Boosting

Tirgul 10
Error Decomposition

Let $h_s$ be an $\text{ERM}_\mathcal{H}$ hypothesis. Then, we can write

$$L_D(h_s) = \epsilon_{\text{app}} + \epsilon_{\text{est}}$$

where: $\epsilon_{\text{app}} = \min_{h \in \mathcal{H}} L_D(h)$, $\epsilon_{\text{est}} = L_D(h_s) - \epsilon_{\text{app}}$

• **The Approximation Error** – the minimum risk achievable by a predictor in the hypothesis class.
  • Does not depend on the sample size
  • Determined by the hypothesis class chosen.
  • Enlarging the hypothesis class can decrease the approximation error.
  • Under the realizability assumption, the approximation error is zero.
Error Decomposition

Let $h_s$ be an $\text{ERM}_\mathcal{H}$ hypothesis. Then, we can write

$$L_D(h_s) = \epsilon_{app} + \epsilon_{est}$$

where: $\epsilon_{app} = \min_{h \in \mathcal{H}} L_D(h)$, $\epsilon_{est} = L_D(h_s) - \epsilon_{app}$

- **The Estimation Error** – the difference between the approximation error and the error achieved by the ERM predictor.
  - The estimation error results because the empirical risk (i.e., training error) is only an estimate of the true risk, and so the predictor minimizing the empirical risk is only an estimate of the predictor minimizing the true risk.
Error Decomposition

• Choosing $H$ to be a very rich class decreases the approximation error but at the same time might increase the estimation error, as a rich $H$ might lead to *overfitting*.

• Choosing $H$ to be a very small set reduces the estimation error but might increase the approximation error or, in other words, might lead to *underfitting*.
Error Decomposition

• **Goal:** minimize the total risk - $L_D(h_S) = \epsilon_{app} + \epsilon_{est}$

Bias- complexity tradeoff
Ultimate predictor – not always possible

• Sometimes designing the optimal (\textit{good enough}) model is a very complex task
  • Computationally hard/impossible
  • Limited resources
• Designing “weak” models – is not that hard (feasible task)
  • Prediction accuracy level low ($\epsilon$)
  • Sometimes it’s really off ($\delta$)
  • Slightly better than a flipping coin
• But what can we do with a weak learner?
Can we use “weak” learners to build a “good” classifier?

Wait... what is a “weak learner”?
Weak Learners

• Intuitively, a “weak” prediction model is an algorithm that uses some kind of simple rule-of-thumb (computationally simple). The model’s output is just a bit better than random output.

• Weak model+ Boosting = Good model
Boosting deals with the following issues:

1. Bias-Complexity Tradeoff
2. Computational Complexity
Issue #1: Bias-Complexity Tradeoff

• **Goal**: find the “correct” tradeoff between approximation and estimation errors.

• Solution concept:
  • Start with a limited hypothesis class
  • Increase as necessary
Issue #2: Computational Complexity

- Designing/Finding the “best” (ERM-wise) hypothesis can be computationally impossible or very hard.
- Boosting increases the accuracy rate of weak models → better prediction model
Reminder: PAC Learning

• A hypothesis class, $\mathcal{H}$, is PAC learnable if there exist $m_{\mathcal{H}} : (0,1)^2 \rightarrow \mathbb{N}$ and a learning algorithm with the following property:
  • For every $\varepsilon, \delta \in (0, 1)$
  • For every distribution $\mathcal{D}$ over $\mathcal{X}$
  • and for every labeling function $f : \mathcal{X} \rightarrow \{\pm 1\}$

• ...if the realizable assumption holds with respect to $f, \mathcal{H}, \mathcal{X}$, then when running the learning algorithm on $m \geq m_{\mathcal{H}} (\varepsilon, \delta)$ i.i.d examples generated by $\mathcal{D}$ and labeled by $f$, the algorithm returns a hypothesis $h$ such that, with probability of at least $1 - \delta$, $L_{\mathcal{D},f}(h) \leq \varepsilon$. 
Weak Learner

