{x}^T \mathbf{w}^*)$.
  - Total hinge-loss is the sum of hinge loss for all examples.

- Suppose we we have a online learning algorithm with good mistake bound. Can we use this algorithm in the batch setting to obtain a hypothesis with low true error?
- More specifically:
  - Suppose online algorithm A has a mistake bound of M.
  - Let S be the training sample from an unknown distribution D.
  - Question: Can we use A as a subroutine in a batch learning algorithm to obtain a hypothesis h such that $err_D(h)$ is small?
- One idea: Random stopping
  - Let $t = |S| = \frac{M}{\varepsilon}$ be the size of the training set.
  - Let $\mathbf{x}_1, ..., \mathbf{x}_t$ be elements of the set S (in any arbitrary order).
  - Algorithm B: Pick a random integer $\ell \in \{1, ..., t\}$. Execute A on the sequence $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_\ell$ and let h denote the hypothesis of A at the end of the execution. Return h.

- Suppose we we have a online learning algorithm with good mistake bound. Can we use this algorithm in the batch setting to obtain a hypothesis with low true error?
- More specifically:
  - Suppose online algorithm A has a mistake bound of M.
  - Let S be the training sample from an unknown distribution D.
  - Question: Can we use A as a subroutine in a batch learning algorithm to obtain a hypothesis h such that $err_D(h)$ is small?
- One idea: Random stopping
  - Let $t = |S| = \frac{M}{\varepsilon}$ be the size of the training set.
  - Let $\mathbf{x}_1, ..., \mathbf{x}_t$ be elements of the set S (in any arbitrary order).
  - Algorithm B: Pick a random integer $\ell \in \{1, ..., t\}$. Execute A on the sequence $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_\ell$ and let h denote the hypothesis of A at the end of the execution. Return h.
  - Claim: $\mathbf{E}[err_D(h)] \leq \varepsilon$.

Boosting

- Consider binary classification such as spam filtering.
- It is easy to come up with simple "rule of thumb" classifiers. For example:
  - Emails with certain word in the subject line should be classified as spam. This word is learnt by looking at the training set.
  - Emails with lots of spelling errors for certain sensitive words should be classified as spam. Again, such list of sensitive words are learnt from the training set.
- The issue is that all these simple "rule of thumb" classifiers may be weak learners. That is, they may not output hypothesis with very high accuracy even though it may be better than random guessing. That is, the hypothesis may be accurate on $(1/2 + \gamma)$ fraction of examples for some small γ.
- Question: How do we convert a weak learner into a strong one?

- Consider binary classification such as spam filtering.
- It is easy to come up with simple "rule of thumb" classifiers. For example:
  - Emails with certain word in the subject line should be classified as spam. This word is learnt by looking at the training set.
  - Emails with lots of spelling errors for certain sensitive words should be classified as spam. Again, such list of sensitive words are learnt from the training set.
- The issue is that all these simple "rule of thumb" classifiers may be weak learners. That is, they may not output hypothesis with very high accuracy even though it may be better than random guessing. That is, the hypothesis may be accurate on $(1/2 + \gamma)$ fraction of examples for some small γ.
- Question: How do we convert a weak learner into a strong one?
  - Boosting: Repeatedly use the weak learner on appropriately re-weighted training set to obtain a sequence of hypothesis $h_1, h_2 \ldots$ (each one being slightly better than random guessing on the corresponding weighted set) and then use the majority of $h_1, h_2, \ldots$ as the classifier.

- Question: How do we convert a weak learner into a strong one?
  - Boosting: Repeatedly use the weak learner on appropriately re-weighted training set to obtain a sequence of hypothesis $h_1, h_2...$ (each one being slightly better than random guessing on the corresponding weighted set) and then use the majority of $h_1, h_2, ...$ as the classifier.

Algorithm

`Boosting`

- Given a sample S of n labeled examples $\mathbf{x}_1, ..., \mathbf{x}_n$, initialise the weight of exam example to $\mathbf{x}_i$ to $w_i = 1$
- Let $\mathbf{w} = (w_1, ..., w_n)$
- For $t = 1$ to t_0
  - Call the weak learner on the weighted sample $(S, \mathbf{w})$ to obtain a hypothesis h_t
  - Multiply the weight of each example that was misclassified by h_t by $\alpha = \frac{\frac{1}{2}+\gamma}{\frac{1}{2}-\gamma}$. Leave other weights as they were
- Return the classifier $MAJ(h_1, ..., h_{t_0})$

Algorithm

`Boosting`

- Given a sample S of n labeled examples $\mathbf{x}_1, ..., \mathbf{x}_n$, initialise the weight of exam example to $\mathbf{x}_i$ to $w_i = 1$
- Let $\mathbf{w} = (w_1, ..., w_n)$
- For $t = 1$ to t_0
  - Call the weak learner on the weighted sample $(S, \mathbf{w})$ to obtain a hypothesis h_t
  - Multiply the weight of each example that was misclassified by h_t by $\alpha = \frac{\frac{1}{2}+\gamma}{\frac{1}{2}-\gamma}$. Leave other weights as they were
- Return the classifier $MAJ(h_1, ..., h_{t_0})$

Definition (γ-Weak learner)

A weak learner is an algorithm that given examples, their labels, and a nonnegative real weights w_i on each example $\mathbf{x}_i$, produces a classifier that correctly labels a subset of examples with total weight at least $(\frac{1}{2} + \gamma) \sum_{i=1}^{n} w_i$.

Algorithm

Boosting

- Given a sample S of n labeled examples $\mathbf{x}_1, ..., \mathbf{x}_n$, initialise the weight of exam example to $\mathbf{x}_i$ to $w_i = 1$
- Let $\mathbf{w} = (w_1, ..., w_n)$
- For $t = 1$ to t_0
  - Call the weak learner on the weighted sample $(S, \mathbf{w})$ to obtain a hypothesis h_t
  - Multiply the weight of each example that was misclassified by h_t by $\alpha = \frac{\frac{1}{2}+\gamma}{\frac{1}{2}-\gamma}$. Leave other weights as they were
- Return the classifier $MAJ(h_1, ..., h_{t_0})$

Definition (γ-Weak learner)

A weak learner is an algorithm that given examples, their labels, and a nonnegative real weights w_i on each example $\mathbf{x}_i$, produces a classifier that correctly labels a subset of examples with total weight at least $(\frac{1}{2} + \gamma) \sum_{i=1}^{n} w_i$.

Theorem

Let A be a γ-weak learner for sample S. Then $t_0 = O(\frac{1}{\gamma^2} \log n)$ is sufficient so that the classifier $MAJ(h_1, ..., h_{t_0})$ produced by boosting algorithm has training error 0.

End