Active learning

Sanjoy Dasgupta

University of California, San Diego

Exploiting unlabeled data

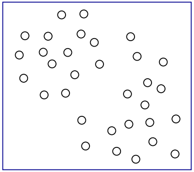
A lot of unlabeled data is plentiful and cheap, eg.

documents off the web

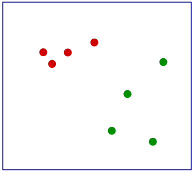
speech samples

images and video

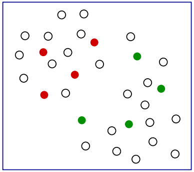
But labeling can be expensive.



Unlabeled points

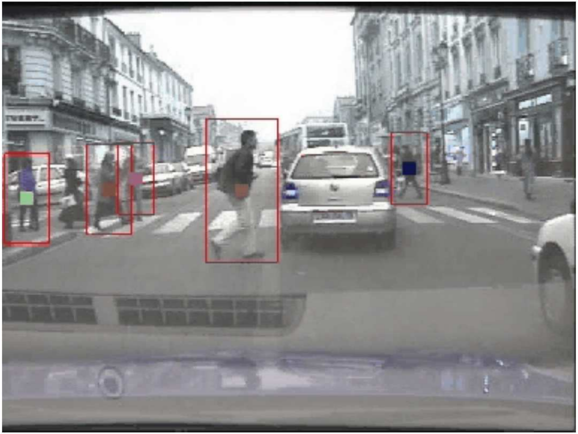


Supervised learning



Semisupervised and active learning

Active learning example: pedestrian detection [Freund et al 03]



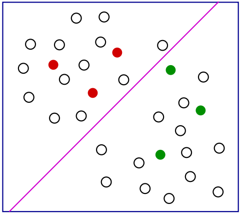
Typical heuristics for active learning

Start with a pool of unlabeled data

Pick a few points at random and get their labels

Repeat

> Fit a classifier to the labels seen so far
> Query the unlabeled point that is closest to the boundary (or most uncertain, or most likely to decrease overall uncertainty,...)



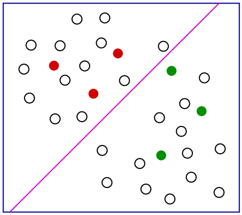
Typical heuristics for active learning

Start with a pool of unlabeled data

Pick a few points at random and get their labels

Repeat

Fit a classifier to the labels seen so far
Query the unlabeled point that is closest to the boundary
(or most uncertain, or most likely to decrease overall
uncertainty,...)



How to analyze such schemes?

The statistical learning theory framework

Unknown, underlying distribution $\mathbb{P}$ on the (data, label) space.
Hypothesis class H of candidate classifiers.
Target: the $h^* \in H$ that has fewest errors on $\mathbb{P}$.

The statistical learning theory framework

Unknown, underlying distribution $\mathbb{P}$ on the (data, label) space.
Hypothesis class H of candidate classifiers.
Target: the $h^* \in H$ that has fewest errors on $\mathbb{P}$.

Get n samples from $\mathbb{P}$, choose $h_n \in H$ that does well on these.

The statistical learning theory framework

Unknown, underlying distribution $\mathbb{P}$ on the (data, label) space.
Hypothesis class H of candidate classifiers.
Target: the $h^* \in H$ that has fewest errors on $\mathbb{P}$.

Get n samples from $\mathbb{P}$, choose $h_n \in H$ that does well on these.

We'd like: $h_n \to h^*$, as rapidly as possible.

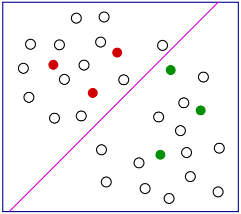
Typical heuristics for active learning

Start with a pool of unlabeled data

Pick a few points at random and get their labels

Repeat

> Fit a classifier to the labels seen so far
> Query the unlabeled point that is closest to the boundary (or most uncertain, or most likely to decrease overall uncertainty,...)



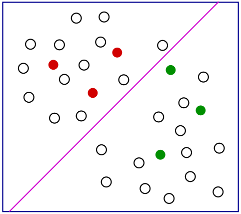
Typical heuristics for active learning

Start with a pool of unlabeled data

Pick a few points at random and get their labels

Repeat

Fit a classifier to the labels seen so far
Query the unlabeled point that is closest to the boundary
(or most uncertain, or most likely to decrease overall
uncertainty,...)



Biased sampling: the labeled points are not representative of the underlying distribution.

Sampling bias

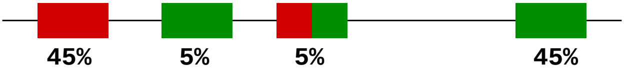
Start with a pool of unlabeled data

Pick a few points at random and get their labels

Repeat

Fit a classifier to the labels seen so far
Query the unlabeled point that is closest to the boundary
(or most uncertain, or most likely to decrease overall uncertainty,...)

Example: data in $\mathbb{R}$, $H = \{\text{thresholds}\}$.



Sampling bias

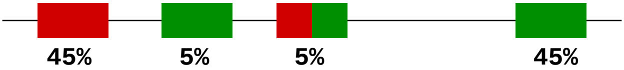
Start with a pool of unlabeled data

Pick a few points at random and get their labels

Repeat

Fit a classifier to the labels seen so far
Query the unlabeled point that is closest to the boundary
(or most uncertain, or most likely to decrease overall uncertainty,...)

Example: data in $\mathbb{R}$, $H = \{\text{thresholds}\}$.



Even with infinitely many labels, converges to a classifier with 5% error instead of the best achievable, 2.5%. *Not consistent.*

Manifestation in practice, eg. Schutze et al 03.