• A learning algorithm, $A$, is a $\gamma$-weak-learner for a hypothesis class, $\mathcal{H}$, if there exist $m_\mathcal{H} : (0, 1) \to \mathbb{N}$ and a learning algorithm with the following property:
  • For every $\delta \in (0, 1)$
  • For every distribution $\mathcal{D}$ over $\mathcal{X}$ and for every labeling function $f : \mathcal{X} \to \{\pm 1\}$ if the realizable assumption holds with respect to $f, \mathcal{H}, \mathcal{X}$, then when running the learning algorithm on $m \geq m_\mathcal{H}(\delta)$ i.i.d examples generated by $\mathcal{D}$ and labeled by $f$, the algorithm returns a hypothesis $h$ such that, with probability of at least $1 - \delta$, $L_{\mathcal{D},f}(h) \leq \frac{1}{2} - \gamma$.

• A hypothesis class $\mathcal{H}$ is $\gamma$-weak-learnable if there exists a $\gamma$-weak-learner for that class.
PAC vs. Weak learner

PAC (Strong learning)
• Strong learnability implies the ability to find an arbitrarily good classifier (with error rate at most $\varepsilon$ for an arbitrarily small $\varepsilon > 0$)

Weak learning
• We only need to output a hypothesis whose error rate is at most $1/2 - \gamma$, namely, whose error rate is slightly better than what a random labeling would give us.

*Assuming that it may be easier to come up with efficient weak learners than with efficient (full) PAC learners.
Weak Learner

• A possible approach: take a “simple” hypothesis class, denoted $B$, and apply ERM with respect to $B$ ($ERM_B$) as the weak learning algorithm.

• For this to work, we need to satisfy two requirements:
  • $ERM_B$ is efficiently implementable.
  • For every sample that is labeled by some hypothesis from $\mathcal{H}$, any $ERM_B$ hypothesis will have an error of at most $1/2 - \gamma$. 
Weak Learner

• The immediate question is whether we can boost an efficient weak learner into an efficient strong learner.
(ε, δ)-Weak-Learnability

• A class ℋ is (ε, δ)-weak-learnable if there exists a learning algorithm, A, and a training set size, m ∈ ℕ, such that for every distribution D over X and every f ∈ ℋ, with probability of at least 1 − δ,

\[ L_{D,f}(A(S)) \leq \epsilon. \]

• Remark:
  • Almost identical to PAC learning, but we only need to succeed for specific ε, δ.
Example #1: Weak Learning of 3-Piece Classifiers Using Decision Stumps

• Let $\mathcal{X} = \mathbb{R}$ and let $\mathcal{H}$ be the class of 3-piece classifiers, namely, $\mathcal{H} = \{h_{\theta_1, \theta_2, b}: \theta_1, \theta_2 \in \mathbb{R}, \theta_1 < \theta_2, b \in \{\pm 1\}\}$, where for every $x$:

$$h_{\theta_1, \theta_2, b}(x) = \begin{cases} 
+ b & \text{if } x < \theta_1 \text{ or } x > \theta_2 \\
- b & \text{if } \theta_1 \leq x \leq \theta_2
\end{cases}$$

An example hypothesis (for $b = 1$) is illustrated as follows:
Example #1: Weak Learning of 3-Piece Classifiers Using Decision Stumps

• Let $B$ be the class of Decision Stumps (2-pieces):
  $$B = \{ x \rightarrow \text{sign}(x - \theta) \cdot b : \theta \in \mathbb{R}, b \in \{\pm 1\}\}$$

• $b$ – the sign of each side
• $\theta$ - the threshold
Example #1: Weak Learning of 3-Piece Classifiers Using Decision Stumps

• Claim: $\text{ERM}_B$ is a $\gamma$-weak learner for $H$, for $\gamma = 1/12$.

