Boosting

prof. dr Arno Siebes

Algorithmic Data Analysis Group
Department of Information and Computing Sciences
Universiteit Utrecht

Relax

The previous time we relaxed the definition of PAC learning by giving up the requirement of uniformity

- this lead to non-uniform learning with the Structural Risk Minimization rule
- and the result that this can be done when the hypothesis class is the countable union of PAC learnable hypothesis classes

This time we loosen another constraint, viz.,

- the strong learning requirement
- and study the effect of this relaxation
  - can we learn a larger class of hypothesis classes by relaxing this constraint?
  - to which the answer is: no. As we will see at the end of today's lecture

Before we do this, first recall the definition of PAC learning

PAC Learnability

A hypothesis class $\mathcal{H}$ is agnostic PAC learnable with respect to a set Z and a loss function $l : Z \times \mathcal{H} \to \mathbb{R}_+$ if there exists a function $m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ and a learning algorithm A with the following property:

- for every $\epsilon, \delta \in (0,1)$
- for every distribution $\mathcal{D}$ over Z
- when running A on $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ i.i.d. samples generated by $\mathcal{D}$
- A returns a hypothesis $h \in \mathcal{H}$ such that with probability at least $1 - \delta$

$$L_{\mathcal{D}}(h) \leq \min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h') + \epsilon$$

Strong Learnability

PAC learnable is a strong requirement in that we require that

- for every $\epsilon > 0$ we get
- $L_{\mathcal{D}}(h) \leq \min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h') + \epsilon$

That is, we require that

- we can can arbitrarily close to optimal

This is pretty strong and may be

- practically unattainable

Recall that we discussed how k-DNF is (probably) hard

- unless $\mathbf{RP} = \mathbf{NP}$

This means that it is not realistic to assume that we can get

- arbitrarily close to optimal in practice

So, why not make our learning requirement weaker?

- perhaps this will allow us to learn a wider class of hypothesis classes in practice.

Weak Learning

If we want to relax the constraint of optimal result approximation

- the obvious question is how to relax it

In the familiar two class case

- coin tossing will give you a 50% score
  - for each individual distribution it can be wildly different, of course
  - but since for each $+1$ minded distribution there is the mirror image -1 preferring distribution
  - so, on average over all distributions we'll score 50/50

It seems eminently reasonable to require that

- our learning algorithm better than completely random guessing
- even if it is just a bit better

Such a learning algorithm is known as

- a *weak learner*

Weak PAC Learnable

A hypothesis class $\mathcal{H}$ is weak PAC learnable with respect to a set Z and a loss function $l : Z \times \mathcal{H} \to \mathbb{R}_+$ if there exists a function $m_{\mathcal{H}} : (0,1) \to \mathbb{N}$ and a learning algorithm A with the following property:

- for some $\gamma > 0$
- for every $\delta \in (0,1)$
- for every distribution $\mathcal{D}$ over Z
- when running A on $m \geq m_{\mathcal{H}}(\delta)$ i.i.d. samples generated by $\mathcal{D}$
- A returns a hypothesis $h \in \mathcal{H}$ such that with probability at least $1 - \delta$

$$L_{\mathcal{D}}(h) \leq \min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h') + \left(\frac{1}{2} - \gamma\right)$$

Or more colloquially:

$$\mathbb{P}\left(err(h) > \frac{1}{2} - \gamma\right) \leq \delta$$

Slightly More Precise

With agnostic PAC learning we already realized that $\mathcal{H}$ does not need to contain the true model. Still we were able to develop the theory

- without referencing this true model

Now, even more than before it is important to realize this. Hence, we could rework the definition of Weak PAC Learnable to emphasize this point, i.e., making the definition even more precise

- however, both for
  - consistency with the previous lectures
  - lowering the probability of confusing you
- we refrain from doing so

When necessary we will point to this true model explicitly.