Three ways to manage sampling bias

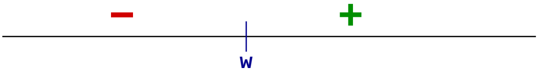
1. Label everything.
2. Use importance weighting.
3. Explicitly manage sampling regions.

Can adaptive querying really help?

Threshold functions on the real line:

$H = \{h_w : w \in \mathbb{R}\}$

$h_w(x) = 1(x \geq w)$



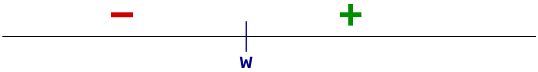
Supervised: for misclassification error $\leq \epsilon$, need $\approx 1/\epsilon$ labeled points.

Can adaptive querying really help?

Threshold functions on the real line:

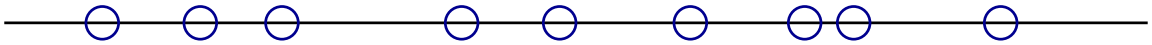
$H = \{h_w : w \in \mathbb{R}\}$

$h_w(x) = 1(x \geq w)$



Supervised: for misclassification error $\leq \epsilon$, need $\approx 1/\epsilon$ labeled points.

Active learning: instead, start with $1/\epsilon$ *unlabeled* points.

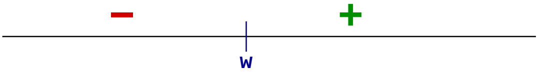


Binary search: need just $\log 1/\epsilon$ labels, from which the rest can be inferred. *Exponential improvement in label complexity*.

Can adaptive querying really help?

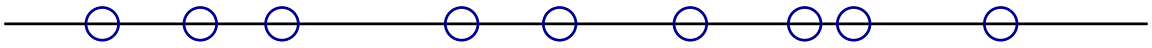
Threshold functions on the real line:

$H = \{h_w : w \in \mathbb{R}\}$
$h_w(x) = 1(x \geq w)$



Supervised: for misclassification error $\leq \epsilon$, need $\approx 1/\epsilon$ labeled points.

Active learning: instead, start with $1/\epsilon$ *unlabeled* points.



Binary search: need just $\log 1/\epsilon$ labels, from which the rest can be inferred. *Exponential improvement in label complexity*.

Challenges: Nonseparable data? Other hypothesis classes?

Three ways to manage sampling bias

1. Label everything.
2. Use importance weighting.
3. Explicitly manage sampling regions.

A generic, mellow active learner [Cohn-Atlas-Ladner '91]

For *separable* data that is streaming in.

H_1 = hypothesis class

Repeat for $t = 1, 2, \ldots$

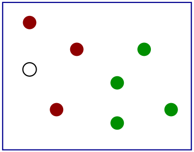
Receive unlabeled point x_t

If there is any disagreement within H_t about x_t's label:

query label y_t and set $H_{t+1} = \{h \in H_t : h(x_t) = y_t\}$

else

$H_{t+1} = H_t$



Is a label needed?

A generic, mellow active learner [Cohn-Atlas-Ladner '91]

For *separable* data that is streaming in.

H_1 = hypothesis class

Repeat for $t = 1, 2, \ldots$

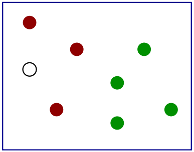
Receive unlabeled point x_t

If there is any disagreement within H_t about x_t's label:

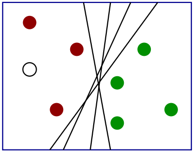
query label y_t and set $H_{t+1} = \{h \in H_t : h(x_t) = y_t\}$

else

$H_{t+1} = H_t$



Is a label needed?



H_t = current candidate hypotheses

A generic, mellow active learner [Cohn-Atlas-Ladner '91]

For *separable* data that is streaming in.

H_1 = hypothesis class

Repeat for $t = 1, 2, \ldots$

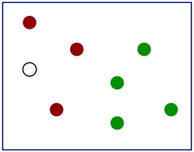
Receive unlabeled point x_t

If there is any disagreement within H_t about x_t's label:

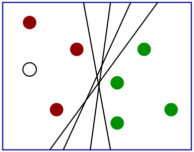
query label y_t and set $H_{t+1} = \{h \in H_t : h(x_t) = y_t\}$

else

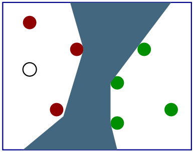
$H_{t+1} = H_t$



Is a label needed?



H_t = current candidate hypotheses



Region of uncertainty

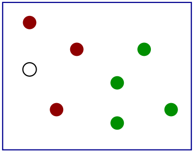
A generic, mellow active learner [Cohn-Atlas-Ladner '91]

For *separable* data that is streaming in.

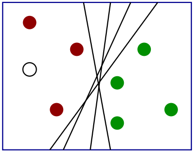
$H_1 =$ hypothesis class

Repeat for $t = 1, 2, \ldots$

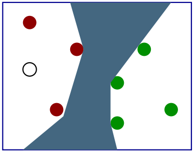
Receive unlabeled point x_t
If there is any disagreement within H_t about x_t's label:
query label y_t and set $H_{t+1} = \{h \in H_t : h(x_t) = y_t\}$
else
$H_{t+1} = H_t$



Is a label needed?



$H_t =$ current candidate hypotheses



Region of uncertainty

Issues: (1) intractable to maintain H_t; (2) nonseparable data; (3) how many labels are used?

Ideas for an agnostic active learner

Hypothesis class H, general (nonseparable) data distribution $\mathbb{P}$ from which data arrives in a stream.