• Proof:

$$D_1 + D_2 + D_3 = 1. \quad \exists i, D_i \leq \frac{1}{3}$$

Why?
Example #1: Weak Learning of 3-Piece Classifiers Using Decision Stumps

\[ L_{D,h}(b) \leq \frac{1}{3} \]

\forall 3 \text{ piece hypothesis } (h) \exists \text{ Decision Stumps } (b): L_{D,h}(b) \leq \frac{1}{3}
Example #1: Weak Learning of 3-Piece Classifiers Using Decision Stumps

Claim: There is a constant $m$, such that $\text{ERM}_B$ over $m$ examples is a $(5/12, 1/2)$-weak learner for $\mathcal{H}$
Boosting

• Suppose we have an \((\epsilon_0, \delta_0)\)-weak-learner algorithm, A, for some class \(\mathcal{H}\).

• Can we use A to construct a strong learner?

• If A is computationally efficient, can we boost it efficiently?

• Two questions:
  • Boosting the confidence - \(\delta\)
  • Boosting the accuracy - \(\epsilon\)
Boosting the confidence - $\delta$

Accuracy rate will decrease

$\epsilon_0 \rightarrow \epsilon_0 + \epsilon$
Boosting the confidence – the algorithm

• Suppose $A$ is an $(\epsilon_0, \delta_0)$-weak-learner algorithm, for some class $\mathcal{H}$, that requires $m_0$ examples.

• For any $\epsilon, \delta \in (0,1)$ we show how to learn $\mathcal{H}$ to accuracy $\epsilon_0 + \epsilon$ with confidence $\delta$.
  • $\implies$ In order to boost the confidence $\delta_0$, we are reducing the accuracy $\epsilon_0$ slightly.
    • The reduction is by a value $\epsilon$ (as small as you wish).
    • Loss will be $\epsilon_0 + \epsilon$, but with confidence $\delta_0$ (also as small as you wish!).
Boosting the confidence – the algorithm

• Suppose $A$ is an $(\epsilon_0, \delta_0)$-weak-learner algorithm, for some class $\mathcal{H}$, that requires $m_0$ examples.

• For any $\epsilon, \delta \in (0,1)$ we show how to learn $\mathcal{H}$ to accuracy $\epsilon_0 + \epsilon$ with confidence $\delta$.

• **Step 1:** Apply $A$ on $k = \left\lceil \frac{\log(\frac{2}{\delta})}{\log(\frac{1}{\delta_0})} \right\rceil$ i.i.d. samples, each with $m_0$ examples, to obtain $h_1, ... , h_k$. 

\[ \epsilon_0 \to \epsilon_0 + \epsilon \]
Boosting the confidence – the algorithm

• Suppose $A$ is an $(\epsilon_0, \delta_0)$-weak-learner algorithm, for some class $\mathcal{H}$, that requires $m_0$ examples.

• For any $\epsilon, \delta \in (0,1)$ we show how to learn $\mathcal{H}$ to accuracy $\epsilon_0 + \epsilon$ with confidence $\delta$.

• **Step 1:** Apply $A$ on $k = \left\lceil \frac{\log(2\delta)}{\log(1/\delta_0)} \right\rceil$ i.i.d. samples, each with $m_0$ examples, to obtain $h_1, \ldots, h_k$.

• **Step 2:** Take additional validation sample of size $|V| \geq \frac{2 \log(4k)}{\epsilon^2}$ and output $\hat{h} \in \arg\min_{h_i} L_V(h_i)$

Applying ERM on the weak learner

$\epsilon_0 \rightarrow \epsilon_0 + \epsilon$

$s_1, s_2, \ldots, s_k$

$wl, wl, \ldots, wl$

$h_1, h_2, \ldots, h_k$
Boosting the confidence

• **Claim:** with probability of at least $1 - \delta$, we have $L_D(\hat{h}) \leq \epsilon_0 + \epsilon$

• **Proof:**

1. The validation procedure guarantees: $P[L_D(\hat{h}) > \min_i L_D(h_i) + \epsilon] \leq \frac{\delta}{2}$
2. $P\left[\min_i L_D(h_i) > \epsilon_0\right] = P[\forall L_D(h_i) > \epsilon_0]$

\[
\leq \prod_{i=1}^{k} P[L_D(h_i) > \epsilon_0] \leq \delta_0^k \leq \frac{\delta}{2}
\]

Plugging in $k = \left\lceil \frac{\log(2/\delta)}{\log(1/\delta_0)} \right\rceil$
Boosting the confidence

