Bagging & Boosting

Bias-Variance Decomposition

我们从一个数据集中生成很多小数据集用来训练和测试。它们得到的 loss 不同，我们可以求 loss 的期望 (数学期望（或均值）是试验中每次可能结果的概率乘以其结果的总和，它反映随机变量平均取值的大小) 来获得 loss 的均值。

Claim: $y_* = \mathbb{E}[t \mid \mathbf{x}]$ is the best possible prediction.

Proof:

$$\begin{aligned}
\mathbb{E}[(y-t)^2 \mid \mathbf{x}] &= \mathbb{E}[y^2 - 2yt + t^2 \mid \mathbf{x}] \\
&= y^2 - 2y\mathbb{E}[t \mid \mathbf{x}] + \mathbb{E}[t^2 \mid \mathbf{x}] \\
&= y^2 - 2y\mathbb{E}[t \mid \mathbf{x}] + \mathbb{E}[t \mid \mathbf{x}]^2 + \mathrm{Var}[t \mid \mathbf{x}] \\
&= y^2 - 2yy_* + y_*^2 + \mathrm{Var}[t \mid \mathbf{x}] \\
&= (y - y_*)^2 + \mathrm{Var}[t \mid \mathbf{x}]
\end{aligned}$$

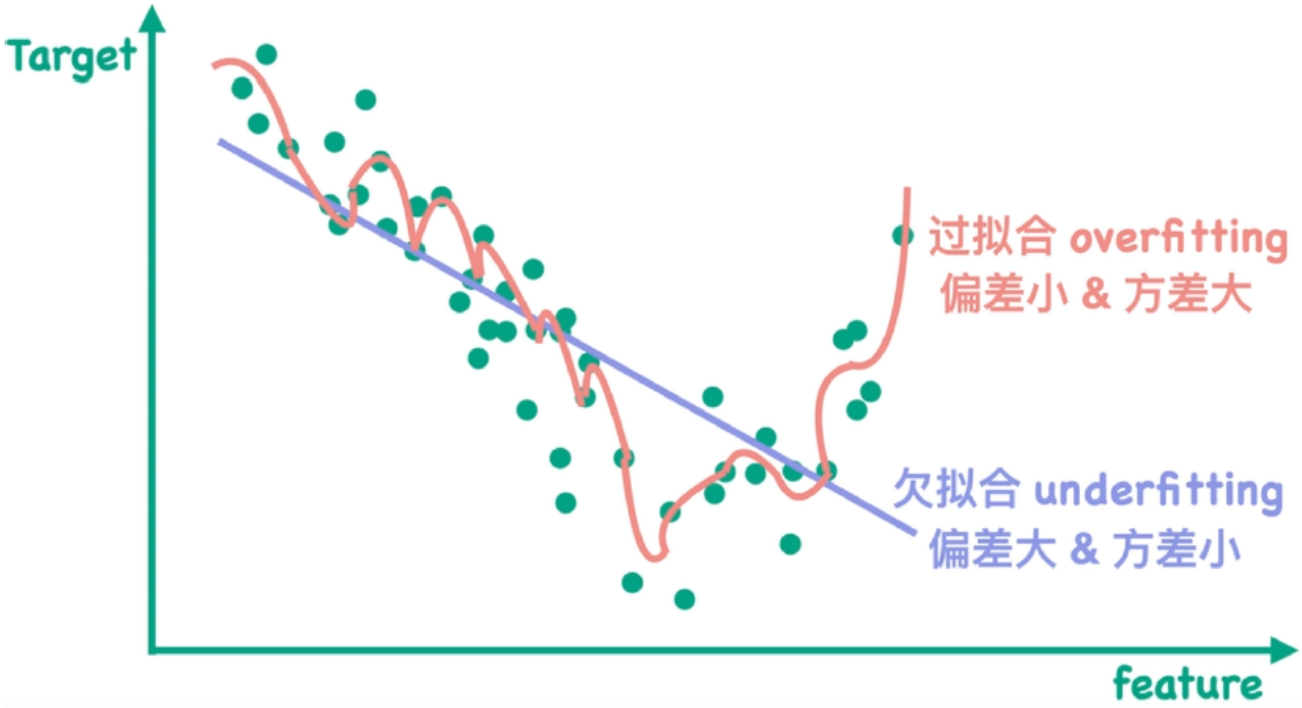
注：这是条件数学期望，$E[(y-t)^2|x]$ 意为在给定 x 的下的 loss。t 为真实值，y 为预测值。