Weak Learner Examples

Weak learners tend to be very simple, such as

- half (hyper)spaces, e.g.,
  - $x_i > c_i$ when the data lives in $\mathbb{R}^n$
  - note that this simply means that we use an axis-parallel hyperplane to distinguish the two classes
  - if the data is not completely random one should be able to find a hyperplane that does better than random
- more general are so-called *decision stumps*
  - one level decision/classification trees
  - that is a condition on a single attribute deciding between the two classes
  - again, if the data is not completely random one should be able to find one such condition that outperforms random

Note that in both cases it is in general

- unlikely that these classifiers would be very strong in any practical application

Intuition Sucks

Since in weak learning we are already satisfied with 51% accuracy rather than, say, 99%

- your intuition might be that it is easier to learn weakly
- not only in an algorithmic sense
- but also in the formal sense that there are hypothesis classes that can be learned weakly but not strongly
  - true models that we can approximate weakly but not strongly

And this latter intuition is

completely false

Boosting is a technique that turns weak classifiers into strong classifiers.

Weak and strong learning are equivalent.

Intuition by Committee

The intuition behind boosting is simple

- don't use one weak learner
- use many and let them vote on the result

The idea is simple,

- learn a weak learner
- note on which parts of the data is performs well
- and on which parts of the data it performs bad
- learn a new weak learner on the part where the previous one preforms badly

If one axis parallel hyperplane doesn't separate the classes

- a collection of such hyperplanes might
  - creating high dimensional "shoe boxes"
  - that circumscribe positive or negative areas

Choosing Data

Training the same (weak) classifier on the same data

- obviously gives you the same model over and over again

In other words we should give our learning algorithm

- slightly different data every time
- focussing on parts we do bad and less on parts were we are good

How we do that?

- By sampling from $D = \{x_1, y_1), \ldots, (x_m, y_m)\}$
- using a distribution that is adapted to misclassified data points every time we learn a new classifier

Note that this is related to but different from

- the bootstrap we discussed earlier

there are techniques that do rely on the bootstrap and its properties.

AdaBoost

$D_1((x_i, y_i)) := \frac{1}{m}$
For $t = 1$ to T

- sample a data set according to t
- get a weak hypothesis $h_t : \mathcal{X} \to \{0, 1\}$, minimizing

$$\epsilon_t = \mathbb{P}_{(x_i,y_i)\sim D_t}[h_t(x_i) \neq y_i] = \frac{1}{2} - \gamma_t$$

- $\alpha_t := \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
- for $i = 1$ to m

$$D_{t+1}((x_i, y_i)) = \frac{D_t((x_i, y_i))}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases}$$
$$= \frac{D_t((x_i, y_i))e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$$

Output: $H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

Some Notes

- Z_t is simply a normalizing term, ensuring that D_{t+1} is a probability distribution
  - $\sum_i D_{t+1}((x_i, y_i)) = 1$
- if $h_t(x_i) = y_i$ we lower the probability of (x_i, y_i)
- if $h_t(x_i) \neq y_i$ we raise the probability of (x_i, y_i)
- usually we assume the *weak learning hypothesis*
  - $\exists \gamma : \forall t : \gamma_t \geq \gamma > 0$
  - that is $\epsilon_t \leq \frac{1}{2} - \gamma$ for all t
  - with a finite number of rounds this is not a problem
  - limit situations require a proof for $\mathcal{H}$, it holds for all reasonable weak learners
- The most important remark
  - note that we construct a new hypothesis class!
  - we create strong learners from weak learners
  - they approximate the same true function
  - making the true function PAC learnable

How Good is AdaBoost?

So, we now have procedure to construct a new hypothesis from a given class of hypotheses

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Natural questions are:

- is H any better than the h_i?
- if it is better, how fast does it get better/converge?

That is, the natural question is

what is the (empirical) loss of H?

To answer this question we have to analyse AdaBoost

- we do that in a few steps
- before formulating and proving the general theorem

Unravelling D_{T+1}

Denote: $F(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$, i.e., $H(x) = sign(F(x))$. Then

$$
\begin{aligned}
D_{T+1}((x_i, y_i)) &= D_1((x_i, y_i)) \times \frac{e^{-\alpha_1 y_i h_1(x_i)}}{Z_1} \times \cdots \times \frac{e^{-\alpha_T y_i h_T(x_i)}}{Z_T} \\
&= \frac{D_1((x_i, y_i)) e^{-y_i \sum_{t=1}^{T} \alpha_t h_t(x)}}{\prod_{t=1}^{T} Z_t} \\
&= \frac{D_1((x_i, y_i)) e^{-y_i F(x_i)}}{\prod_{t=1}^{T} Z_t} \\
&= \frac{e^{-y_i F(x_i)}}{m \prod_{t=1}^{T} Z_t}
\end{aligned}
$$