• **Claim:** with probability of at least $1 - \delta$, we have $L_D(\hat{h}) \leq \epsilon_0 + \epsilon$

• **Proof:**

1. $P[L_D(\hat{h}) > \min_i L_D(h_i) + \epsilon] \leq \frac{\delta}{2}$  \hspace{1cm} \text{type 1 mistake}

2. $P\left[\min_i L_D(h_i) > \epsilon_0\right] \leq \frac{\delta}{2}$  \hspace{1cm} \text{type 2 mistake}

• Apply the union bound to conclude the proof: $P\left(\bigcup_i A_i\right) \leq \sum_i P(A_i)$.

• The probability that one of the mistakes would happen is $\leq \frac{\delta}{2} + \frac{\delta}{2} = \delta$. 

$\delta_0 \to \delta$
Boosting the confidence

**Claim:** with probability of at least $1 - \delta$, we have $L_D(\hat{h}) \leq \epsilon_0 + \epsilon$

**Proof:**

1. $P[L_D(\hat{h}) > \min_i L_D(h_i) + \epsilon] \leq \frac{\delta}{2}$ — type 1 mistake
2. $P[\min_i L_D(h_i) > \epsilon_0] \leq \frac{\delta}{2}$ — type 2 mistake

$P(\text{mistake}_1) \leq \frac{\delta}{2} + \frac{\delta}{2} = \delta$.

If both mistakes did not happen:

- $L_D(\hat{h}) \leq \min_i L_D(h_i) + \epsilon$
- $\min_i L_D(h_i) \leq \epsilon_0$

**Combined:** $L_D(\hat{h}) \leq \epsilon_0 + \epsilon$

$\Rightarrow$ i.e., with probability of at least $1 - \delta$, $L_D(\hat{h}) \leq \epsilon_0 + \epsilon$
Boosting the accuracy - $\epsilon$

ADA Boost
AdaBoost (Adaptive Boosting)

- **Input**: training set $S = (x_1, y_1), \ldots, (x_m, y_m)$, weak learner $WL$, number of rounds $T$

- **Initialize** $D^{(1)} = \left( \frac{1}{m}, \ldots, \frac{1}{m} \right)$ Vector of distributions over $S$. Initially a uniform distribution.
AdaBoost (Adaptive Boosting)

- **Input**: training set $S = (x_1, y_1), \ldots, (x_m, y_m)$, weak learner WL, number of rounds $T$
- **Initialize** $D^{(1)} = \left(\frac{1}{m}, \ldots, \frac{1}{m}\right)$
- **For** $t = 1, \ldots, T$
  - Invoke weak learner $h_t = WL(D^{(t)}, S)$
  - Compute $\epsilon_t = L_{D^{(t)}}(h_t) = \sum_{i=1}^{m} D^{(t)}_i \mathbb{I}_{[y_i \neq h_t(x_i)]}$

  *Invoke weak learner over $S$ with the distribution $D$ of this iteration.*

  *Calculating a weighted loss.*
AdaBoost (Adaptive Boosting)

- **Input**: training set $S = (x_1, y_1), \ldots, (x_m, y_m)$, weak learner WL, number of rounds $T$
- **Initialize** $D^{(1)} = \left(\frac{1}{m}, \ldots, \frac{1}{m}\right)$
- **For** $t = 1, \ldots, T$:
  - Invoke weak learner $h_t = WL(D^{(t)}, S)$
  - Compute $\epsilon_t = L_D^{(t)}(h_t) = \sum_{i=1}^{m} D_i^{(t)} \mathbb{I}[y_i \neq h_t(x_i)]$
  - Let $w_t = \frac{1}{2} \log \left(\frac{1}{\epsilon_t} - 1\right)$

  Assuming $h_t$ is a hypothesis given by a weak learner, the loss $\epsilon_t$ should be $< \frac{1}{2}$. Thus:
  - $\epsilon_t < \frac{1}{2} \rightarrow \frac{1}{\epsilon_t} > 2 \rightarrow \frac{1}{\epsilon_t} - 1 > 1$
  - Thus the log value is positive.
AdaBoost (Adaptive Boosting)