- The first term is nonnegative, and can be made 0 by setting $y = y_*$.
- The second term corresponds to the inherent unpredictability, or noise, of the targets, and is called the Bayes error.
  - This is the best we can ever hope to do with any learning algorithm. An algorithm that achieves it is Bayes optimal.
  - Notice that this term doesn't depend on y.

现在我们可以得到 loss 的期望：

$$
\begin{aligned}
\mathbb{E}[(y-t)^2] &= \mathbb{E}[(y-y_\star)^2] + \mathrm{Var}(t) \\
&= \mathbb{E}[y_\star^2 - 2y_\star y + y^2] + \mathrm{Var}(t) \\
&= y_\star^2 - 2y_\star \mathbb{E}[y] + \mathbb{E}[y^2] + \mathrm{Var}(t) \\
&= y_\star^2 - 2y_\star \mathbb{E}[y] + \mathbb{E}[y]^2 + \mathrm{Var}(y) + \mathrm{Var}(t) \\
&= \underbrace{(y_\star - \mathbb{E}[y])^2}_{\text{bias}} + \underbrace{\mathrm{Var}(y)}_{\text{variance}} + \underbrace{\mathrm{Var}(t)}_{\text{Bayes error}}
\end{aligned}
$$

bias (偏差)：期望和实际结果的差距。高偏差意味着模型的准确率很差，即欠拟合。
variance (方差)：模型和期望的方差。高方差意味着模型的泛化能力不好，即过拟合。



Bagging

从数据集中生成 m 个子数据集，求出它们的预测值的均值 $y = \frac{1}{m}\sum_{i=1}^{m} y_i$。

这样的操作给 loss 的期望带来以下影响：

- **Bayes error: unchanged**, since we have no control over it
- **Bias: unchanged**, since the averaged prediction has the same expectation

$$\mathbb{E}[y] = \mathbb{E}\left[\frac{1}{m}\sum_{i=1}^{m} y_i\right] = \mathbb{E}[y_i]$$

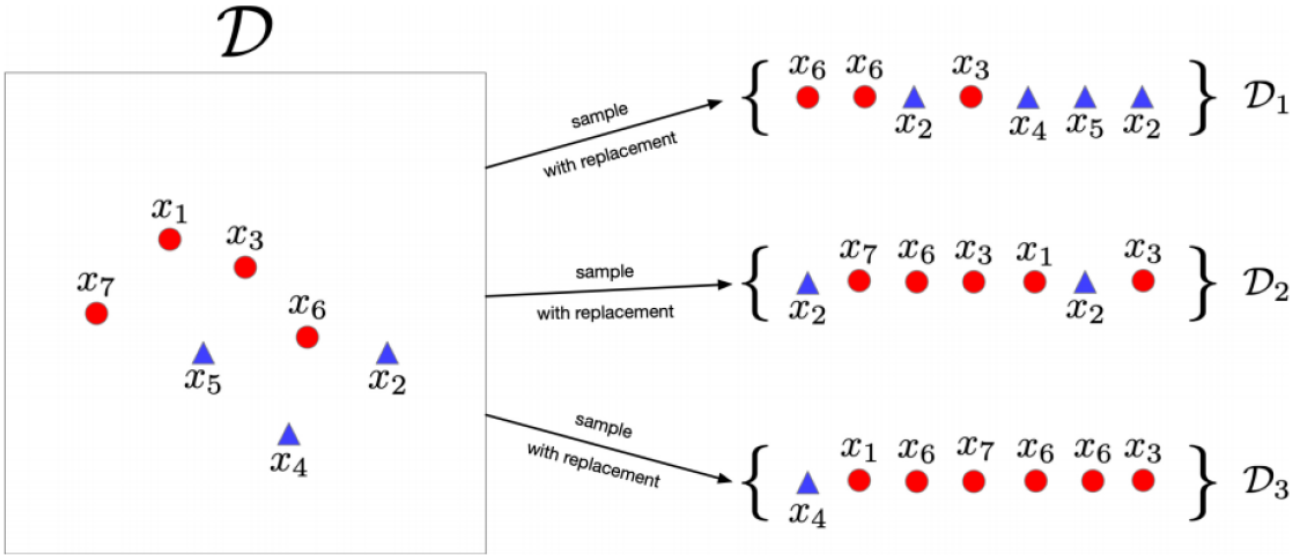
- **Variance: reduced**, since we're averaging over independent samples