1. Label everything.
   - I: points whose labels are inferred.
   - Q: points whose labels are queried.
2. Invariant: labels in I agree with the optimal hypothesis h^*.
3. "Version space" maintained implicitly through I, Q.

Ideas for an agnostic active learner

Hypothesis class H, general (nonseparable) data distribution $\mathbb{P}$ from which data arrives in a stream.

1. Label everything.
   - I: points whose labels are inferred.
   - Q: points whose labels are queried.
2. Invariant: labels in I agree with the optimal hypothesis h^*.
3. "Version space" maintained implicitly through I, Q.

To make this happen, we assume $\exists$ supervised learning black box that takes *two* labeled data sets, I and Q.

> *LEARN*(I, Q) *returns a hypothesis* $h \in H$ *that is consistent with* I *and has minimum error on* Q.

Active learning algorithm [D-Hsu-Monteleoni '07]

Recall: LEARN(I, Q) returns a hypothesis $h \in H$ that is consistent with I and has minimum error on Q.

1. Initialize $I = Q = \emptyset$.
2. For $t = 1, 2, \ldots$:
   - Receive a new point x_t.
   - $h_+ = \text{LEARN}(I \cup (x_t, +1), Q)$ and $h_- = \text{LEARN}(I \cup (x_t, -1), Q)$.
   - If $\text{err}(h_-, I \cup Q) - \text{err}(h_+, I \cup Q) > \Delta_t$: add $(x_t, +1)$ to I.
   - If $\text{err}(h_+, I \cup Q) - \text{err}(h_-, I \cup Q) > \Delta_t$: add $(x_t, -1)$ to I.
   - Otherwise query x_t's label y_t, and add (x_t, y_t) to Q.
3. When done: return LEARN(I, Q).

Here Δ_t comes from a generalization bound.

Active learning algorithm [D-Hsu-Monteleoni '07]

Recall: LEARN(I, Q) returns a hypothesis $h \in H$ that is consistent with I and has minimum error on Q.

1. Initialize $I = Q = \emptyset$.
2. For $t = 1, 2, \ldots$:
   - Receive a new point x_t.
   - $h_+ = \text{LEARN}(I \cup (x_t, +1), Q)$ and $h_- = \text{LEARN}(I \cup (x_t, -1), Q)$.
   - If $\text{err}(h_-, I \cup Q) - \text{err}(h_+, I \cup Q) > \Delta_t$: add $(x_t, +1)$ to I.
   - If $\text{err}(h_+, I \cup Q) - \text{err}(h_-, I \cup Q) > \Delta_t$: add $(x_t, -1)$ to I.
   - Otherwise query x_t's label y_t, and add (x_t, y_t) to Q.
3. When done: return LEARN(I, Q).

Here Δ_t comes from a generalization bound.

But I might not agree with the actual (unseen) labels – so isn't there sampling bias?

Sampling bias

Inferred labels I might differ from the actual labels $\tilde{I}$.

Standard generalization (eg. Vapnik-Chervonenkis) theory lets us bound how much empirical estimates based on $\tilde{I} \cup Q$ differ from true values based on the underlying distribution $\mathbb{P}$.

Sampling bias

Inferred labels I might differ from the actual labels $\tilde{I}$.

Standard generalization (eg. Vapnik-Chervonenkis) theory lets us bound how much empirical estimates based on $\tilde{I} \cup Q$ differ from true values based on the underlying distribution $\mathbb{P}$.

- The difference between empirical error and true error in H:

$$\sup_{h \in H} \left| \mathrm{err}(h, \tilde{I} \cup Q) - \mathrm{err}(h, \mathbb{P}) \right|.$$

- The same thing for differences between pairs of classifiers:

$$\sup_{h, h' \in H} (\mathrm{err}(h, \tilde{I} \cup Q) - \mathrm{err}(h', \tilde{I} \cup Q)) - (\mathrm{err}(h, \mathbb{P}) - \mathrm{err}(h', \mathbb{P})).$$

Sampling bias

Inferred labels I might differ from the actual labels $\tilde{I}$.

Standard generalization (eg. Vapnik-Chervonenkis) theory lets us bound how much empirical estimates based on $\tilde{I} \cup Q$ differ from true values based on the underlying distribution $\mathbb{P}$.

- The difference between empirical error and true error in H:

$$\sup_{h \in H} |\mathrm{err}(h, \tilde{I} \cup Q) - \mathrm{err}(h, \mathbb{P})|.$$

- The same thing for differences between pairs of classifiers:

$$\sup_{h, h' \in H} (\mathrm{err}(h, \tilde{I} \cup Q) - \mathrm{err}(h', \tilde{I} \cup Q)) - (\mathrm{err}(h, \mathbb{P}) - \mathrm{err}(h', \mathbb{P})).$$

All hypotheses h, h' that we handle are consistent with I, so distortions in I's labels translates their error estimates equally:

$$\mathrm{err}(h, I \cup Q) - \mathrm{err}(h', I \cup Q) = \mathrm{err}(h, \tilde{I} \cup Q) - \mathrm{err}(h', \tilde{I} \cup Q).$$

Consistency

Recall: LEARN(I, Q) returns a hypothesis $h \in H$ that is consistent with I and has minimum error on Q.

1. Initialize $I = Q = \emptyset$.
2. For $t = 1, 2, \ldots$:
   - Receive a new point x_t.
   - $h_+ = \text{LEARN}(I \cup (x_t, +1), Q)$ and $h_- = \text{LEARN}(I \cup (x_t, -1), Q)$.
   - If $\text{err}(h_-, I \cup Q) - \text{err}(h_+, I \cup Q) > \Delta_t$: add $(x_t, +1)$ to I.
   - If $\text{err}(h_+, I \cup Q) - \text{err}(h_-, I \cup Q) > \Delta_t$: add $(x_t, -1)$ to I.
   - Otherwise query x_t's label y_t, and add (x_t, y_t) to Q.
3. When done: return LEARN(I, Q).