A Note on Errors

Denote by $\mathbb{1}_A$ the indicator function on the set A, i.e.,

- $\mathbb{1}_A(x) = 1$ if $x \in A$
- $\mathbb{1}_A(x) = 0$ if $x \notin A$

More general for a condition ϕ, $\mathbb{1}_\phi = 1$ iff ϕ is true

Now note that

- if $H(x) \neq y$ one of the two is $+1$, the other is -1
- $H(x) = -1$ iff $F(x) \leq 0$
- hence $yF(x) \leq 0$
- and so $e^{-yF(x)} \geq 1$

Which means that

$$\mathbb{1}_{\{H(x_i) \neq y_i\}} \leq e^{-yF(x)}$$

Bounding the Training Error

For the error of H on the original data set, i.e., D_1 we have

$$\begin{aligned}
\mathbb{P}_{(x_i,y_i)\sim D_1}[H(x_i) \neq y_i] &= \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}_{\{H(x_i)\neq y_i\}} \\
&\leq \frac{1}{m}\sum_{i=1}^{m} e^{-y_i F(x_i)} \\
&= \frac{1}{m}\sum_{i=1}^{m} D_{T+1}((x_i, y_i)) m \prod_{t=1}^{T} Z_t \\
&= \sum_{i=1}^{m} D_{T+1}((x_i, y_i)) \prod_{t=1}^{T} Z_t \\
&= \prod_{t=1}^{T} Z_t \qquad D_{T+1} \text{ is a distribution}
\end{aligned}$$

Computing Z_t

$$
\begin{aligned}
Z_t &= \sum_{i=1}^{m} D_t((x_i, y_i))e^{-\alpha_t y_i h_t(x_i)} \\
&= \sum_{i:y_i=h_t(x_i)} D_t((x_i, y_i))e^{-\alpha_t} + \sum_{i:y_i \neq h_t(x_i)} D_t((x_i, y_i))e^{\alpha_t} \\
&= e^{-\alpha_t}(1-\epsilon_t) + e^{\alpha_t}\epsilon_t \\
&= e^{-\alpha_t}\left(\frac{1}{2}+\gamma_t\right) + e^{\alpha_t}\left(\frac{1}{2}-\gamma_t\right) \\
&= \sqrt{1-4\gamma_t^2}
\end{aligned}
$$

Note that our choice for α_t minimizes

- $e^{-\alpha_t}(1-\epsilon_t) + e^{\alpha_t}\epsilon_t$
- and thus Z_t
- and thus the empirical error.

AdaBoost is Good

From this series of partial results, we conclude:

$$\mathbb{P}_{(x_i,y_i)\sim D_1}[H(x_i) \neq y_i] \leq \prod_{t=1}^{T} \sqrt{1-4\gamma_t^2} \leq e^{-2\sum_{t=1}^{T}\gamma_t^2}$$

And if the weak learning assumption holds, we have:

$$\mathbb{P}_{(x_i,y_i)\sim D_1}[H(x_i) \neq y_i] \leq \left(\sqrt{1-4\gamma^2}\right)^T \leq e^{-2\gamma^2 T}$$

In other words, the error decreases exponentially

- at least for the error on the training set

The next question is, of course,

- what about the true error?

Towards a New Hypothesis Class

AdaBoost creates a new hypothesis class

- $\mathcal{H}$ is our *base class*

and the AdaBoost computes a weighted majority class

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

We can rewrite this as

$$H(x) = \sigma(h_1(x), \ldots, h_T(x))$$

where $\sigma : \mathbb{R}^T \to \{-1, 1\}$ is a linear threshold function of the form

$$\sigma(\vec{x}) = sign(\vec{w} \cdot \vec{x})$$

for some $\vec{w} \in \mathbb{R}^T$, like we encountered before. Denote the space of all such classifiers by Σ_T and recall that $VC(\Sigma_T) = T$ as we have also seen before.

