Activized Learning:
Transforming Passive to Active with Improved Label Complexity

Steve Hanneke

Machine Learning Department
Carnegie Mellon University
shanneke@cs.cmu.edu
Passive Learning

Learning Algorithm

Data Source

Raw Unlabeled Data

Labeled examples

Algorithm outputs a classifier

Expert / Oracle
Active Learning

Learning Algorithm

Data Source

Raw Unlabeled Data

Expert / Oracle

Request for the label of an example

The label of that example

Request for the label of another example

The label of that example

Algorithm outputs a classifier
Active Learning

How many label requests are required to learn?

Label Complexity

e.g., Das04, Das05, DKM05, BBL06, Kaa06, Han07a&b, BBZ07, DHM07, BHW08
Activized Learning

“Activizer” Meta-algorithm

Data Source

Request for the label of an example
The label of that example
Request for the label of another example
The label of that example

Raw Unlabeled Data

Algorithm outputs a classifier

Passive Learning
Algorithm
(Supervised / Semi-Supervised)

Steve Hanneke
Passive Learning

Algorithm
(Supervised / Semi-Supervised)

Classifier
Dataset

Request for the label of another example
Request for the label of an example

“Activizer”
Meta-algorithm

Data
Source

Raw Unlabeled Data

Expert / Oracle

Algorithm outputs a classifier

Are there general-purpose activizers that strictly improve the label complexity of any passive algorithm?
An Example: Threshold Classifiers

A simple activizer for any threshold-learning algorithm.
An Example: Threshold Classifiers

A simple activizer for any threshold-learning algorithm.

- Take $n/2$ unlabeled examples, request their labels
- Locate the closest -/+ points: $a, b$
- Estimate $P([a,b])$, and sample $\approx n/(4P([a,b]))$ unlabeled examples
- Request the labels in $[a,b]$
- Label rest ourselves.
- Train passive alg on all examples.

Used only $n$ label requests,
but get a classifier trained on $\Omega(n^2)$ examples!

Improvement in label complexity over passive.
(in this case, apply idea sequentially to get exponential improvement)
Outline

- Formal model
- Exciting New Results 😊
- Dealing with noise?
- Conclusions & open problems
Formal Model

\( \mathcal{X} \): Instance space
\( \mathbb{C} \): Concept space (a set of classifiers \( h : \mathcal{X} \to \{-1, 1\} \))
\( d \): VC dimension of \( \mathbb{C} \) (assume \( d < \infty \))
\( \mathcal{D} \): Distribution over \( \mathcal{X} \)

Unknown target function \( f \in \mathbb{C} \)
\( \text{er}(h) = \mathbb{P}_{X \sim \mathcal{D}}[h(X) \neq f(X)] \)

Sequence of i.i.d. training examples \( x_1, x_2, \ldots \sim \mathcal{D} \)

Algorithm chooses any \( x_i \), receives label \( f(x_i) \), repeat

The objective is to produce some \( h : \mathcal{X} \to \{-1, 1\} \) s.t. \( \text{er}(h) \) is small.
Formal Model

**Definition:** An algorithm $A(n, \delta)$ achieves label complexity $\Lambda(\epsilon, \delta, f, \mathcal{D})$ for $\mathcal{C}$ if it outputs a classifier $h_n$ after at most $n$ label requests, and for any target function $f \in \mathcal{C}$, distribution $\mathcal{D}$, $\epsilon > 0$, $\delta > 0$, for any $n \geq \Lambda(\epsilon, \delta, f, \mathcal{D})$,

$$\mathbb{P}[er(h_n) \leq \epsilon] \geq 1 - \delta.$$

**Definition:** Suppose $A_p$ is a passive algorithm achieving a label complexity $\Lambda_p(\epsilon, \delta, f, \mathcal{D})$ for $\mathcal{C}$. A (meta-)algorithm $A_a$ *activizes* $A_p$ for $\mathcal{C}$ if $A_a(A_p, n, \delta)$ achieves a label complexity $\Lambda_a(\epsilon, \delta, f, \mathcal{D})$ for $\mathcal{C}$, where $\exists c < \infty$ s.t. $\forall f \in \mathcal{C}, \mathcal{D}: 1 \ll \Lambda_p(\epsilon, \delta, f, \mathcal{D}) \ll \infty$,

$$\Lambda_a(c\epsilon, c\delta, f, \mathcal{D}) = o(\Lambda_p(\epsilon, \delta, f, \mathcal{D})).$$

Recall $s(\epsilon) = o(t(\epsilon))$ iff $\lim_{\epsilon \to 0} \frac{s(\epsilon)}{t(\epsilon)} = 0$.  