Guarantee: (with high probability) all inferred labels are consistent with the optimal $h^* \in H$.

Label complexity bounds

The label complexity of mellow active learning can be captured by the *disagreement coefficient* θ and the VC dimension d.

Label complexity bounds

The label complexity of mellow active learning can be captured by the *disagreement coefficient* θ and the VC dimension d.

- Separable case (CAL) [Hanneke '07].

  To achieve misclassification rate ϵ with probability 0.9, suffices to have

  $$\theta d \log \frac{1}{\epsilon}$$

  labels. Usual supervised requirement: d/ϵ.

Label complexity bounds

The label complexity of mellow active learning can be captured by the *disagreement coefficient* θ and the VC dimension d.

- Separable case (CAL) [Hanneke '07].

  To achieve misclassification rate ϵ with probability 0.9, suffices to have

  $$\theta d \log \frac{1}{\epsilon}$$

  labels. Usual supervised requirement: d/ϵ.

- Nonseparable case [Balcan-Beygelzimer-Langford '06, D-Hsu-Monteleoni '07, Koltchinskii '10, Hanneke '11].

  If best achievable error rate is ν, suffices to have

  $$\theta \left(d \log^2 \frac{1}{\epsilon} + \frac{d\nu^2}{\epsilon^2} \right)$$

  labels. Usual supervised requirement: $d/\epsilon + d\nu/\epsilon^2$.

Label complexity bounds

The label complexity of mellow active learning can be captured by the *disagreement coefficient* θ and the VC dimension d.

- Separable case (CAL) [Hanneke '07].

  To achieve misclassification rate ϵ with probability 0.9, suffices to have

  $$\theta d \log \frac{1}{\epsilon}$$

  labels. Usual supervised requirement: d/ϵ.

- Nonseparable case [Balcan-Beygelzimer-Langford '06, D-Hsu-Monteleoni '07, Koltchinskii '10, Hanneke '11].

  If best achievable error rate is ν, suffices to have

  $$\theta \left(d \log^2 \frac{1}{\epsilon} + \frac{d\nu^2}{\epsilon^2} \right)$$

  labels. Usual supervised requirement: $d/\epsilon + d\nu/\epsilon^2$.

- Lower bound [Beygelzimer-D-Langford '09].

  In the nonseparable case, a factor of $d\nu^2/\epsilon^2$ is inevitable.

Disagreement coefficient [Hanneke]

Let $\mathbb{P}$ be the underlying probability distribution on input space $\mathcal{X}$.
Induces (pseudo-)metric on hypotheses: $d(h, h') = \mathbb{P}[h(X) \neq h'(X)]$.
Corresponding notion of *ball* $B(h, r) = \{h' \in H : d(h, h') < r\}$.

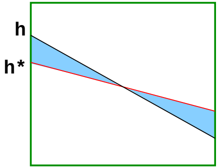
Disagreement region of any set of candidate hypotheses $V \subseteq H$:

$$\text{DIS}(V) = \{x : \exists h, h' \in V \text{ such that } h(x) \neq h'(x)\}.$$

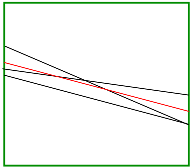
Disagreement coefficient for target hypothesis $h^* \in H$:

$$\theta = \sup_{r>0} \frac{\mathbb{P}[\text{DIS}(B(h^*, r))]}{r}.$$

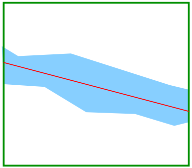
(cf. Alexander capacity function)



$d(h^*, h) = \mathbb{P}$[shaded region]



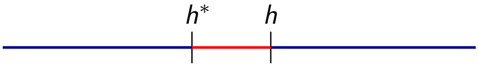
Some elements of $B(h^*, r)$



$\text{DIS}(B(h^*, r))$

Example: $H = \{\text{thresholds in } \mathbb{R}\}$

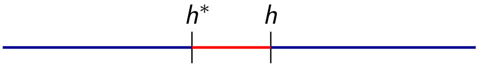
Consider any two thresholds h^*, h:



$$
\begin{aligned}
d(h^*, h) &= \mathbb{P}[h^*(X) \neq h(X)] \\
&= \text{probability mass of (data in) the red region}
\end{aligned}
$$

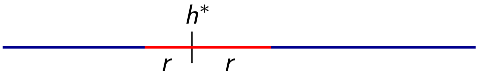
Example: $H = \{\text{thresholds in } \mathbb{R}\}$

Consider any two thresholds h^*, h:



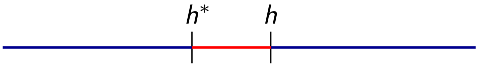
$$d(h^*, h) = \mathbb{P}[h^*(X) \neq h(X)]$$
$$= \text{probability mass of (data in) the red region}$$

$B(h^*, r)$ consists of thresholds within r probability mass of h^*:



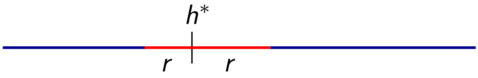
Example: $H = \{\text{thresholds in } \mathbb{R}\}$

Consider any two thresholds h^*, h:



$$d(h^*, h) = \mathbb{P}[h^*(X) \neq h(X)]$$
$$= \text{probability mass of (data in) the red region}$$

$B(h^*, r)$ consists of thresholds within r probability mass of h^*:



Therefore the disagreement coefficient is

$$\theta = \sup_r \frac{\mathbb{P}[\text{DIS}(B(h^*, r))]}{r} = 2.$$

Disagreement coefficient: examples

- Thresholds in $\mathbb{R}$, any data distribution.

$$\theta = 2.$$

Label complexity $O(\log 1/\epsilon)$.