The New Hypothesis Class

From this brief discussion we see that from a base class $\mathcal{H}$ AdaBoost creates a new class of hypotheses:

$$\mathcal{C}_T(\mathcal{H}) = \{x \to \sigma(h_1(x), \ldots, h_T(x)) \mid \sigma \in \Sigma_T, h_i \in \mathcal{H}\}$$

That is, to bound the true loss of AdaBoost

- we do it in terms of $\mathcal{C}_T(\mathcal{H})$
- rather than in terms of $\mathcal{H}$ directly

Moreover, note that since Σ_T is infinite. so is $\mathcal{C}_T(\mathcal{H})$

- whether or not $\mathcal{H}$ is finite or infinite

For the bounds, however, it still matters whether $\mathcal{H}$ is finite or infinite

The Finite Case

Let $D = \{x_1, \ldots, x_m\}$ with $m \geq T \geq 1$ Now consider a *fixed sequence* of base hypotheses

- $h_1, \ldots, h_T \in \mathcal{H}$

Then we map D to a new sample $D' \subset \mathbb{R}^T$ by

$$x \to (h_1(x), \ldots, h_T(x))$$

Since $VC(\Sigma_T) = T$ we have by Sauer's Lemma

$$\left|\tau_{\Sigma_T}(|D'|)\right| \leq \left(\frac{em}{T}\right)^T$$

Now the number choices for $h_1, \ldots, h_T$ is $|\mathcal{H}|^T$, hence

$$\left|\tau_{\mathcal{C}_T(\mathcal{H})}(|D|)\right| \leq \left|\tau_{\mathcal{C}_T(\mathcal{H})}(m)\right| \leq \left(\frac{em}{T}\right)^T |\mathcal{H}|^T$$

Bounding the Loss

After the proof of the Fundamental Theorem we discussed how the loss can be bounded using the growth function $\tau_{\mathcal{H}}$. Using this bound with the result on the previous slide, we have that

The true loss of the combined classifier with probability at least $1 - \delta$ for a D of size $m \geq T$ is bounded by

$$L_{\mathcal{D}}(H) \leq L_D(H) + \sqrt{\frac{32\left[\ln(em|\mathcal{H}|/T) + \ln(8/\delta)\right]}{m}}$$

And, if H is consistent with D (zero loss on the training set)

$$L_{\mathcal{D}}(H) \leq \frac{2T\log(em|\mathcal{H}|/T) + 2\log(2/\delta)}{m}$$

Convergence in Terms of m

If we take

$$T = \left\lceil \frac{\ln(m)}{2\gamma^2} \right\rceil$$

rounds, then under the weak learning assumption the training error of H is bounded by

$$e^{-2\gamma^2 T} < \frac{1}{m}$$

Since $|D| = m$, this means that $L_D(H) = 0$ and we have

$$L_{\mathcal{D}}(H) = O\left(\frac{1}{m}\left[\frac{\ln(m)(\ln(m) + \ln|\mathcal{H}|)}{\gamma^2} + \ln\frac{1}{\delta}\right]\right)$$

That is, in terms of the sample size m this bound converges to 0 as

$$O\left(\frac{(\ln(m))^2}{m}\right)$$

and can thus be made smaller than any $\epsilon > 0$

- moreover, it is polynomial in the parameters $1/\gamma, 1/\epsilon, 1/\delta$ and $\log|\mathcal{H}|$

Convergence in Terms of T

From our bounds, we have that

$$L_{\mathcal{D}}(H) \leq L_D(H) + \sqrt{\frac{32\left[\ln(em|\mathcal{H}|/T) + \ln(8/\delta)\right]}{m}}$$
$$\leq e^{-2\gamma^2 T} + O\left(\sqrt{\frac{T\ln(m|\mathcal{H}|/T) + \ln(1/\delta)}{m}}\right)$$

This is a function in T that

- has a (unique) global minimum for some $T > 0$

More precisely

- this function first decreases towards this global minimum
- and from then on increases

Too many rounds lead to overfitting, it seems best to run Adaboost until the error on the training set is 0