$$\mathrm{Var}[y] = \mathrm{Var}\left[\frac{1}{m}\sum_{i=1}^{m} y_i\right] = \frac{1}{m^2}\sum_{i=1}^{m}\mathrm{Var}[y_i] = \frac{1}{m}\mathrm{Var}[y_i].$$

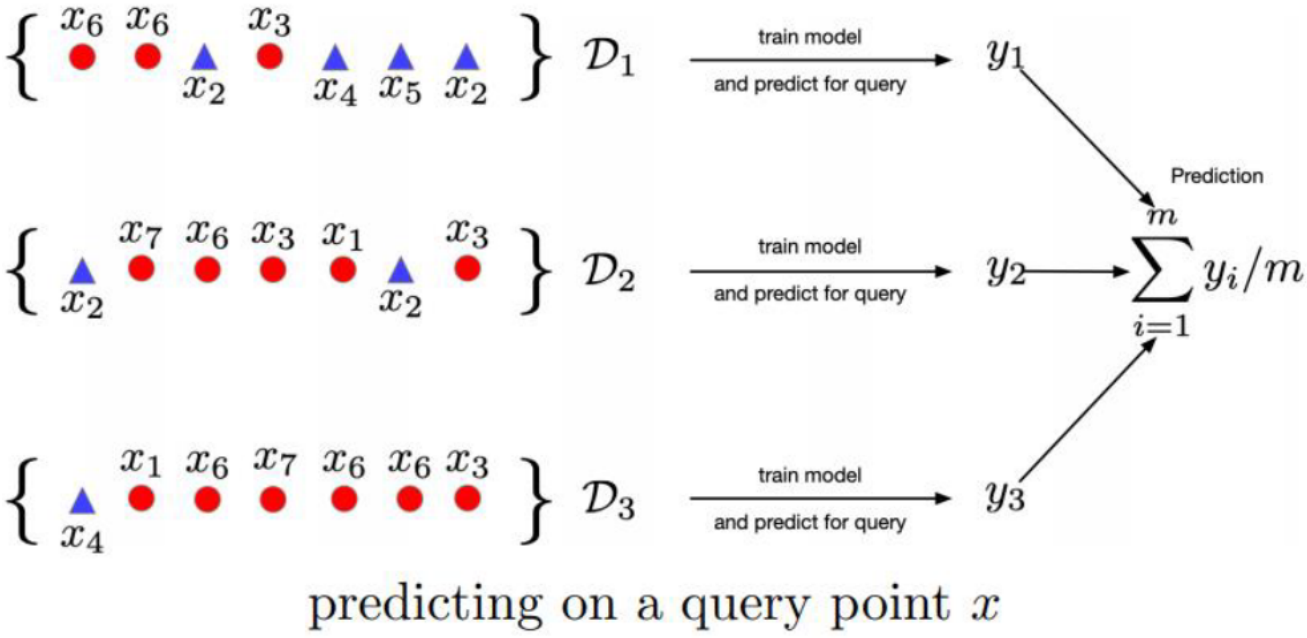
因此，bagging 的原理是通过这种操作降低 variance 来提升模型性能。