Disagreement coefficient: examples

- Thresholds in $\mathbb{R}$, any data distribution.

  $$\theta = 2.$$

  Label complexity $O(\log 1/\epsilon)$.

- Linear separators through the origin in $\mathbb{R}^d$, uniform data distribution. [Hanneke '07]

  $$\theta \leq \sqrt{d}.$$

  Label complexity $O(d^{3/2} \log 1/\epsilon)$.

Disagreement coefficient: examples

- Thresholds in $\mathbb{R}$, any data distribution.

  $$\theta = 2.$$

  Label complexity $O(\log 1/\epsilon)$.

- Linear separators through the origin in $\mathbb{R}^d$, uniform data distribution. [Hanneke '07]

  $$\theta \leq \sqrt{d}.$$

  Label complexity $O(d^{3/2} \log 1/\epsilon)$.

- Linear separators in $\mathbb{R}^d$, smooth data density bounded away from zero. [Friedman '09]

  $$\theta \leq c(h^*)d$$

  where $c(h^*)$ is a constant depending on the target h^*.
  Label complexity $O(c(h^*)d^2 \log 1/\epsilon)$.

Label complexity: intuition

- Let's study the separable case (with CAL algorithm). Suppose $h^* \in H$ has zero error.

Label complexity: intuition

- Let's study the separable case (with CAL algorithm). Suppose $h^* \in H$ has zero error.
- After t points are seen, the effective version space H_t consists of classifiers with error at most about $\Delta_t = d/t$, where $d = \mathrm{VC}(H)$.

Label complexity: intuition

- Let's study the separable case (with CAL algorithm). Suppose $h^* \in H$ has zero error.
- After t points are seen, the effective version space H_t consists of classifiers with error at most about $\Delta_t = d/t$, where $d = \mathrm{VC}(H)$.
- Let $\mathrm{DIS}(H_t)$ be the part of the input space on which there is disagreeement within H_t. Points outside $\mathrm{DIS}(H_t)$ are not queried.

Label complexity: intuition

- Let's study the separable case (with CAL algorithm). Suppose $h^* \in H$ has zero error.
- After t points are seen, the effective version space H_t consists of classifiers with error at most about $\Delta_t = d/t$, where $d = \text{VC}(H)$.
- Let $\text{DIS}(H_t)$ be the part of the input space on which there is disagreeement within H_t. Points outside $\text{DIS}(H_t)$ are not queried.
- The expected number of queries, upto time T, is thus:

$$\sum_{t=1}^{T} \mathbb{P}(\text{DIS}(H_t)).$$

Label complexity: intuition

- Let's study the separable case (with CAL algorithm). Suppose $h^* \in H$ has zero error.
- After t points are seen, the effective version space H_t consists of classifiers with error at most about $\Delta_t = d/t$, where $d = \mathrm{VC}(H)$.
- Let $\mathrm{DIS}(H_t)$ be the part of the input space on which there is disagreeement within H_t. Points outside $\mathrm{DIS}(H_t)$ are not queried.
- The expected number of queries, upto time T, is thus:

$$\sum_{t=1}^{T} \mathbb{P}(\mathrm{DIS}(H_t)).$$

- Now, $\mathbb{P}(\mathrm{DIS}(H_t)) \leq \mathbb{P}(\mathrm{DIS}(B(h^*, \Delta_t))) \leq \theta\Delta_t \approx \theta d/t$.

Label complexity: intuition

- Let's study the separable case (with CAL algorithm). Suppose $h^* \in H$ has zero error.
- After t points are seen, the effective version space H_t consists of classifiers with error at most about $\Delta_t = d/t$, where $d = \mathrm{VC}(H)$.
- Let $\mathrm{DIS}(H_t)$ be the part of the input space on which there is disagreeement within H_t. Points outside $\mathrm{DIS}(H_t)$ are not queried.
- The expected number of queries, upto time T, is thus:

$$\sum_{t=1}^{T} \mathbb{P}(\mathrm{DIS}(H_t)).$$

- Now, $\mathbb{P}(\mathrm{DIS}(H_t)) \leq \mathbb{P}(\mathrm{DIS}(B(h^*, \Delta_t))) \leq \theta \Delta_t \approx \theta d/t$.
- Therefore, expected number of queries up to time T is roughly

$$\sum_{t=1}^{T} \frac{\theta d}{t} = d\theta \log T.$$

Label complexity: intuition

- Let's study the separable case (with CAL algorithm). Suppose $h^* \in H$ has zero error.
- After t points are seen, the effective version space H_t consists of classifiers with error at most about $\Delta_t = d/t$, where $d = \mathrm{VC}(H)$.
- Let $\mathrm{DIS}(H_t)$ be the part of the input space on which there is disagreeement within H_t. Points outside $\mathrm{DIS}(H_t)$ are not queried.
- The expected number of queries, upto time T, is thus:

$$\sum_{t=1}^{T} \mathbb{P}(\mathrm{DIS}(H_t)).$$

- Now, $\mathbb{P}(\mathrm{DIS}(H_t)) \leq \mathbb{P}(\mathrm{DIS}(B(h^*, \Delta_t))) \leq \theta \Delta_t \approx \theta d/t$.
- Therefore, expected number of queries up to time T is roughly

$$\sum_{t=1}^{T} \frac{\theta d}{t} = d\theta \log T.$$

- We need $T \approx d/\epsilon$ to get error ϵ.