Steve Hanneke  11
Naïve Approach

Algorithm: \textbf{NaiveActivizer}(\mathcal{A}_p,n,\delta)

0. Sample \( n/2 \) examples \( Q \), request their labels
1. Let \( V \leftarrow \{ h \in \mathcal{C} : \text{er}_Q(h) = 0 \} \)
2. Estimate \( \hat{\Delta} \approx \mathbb{P}(x : \exists h_1, h_2 \in V \text{ s.t. } h_1(x) \neq h_2(x)) \)
3. Sample \( \approx n/(4\hat{\Delta}) \) examples \( \mathcal{L} \)
4. Request label of all \( x \) s.t. \( \exists h_1, h_2 \in V : h_1(x) \neq h_2(x) \)
5. Label the rest ourselves
6. Return the output of \( \mathcal{A}_p(\mathcal{L},\delta) \)

Produces a perfectly labeled data set, which we can feed into any passive algorithm! So we get a natural fallback guarantee.

But does it always improve over the passive algorithm?
Naïve Approach

Algorithm: \textbf{Naive Activizer}(\mathcal{A}_p,n,\delta)

0. Sample \( n/2 \) examples \( Q \), request their labels
1. Let \( V \leftarrow \{h \in \mathcal{C} : err_Q(h) = 0\} \)
2. Estimate \( \hat{\Delta} \approx P(x : \exists h_1, h_2 \in V \text{ s.t. } h_1(x) \neq h_2(x)) \)
3. Sample \( \approx n/(4\hat{\Delta}) \) examples \( \mathcal{L} \)
4. Request label of all \( x \) s.t. \( \exists h_1, h_2 \in V : h_1(x) \neq h_2(x) \)
5. Label the rest ourselves
6. Return the output of \( \mathcal{A}_p(\mathcal{L}, \delta) \)

A more subtle example: Intervals

\[
\begin{array}{c}
0 & - & + & - & 1 \\
\end{array}
\]
Naïve Approach

Algorithm: NaiveActivizer(\(A_p, n, \delta\))
0. Sample \(n/2\) examples \(Q\), request their labels
1. Let \(V \leftarrow \{h \in \mathcal{C} : er_Q(h) = 0\}\)
2. Estimate \(\hat{\Delta} \approx \mathbb{P}(x : \exists h_1, h_2 \in V \text{ s.t. } h_1(x) \neq h_2(x))\)
3. Sample \(\approx n/(4\hat{\Delta})\) examples \(\mathcal{L}\)
4. Request label of all \(x\) s.t. \(\exists h_1, h_2 \in V : h_1(x) \neq h_2(x)\)
5. Label the rest ourselves
6. Return the output of \(A_p(\mathcal{L}, \delta)\)

A more subtle example: Intervals

Suppose the target labels everything “-1”

Passive algorithm still trained with just \(O(n)\) examples. No improvements. 😞
A Simple Activizer

Algorithm: \textbf{SimpleActivizer}(A_p,n,\delta)

0. Sample \(n/3\) examples \(Q\), request their labels
1. Let \(V \leftarrow \{h \in \mathbb{C} : \text{err}_Q(h) = 0\}\), \(S \leftarrow \{\}\)
2. For \(k = 1, 2, \ldots, d + 1\) (where \(d = VC(\mathbb{C})\))
3. Estimate \(\hat{\Delta} \approx \mathbb{P}(x : V \text{ shatters } S \cup \{x\})\)
4. Sample \(\approx n/(6d\hat{\Delta})\) examples \(L_k\)
5. Request label of all \(x\) s.t. \(V\) shatters \(S \cup \{x\}\)
6. Label the rest ourselves (opposite to unrealizable labels)
7. Sample \(x_k\) s.t. \(V\) shatters \(S \cup \{x_k\}\) (if exists), add to \(S\)
8. Return \textbf{ActiveSelect}(\{A_p(L_1, \delta), \ldots, A_p(L_{d+1}, \delta)\}, n/3)

Subroutine: \textbf{ActiveSelect}(\{h_1, h_2, \ldots, h_{d+1}\}, m)

0. For each pair \(h_i, h_j\)
1. Sample \(m/(d + 1)^2\) examples \(x\) s.t. \(h_i(x) \neq h_j(x)\)
2. Let \(m_{ij}\) denote the number of mistakes \(h_i\) makes
3. Return \(h_{\hat{i}}\), where \(\hat{i} = \arg\min_i \max_j m_{ij}\)
A Simple Activizer