• **Input**: training set \( S = (x_1, y_1), \ldots, (x_m, y_m) \), weak learner WL, number of rounds \( T \)

• **Initialize** \( D^{(1)} = \left( \frac{1}{m}, \ldots, \frac{1}{m} \right) \)

• **For** \( t = 1, \ldots, T \):
  • Invoke weak learner \( h_t = WL(D^{(t)}, S) \)
  • Compute \( \epsilon_t = L_{D^{(t)}}(h_t) = \sum_{i=1}^{m} D^{(t)}_i \mathbb{I}_{[y_i \neq h_t(x_i)]} \)
  • Let \( w_t = \frac{1}{2} \log \left( \frac{1}{\epsilon_t} - 1 \right) \)
  • Update \( D^{(t+1)}_i = \frac{D^{(t)}_i \exp(-w_t y_i h_t(x_i))}{\sum_{j=1}^{m} D^{(t)}_j \exp(-w_t y_j h_t(x_j))} \) for all \( i = 1, \ldots, m \)

*Updating the distribution for the next iteration*
AdaBoost (Adaptive Boosting)

- Updating the distribution:

\[
D_i^{(t+1)} = \frac{D_i^{(t)} \exp(-w_t y_i h_t(x_i))}{\sum_{j=1}^{m} D_j^{(t)} \exp(-w_t y_j h_t(x_j))}
\]

- Denominator: sum over all possible \(i\)’s.
  - Normalizing the value so it sums up to 1 (because it’s a distribution).
AdaBoost (Adaptive Boosting)

- Updating the distribution:

\[
D_i^{(t+1)} = \frac{D_i^{(t)} \exp(-w_t y_i h_t(x_i))}{\sum_{j=1}^{m} D_j^{(t)} \exp(-w_t y_j h_t(x_j))}
\]

- Nominator:
  - \(y_i \in \{-1, 1\}, \ h_t(x_i) \in \{-1, 1\}\)
  
  \[
y_i h_t(x_i) = \begin{cases} 
1 & \text{prediction is correct} \\
-1 & \text{prediction incorrect}
\end{cases}
\]

\[
D_i^{(t+1)} = \begin{cases} 
D_i^{(t)} e^{-w_t} & \text{prediction is correct} \\
D_i^{(t)} e^{w_t} & \text{prediction incorrect}
\end{cases}
\]

Reducing the weight of examples predicted correctly.

Increasing the weight of examples predicted incorrectly, so that the WL will focus on them in the next iteration.
AdaBoost (Adaptive Boosting)

- **Input**: training set $S = (x_1, y_1), \ldots, (x_m, y_m)$, weak learner WL, number of rounds $T$

- **Initialize** $D^{(1)} = \left( \frac{1}{m}, \ldots, \frac{1}{m} \right)$

- **For** $t = 1, \ldots, T$:
  - Invoke weak learner $h_t = WL(D^{(t)}, S)$
  - Compute $\varepsilon_t = L_{D^{(t)}}(h_t) = \sum_{i=1}^{m} D_i^{(t)} \mathbb{I}[y_i \neq h_t(x_i)]$
  - Let $w_t = \frac{1}{2} \log \left( \frac{1}{\varepsilon_t} - 1 \right)$
  - Update $D_i^{(t+1)} = \frac{D_i^{(t)} \exp(-w_t y_i h_t(x_i))}{\sum_{j=1}^{m} D_j^{(t)} \exp(-w_t y_j h_t(x_j))}$ for all $i = 1, \ldots, m$

- **Output** the hypothesis $h_s(x) = \text{sign}(\sum_{t=1}^{T} w_t h_t(x))$. *A weighted vote on all weak hypotheses.*

$h_s$ denotes the “strong” hypothesis
AdaBoost

Initial uniform weight on training examples
Incorrect classifications re-weighted more heavily
Incorrect classification re-weighted more heavily

Final classifier is weighted combinations of weak classifiers:

\[ H(x) = \text{sign}(\alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x)) \]