Problems in scaling up to higher dimension

1. Loose bounds.
   These algorithms use generalization bounds to define the current version space.
2. Intractability of minimizing $0 - 1$ loss.
   Move to convex surrogate loss functions?

Three ways to manage sampling bias

1. Label everything.
2. Use importance weighting.
3. Explicitly manage sampling regions.

Importance weighting [Beygelzimer-D-Langford '09]

Standard classification setup with loss functions:

- Input space $\mathcal{X}$, label space $\mathcal{Y}$
- Hypotheses H map $\mathcal{X} \to \mathcal{Z}$
- Loss function $\ell : \mathcal{Z} \times \mathcal{Y} \to \mathbb{R}$
- Want $h \in H$ minimizing $L(h) = \mathbb{E}_{(X,Y)\sim\mathbb{P}} \ell(h(X), Y)$.

eg. $\mathcal{Y} = \{-1, 1\}$, $\mathcal{Z} = \mathbb{R}$, $\ell(z, y) = \ln(1 + e^{-yz})$.

Importance weighting [Beygelzimer-D-Langford '09]

Standard classification setup with loss functions:

- Input space $\mathcal{X}$, label space $\mathcal{Y}$
- Hypotheses H map $\mathcal{X} \to \mathcal{Z}$
- Loss function $\ell : \mathcal{Z} \times \mathcal{Y} \to \mathbb{R}$
- Want $h \in H$ minimizing $L(h) = \mathbb{E}_{(X,Y)\sim\mathbb{P}} \ell(h(X), Y)$.

eg. $\mathcal{Y} = \{-1, 1\}$, $\mathcal{Z} = \mathbb{R}$, $\ell(z, y) = \ln(1 + e^{-yz})$.

Importance-weighted active learning boilerplate:

1. Initialize $S = \emptyset$.
2. For $t = 1, 2, \ldots, T$:
   - Receive a new point x_t.
   - Choose a probability p_t.
   - With probability p_t: query label y_t and add $(x_t, y_t, 1/p_t)$ to S.
3. Return classifier $h \in H$ minimizing the weighted empirical loss

$$L_T(h) = \sum_{(x,y,w)\in S} w\, \ell(h(x), y).$$

Consistency of importance-weighted active learning

1. Initialize $S = \emptyset$.
2. For $t = 1, 2, \ldots, T$:
   - Receive a new point x_t.
   - Choose a probability p_t.
   - With probability p_t: query label y_t and add $(x_t, y_t, 1/p_t)$ to S.
3. Return the classifier $h \in H$ minimizing the weighted empirical loss $L_T(h) = \sum_{(x,y,w)\in S} w\, \ell(h(x), y)$.

Consistency of importance-weighted active learning

1. Initialize $S = \emptyset$.
2. For $t = 1, 2, \ldots, T$:
   - Receive a new point x_t.
   - Choose a probability p_t.
   - With probability p_t: query label y_t and add $(x_t, y_t, 1/p_t)$ to S.
3. Return the classifier $h \in H$ minimizing the weighted empirical loss $L_T(h) = \sum_{(x,y,w)\in S} w\,\ell(h(x), y)$.

Define $Q_t = 1(y_t \text{ is queried})$. Then $\mathbb{E}[Q_t|p_t] = p_t$, and

$$\mathbb{E}[L_T(h)] \;=\; \mathbb{E}\left[\sum_{t=1}^{T} \frac{Q_t}{p_t}\ell(h(x_t), y_t)\right] \;=\; \mathbb{E}_{(X,Y)\sim\mathbb{P}}\ell(h(X), Y) \;=\; L(h).$$

Consistency of importance-weighted active learning

1. Initialize $S = \emptyset$.
2. For $t = 1, 2, \ldots, T$:
   - Receive a new point x_t.
   - Choose a probability p_t.
   - With probability p_t: query label y_t and add $(x_t, y_t, 1/p_t)$ to S.
3. Return the classifier $h \in H$ minimizing the weighted empirical loss $L_T(h) = \sum_{(x,y,w)\in S} w\,\ell(h(x), y)$.

Define $Q_t = 1(y_t \text{ is queried})$. Then $\mathbb{E}[Q_t | p_t] = p_t$, and

$$\mathbb{E}[L_T(h)] \;=\; \mathbb{E}\left[\sum_{t=1}^{T} \frac{Q_t}{p_t} \ell(h(x_t), y_t)\right] \;=\; \mathbb{E}_{(X,Y)\sim\mathbb{P}} \ell(h(X), Y) \;=\; L(h).$$

If H is finite and all $p_t \geq p_{min}$, then with probability at least $1 - \delta$,

$$\max_{h\in H} |L_T(h) - L(h)| \;\leq\; \sqrt{\frac{2 \ln |H|/\delta}{T p_{min}}}.$$

One way to pick the sampling probabilities

- L_t^* = minimum empirical loss at time t, achieved by $h_t \in H$
- H_t = hypotheses $h \in H$ such that for all $t' < t$,

$$L_{t'}(h) \leq L_{t'}^* + \Delta_{t'}$$

where $\Delta_t \sim \sqrt{(\log |H|)/t}$ is from a generalization bound.

One way to pick the sampling probabilities

- L_t^* = minimum empirical loss at time t, achieved by $h_t \in H$
- H_t = hypotheses $h \in H$ such that for all $t' < t$,

$$L_{t'}(h) \leq L_{t'}^* + \Delta_{t'}$$

where $\Delta_t \sim \sqrt{(\log |H|)/t}$ is from a generalization bound.