Algorithm: \textbf{SimpleActivizer}(A_p,n,\delta)
0. Sample \(n/3\) examples \(Q\), request their labels
1. Let \(V \leftarrow \{h \in \mathbb{C} : er_Q(h) = 0\}\), \(S \leftarrow \{\}\)
2. For \(k = 1, 2, \ldots, d + 1\) (where \(d = VC(\mathbb{C})\))
3. Estimate \(\hat{\Delta} \approx \mathbb{P}(x : V \text{ shatters } S \cup \{x\})\)
4. Sample \(\approx n/(6d\hat{\Delta})\) examples \(\mathcal{L}_k\)
5. Request label of all \(x\) s.t. \(V\) shatters \(S \cup \{x\}\)
6. Label the rest ourselves (opposite to unrealizable labels)
7. Sample \(x_k\) s.t. \(V\) shatters \(S \cup \{x_k\}\) (if exists), add to \(S\)
8. Return \textbf{ActiveSelect}\((\{A_p(\mathcal{L}_1, \delta), \ldots, A_p(\mathcal{L}_{d+1}, \delta)\}, n/3)\)

Intervals revisited

Again, suppose the target labels everything “-1”

Passive algorithm trained on \(\Omega(n^2)\) samples. Improved label complexity. 😊

(can apply steps 0/1 and 5 sequentially, updating \(V\) after every label request, for more savings)
Does This Activize Any Passive Algorithm?
This Activizes Any Passive Algorithm!

**Theorem:** For any $\mathcal{C}$, SimpleActivizer activizes any passive learning algorithm.

**Corollary:** For any $\mathcal{C}$, there is an active learning algorithm that achieves a label complexity $\Lambda_a(\epsilon, \delta, f, \mathcal{D})$ such that $\forall f \in \mathcal{C}, \mathcal{D}$,

$$\Lambda_a(\epsilon, \delta, f, \mathcal{D}) = o(1/\epsilon).$$

[HLW94] passive algorithm has $O(1/\epsilon)$ sample complexity.
This Activizes Any Passive Algorithm!

**Theorem:** For any $C$, SimpleActivizer activates any passive learning algorithm.

Proof idea: if $\hat{\Delta} \to 0$ for $k = 1$, we’re done.
Otherwise, $\lim_{n \to \infty} \mathbb{P}\{x : \exists h_1, h_2 \in V, h_1(x) \neq h_2(x)\} > c$, for some $c$.
For large enough $n$, $x_1$ will be in this limiting region.
In particular, $\inf_{h \in V : h(x) = +1} er(h) = \inf_{h \in V : h(x) = -1} er(h) = 0$.
So (w.p.1), for any $x$ agreed upon by all $h \in V : h(x_1) = +1$ or all $h \in V : h(x_1) = -1$, the agreed upon label is correct.

So basically, we know the label of any $x$ s.t. $\{x_1, x\}$ is not shattered.
Repeat the argument for $k > 1$ until we get a $k$ where $\hat{\Delta} \to 0$, but then $|\mathcal{L}_k| \gg n$, so we’re done.
Efficiency?

- Need to be able to test shatterability of a set of $\leq d$ points, subject to consistency with a set of $O(n)$ labeled examples.

- For some concept spaces, could be exponential in $d$ (or worse).
- But in many cases, it may be efficient. (e.g., linear separators?)
Dealing with Noise

Have an arbitrary distribution $\mathcal{D}_{XY}$ over $\mathcal{X} \times \{-1, +1\}$, so label complexity for $\mathcal{C}$ is written $\Lambda(\epsilon, \delta, \mathcal{D}_{XY})$. Now $\epsilon$ represents excess over best error rate in $\mathcal{C}$: want to guarantee

$$\mathbb{P} \left[ er(h_n) - \inf_{f \in \mathcal{C}} er(f) \leq \epsilon \right] \geq 1 - \delta.$$
Dealing with Noise

Replace version space \( V = \{ h \in \mathbb{C} : er_Q(h) = 0 \} \) with noise-robust version space

\[
V = \{ h \in \mathbb{C} : er_Q(h) - \min_{h' \in \mathbb{C}} er_Q(h') \leq O(n^{-1/2}) \}.
\]

Applied to a particular passive algorithm, this modification of SimpleActivizer achieves label complexity\(^1\)

\[
\Lambda_a(\epsilon, \delta, D_{XY}) = o(1/\epsilon^2).
\]

Under Tsybakov’s noise conditions w/ exponent \( \kappa \), a more careful variant achieves

\[
\Lambda_a(\epsilon, \delta, D_{XY}) = o(1/\epsilon^{2-1/\kappa}).
\]

Open Question: Can we activize any passive algorithm, even with noise?
Open Question: Can we activize some empirical error minimizing algorithm?

\(^1\)Technically, an additional slight modification is needed to handle the case where the Bayes optimal classifier is not in \( \mathbb{C} \). Details included in a forthcoming paper.
Conclusions & Open Questions

- Can activize any passive learning algorithm (in the zero-error, finite VC dimension case)

- Question: What about infinite VC dimension?
- Question: Can we give more detailed bounds on $\Lambda_a$?
- Question: Can we always activize, even when there is noise?