- although in practice running it a bit longer sometimes improves the result

The Infinite Case

Let $VC(\mathcal{H}) = d < \infty$, and $D = \{x_1, \ldots, x_m\}$ with $m \geq \max\{d, T\}$
Choose $\mathcal{H}' \subset \mathcal{H}$ such that

$$\forall h \in \mathcal{H}\, \exists h' \in \mathcal{H}' : \forall x_i\, h(x_i) = h'(x_i)$$

Since D is finite, so is $\mathcal{H}'$, in fact, by Sauer's Lemma

$$|\mathcal{H}'| = |\tau_{\mathcal{H}}(|D|)| \leq \left(\frac{em}{d}\right)^d$$

Similar to the finite case, we thus have:

$$\begin{aligned} \left|\tau_{\mathcal{C}_T(\mathcal{H})}(|D|)\right| &\leq \left(\frac{em}{T}\right)^T \left|\mathcal{H}'\right|^T \\ &\leq \left(\frac{em}{T}\right)^T \left(\frac{em}{d}\right)^{dT} \end{aligned}$$

Bounding the Loss

Similar to the finite case, we get

Let AdaBoost run T steps on a sample D of size m with base classifiers from $\mathcal{H}$ with $VC(\mathcal{H}) = d < \infty$. If $m \geq \max\{d, T\}$, then with probability of at least $1 - \delta$, the combined classifier H satisfies

$$L_{\mathcal{D}}(H) \leq L_D(H) + \sqrt{\frac{32[T \ln(em/T) + d \ln(em/d) + ln(8/\delta)}{m}}$$

And if $L_D(H) = 0$, then

$$L_{\mathcal{D}}(H) \leq \frac{2T(\log(2em/T) + d \log(2em/d) + 2 \log(2/\delta)}{m}$$

And under the weak learning assumption, when $T = \left\lceil \frac{\ln(m)}{2\gamma^2} \right\rceil$

$$L_{\mathcal{D}}(H) = O\left(\frac{1}{m}\left[\frac{\ln(m)}{\gamma^2}\left(\ln(m) + d \ln\left(\frac{m}{d}\right)\right) + \ln\left(\frac{1}{\delta}\right)\right]\right)$$

How About m?

In the finite case, we saw that error in terms of the sample size

- decreased as $O\left(\frac{(\ln(m))^2}{m}\right)$

and our error bound is polynomial in all the relevant parameters

- $1/\gamma, 1/\epsilon, 1/\delta$ and $\log|\mathcal{H}|$

That is, we see PAC Learning!

In the infinite case we have seen polynomial behaviour in the relevant parameters

- but we don't have a polynomial for the sample size

Unfortunately, that is a bit harder than for the finite case

- actually one can show that AdaBoost only looks at a fraction of the complete training set

For that reason, we will not discuss that analysis.

Weak vs Strong Learning

Clearly, strong learnability implies weak learnability

- your strong learner satisfies all the requirements of a weak learner

So, if the true model is strong learnable, it is also weak learnable.

Adaboost shows that this is also true the other way around

- we did not discuss all details, but still

If we have a weak learner for a true hypothesis

- Adaboost constructs a strong learner for it from a set of weak learners

That is,

weak learnable and strong learnable are equivalent!

Recapitulating

From our analysis of the problem of classification

- making no assumption on the distribution $\mathcal{D}$

We essentially "derived" the notion of

- Probably Approximately Correct Learning

And we learned that a hypothesis set is PAC learnable

- exactly when its VC dimension is finite

PAC learning has two constraints for which relaxing might increase the class of learnable hypotheses

- uniform learnability
- strong learnability

In both cases we have seen that even if we relax

- we remain essentially in the realm of PAC learning

  for distribution free learning, PAC learning is an eminently reasonable approach

But How Useful Is It?

That PAC learning is a reasonable approach is nice

- that is a like a conditio sine qua non
  - i.e., an essential condition

But that is not enough

- it should also be useful!

And that is what we are going to do next

- we are going to study PAC learning in the context of Frequent Itemset Mining

Frequent pattern mining is a class of techniques

- that has a wide variety of applications in a wide variety of domains

and it is a very good example

- of mining (very) Big Data