Upon seeing example x_t, set the probability of sampling its label to

$$p_t \propto \max_{f,g \in H_t} \max_{y \in \mathcal{Y}} \ell(f(x_t), y) - \ell(g(x_t), y).$$

Linear separators, convex loss: this is a convex optimization.

One way to pick the sampling probabilities

- L_t^* = minimum empirical loss at time t, achieved by $h_t \in H$
- H_t = hypotheses $h \in H$ such that for all $t' < t$,

$$L_{t'}(h) \leq L_{t'}^* + \Delta_{t'}$$

where $\Delta_t \sim \sqrt{(\log |H|)/t}$ is from a generalization bound.

Upon seeing example x_t, set the probability of sampling its label to

$$p_t \;\propto\; \max_{f,g \in H_t} \max_{y \in \mathcal{Y}} \ell(f(x_t), y) - \ell(g(x_t), y).$$

Linear separators, convex loss: this is a convex optimization.

With probability at least $1 - \delta$, final hypothesis h_T satisfies

$$L(h_T) - L(h^*) \leq 2\Delta_{T-1}.$$

One way to pick the sampling probabilities

- $L_t^* =$ minimum empirical loss at time t, achieved by $h_t \in H$
- $H_t =$ hypotheses $h \in H$ such that for all $t' < t$,

$$L_{t'}(h) \leq L_{t'}^* + \Delta_{t'}$$

where $\Delta_t \sim \sqrt{(\log |H|)/t}$ is from a generalization bound.

Upon seeing example x_t, set the probability of sampling its label to

$$p_t \;\propto\; \max_{f,g \in H_t} \max_{y \in \mathcal{Y}} \ell(f(x_t), y) - \ell(g(x_t), y).$$

Linear separators, convex loss: this is a convex optimization.

With probability at least $1 - \delta$, final hypothesis h_T satisfies

$$L(h_T) - L(h^*) \leq 2\Delta_{T-1}.$$

Label complexity from disagreement-region considerations.

Moving towards a practical algorithm

The scheme described:

- Makes querying decisions based on (loose) generalization bounds.
- Requires two convex optimization procedures to be run for each data point that appears.

Moving towards a practical algorithm

The scheme described:

- Makes querying decisions based on (loose) generalization bounds.
- Requires two convex optimization procedures to be run for each data point that appears.

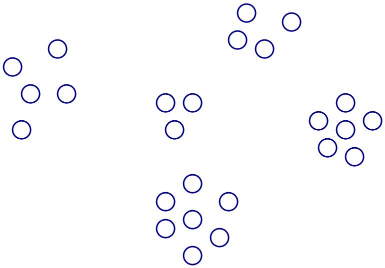
Very practical variant: Karampatziakis-Langford '11. But neither consistency nor label complexity has been characterized.

Three ways to manage sampling bias

1. Label everything.
2. Use importance weighting.
3. Explicitly manage sampling regions.

Exploiting cluster structure in data

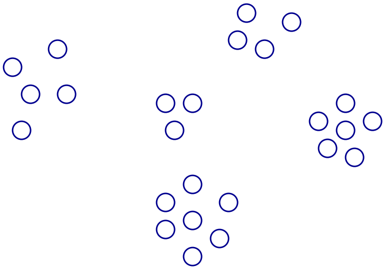
Suppose the unlabeled data looks like this.



Then perhaps we just need five labels.

Exploiting cluster structure in data

Suppose the unlabeled data looks like this.



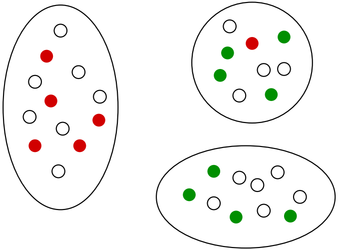
Then perhaps we just need five labels.

Challenges: In general, the cluster structure (i) is not so clearly defined and (ii) exists at many levels of granularity. And (iii) the clusters may not be pure in their labels.

Exploiting cluster structure in data [D-Hsu '08]

Basic primitive:

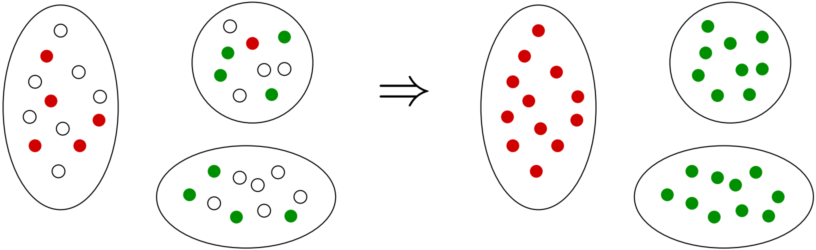
- Find a clustering of the data
- Sample a few *randomly-chosen* points in each cluster
- Assign each cluster its majority label
- Now use this fully labeled data set to build a classifier



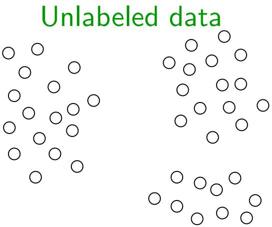
Exploiting cluster structure in data [D-Hsu '08]

Basic primitive:

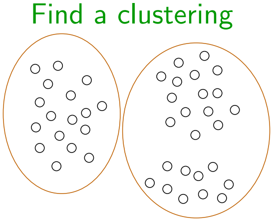
- Find a clustering of the data
- Sample a few *randomly-chosen* points in each cluster
- Assign each cluster its majority label
- Now use this fully labeled data set to build a classifier



Finding the right granularity

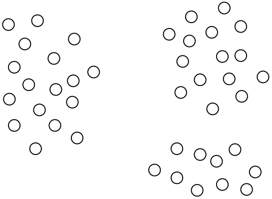


Finding the right granularity

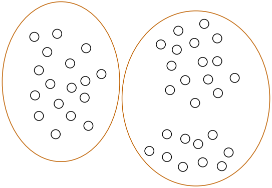


Finding the right granularity

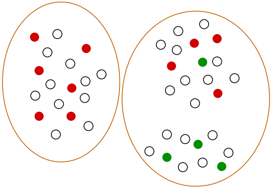
Unlabeled data



Find a clustering



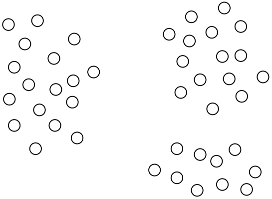
Ask for some labels



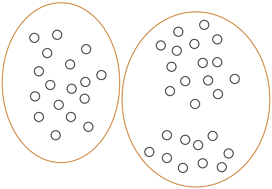
(random sampling within clusters)

Finding the right granularity

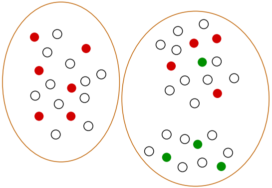
Unlabeled data



Find a clustering



Ask for some labels

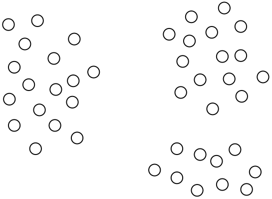


(random sampling within clusters)

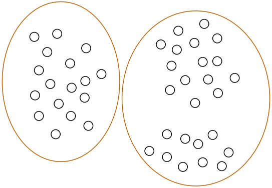
Now what?

Finding the right granularity

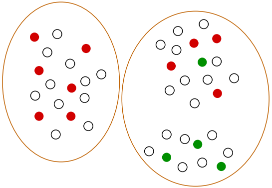
Unlabeled data



Find a clustering



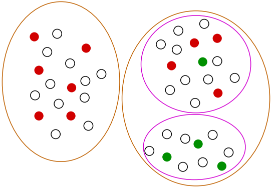
Ask for some labels



(random sampling within clusters)

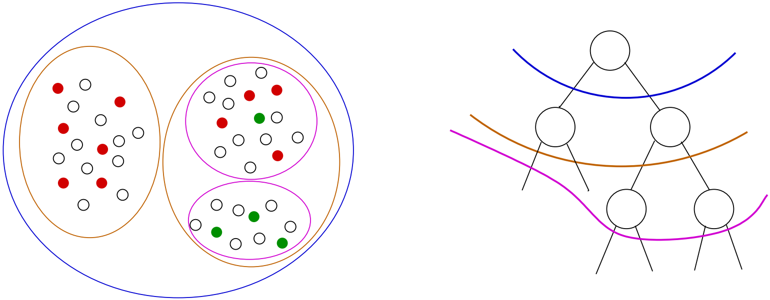
Now what?

Refine the clustering



Queried points are also randomly distributed within the new clusters.

Using a hierarchical clustering



Rules:

- Always work with some pruning of the hierarchy: a clustering induced by the tree.
- To make a query, pick a cluster, whereupon a random point in that cluster will be chosen and its label will be queried.
- As time progresses, the current pruning can only move down the tree, not back up.

Hierarchical sampling framework

So far we have described a framework for sampling that avoids bias. Still need to specify:

1. How the initial hierarchical clustering is built.
2. Rule for deciding which cluster to query.
3. Rule for deciding when to move down from a cluster to its children.

Hierarchical sampling framework

So far we have described a framework for sampling that avoids bias. Still need to specify:

1. How the initial hierarchical clustering is built.
2. Rule for deciding which cluster to query.
3. Rule for deciding when to move down from a cluster to its children.

D-Hsu '08:

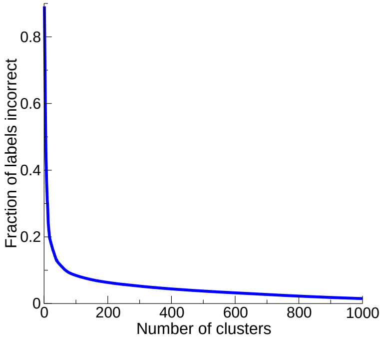
- Tree: Ward's agglomerative hierarchical clustering.
- Query least-pure cluster.
- Move down when confidence intervals indicate cluster's purity is below some threshold.

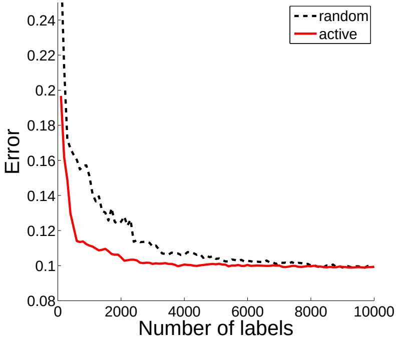
Urner-Wulff-Ben-David '12:

- Tree: k-d tree or RP tree.
- Query fixed number of points in each cluster.
- Move down if there is any disagreement in the labels obtained for a cluster.

Example: MNIST digits

Hierarchy built using Ward's agglomerative clustering (k-means cost function) with Euclidean distance.





Thanks

To my co-authors Alina Beygelzimer, Daniel Hsu, John Langford, and Claire Monteleoni.

And to the National Science Foundation for support under grants IIS-0347646, IIS-0713540, IIS-0812598, and CPS-0932403.