The Idea

对于一个有 n 个数据的数据集，我们根据它生成 m 个同样大小的数据集 (里面的数据可以重复)。之后在这些数据集上进行训练，最后求所有模型的预测的平均值。这种方法叫做 bootstrap aggregation or bagging。



in this example $n = 7$, $m = 3$



这样，我们就使用了均值 y 来充当最后的预测结果。

对于回归问题，bagging 使用均值来得到结果；对于分类问题，bagging 使用众数作为结果。

如果我们所有的数据集都是不相关的，我们可以极大的降低方差，但事实上并不是，因此降低的方差会变小。

- Problem: the datasets are not independent, so we don't get the $1/m$ variance reduction.
  - Possible to show that if the sampled predictions have variance σ^2 and correlation ρ, then

$$\mathrm{Var}\left(\frac{1}{m}\sum_{i=1}^{m} y_i\right) = \frac{1}{m}(1-\rho)\sigma^2 + \rho\sigma^2.$$

Random Forests

随机森林：随机选取数据集中的一部分数据和特征，对于这些数据构建决策树。重复多次，多个决策树构成随机森林。

Bagging Summary

- Bagging reduces overfitting by averaging predictions.
- Used in most competition winners
  - Even if a single model is great, a small ensemble usually helps.
- Limitations:
  - Does not reduce bias in case of squared error.
  - There is still correlation between classifiers.
    - Random forest solution: Add more randomness.
  - Naive mixture (all members weighted equally).
    - If members are very different (e.g., different algorithms, different data sources, etc.), we can often obtain better results by using a principled approach to weighted ensembling.
- Boosting, up next, can be viewed as an approach to weighted ensembling that strongly decorrelates ensemble members.

Boosting

和 bagging 不同，boosting 致力于使集成的模型不相关 (decorrelates)，即学习弱点。为此，它使用了加权的训练集。

Change cost function:

$$\sum_{n=1}^{N} \frac{1}{N} \mathbb{I}[h(x^{(n)}) \neq t^{(n)}] \quad \text{becomes} \quad \sum_{n=1}^{N} w^{(n)} \mathbb{I}[h(x^{(n)}) \neq t^{(n)}]$$

Usually require each $w^{(n)} > 0$ and $\sum_{n=1}^{N} w^{(n)} = 1$

这个意思是：对于上一个模型分错类的数据，我们要改变该数据的权重，使它在之后的训练中得到更多的重视。

Boosting 通过降低偏差来提升性能。

AdaBoost (Adaptive Boosting)

AdaBoost 算法：

对于一个 base classifier，训练它并进行测试，将错误的数据进行 re-weight，用来训练新的 classifier。这样训练多个 weak classifiers / learners，最后将这些 weak classifier 以适当的权重集成到最终的模型里。

对于 base classifier，需要最小化它的 weighted error；同时，由于最终的模型可能很大，因此我们需要 base classifier 训练起来足够快 (不太考虑模型的性能)，因此我们使用决策树桩 (decision stump) 而不是决策树来充当 base classifier。

Ada Boost Algorithm

- Input: Data $\mathcal{D}_N = \{\mathbf{x}^{(n)}, t^{(n)}\}_{n=1}^N$ where $t^{(n)} \in \{-1, +1\}$
  - This is different from previous lectures where we had $t^{(n)} \in \{0, +1\}$
  - It is for notational convenience, otw equivalent.
- A classifier or hypothesis $h : \mathbf{x} \to \{-1, +1\}$
- 0-1 loss: $\mathbb{I}[h(x^{(n)}) \neq t^{(n)}] = \frac{1}{2}(1 - h(x^{(n)}) \cdot t^{(n)})$

注：D_N 代表数据集有 N 个 feature (N 维)。最后的 0-1 loss 中，$h(x^{(n)})$ 和 $t^{(n)}$ 的结果都是 1 或 -1。因此 $\frac{1}{2}(1 - h(x^{(n)}) \cdot t^{(n)}))$ 的结果就为 0 和 1。

- Input: Data $\mathcal{D}_N$, weak classifier WeakLearn (a classification procedure that returns a classifier h, e.g. best decision stump, from a set of classifiers $\mathcal{H}$, e.g. all possible decision stumps), number of iterations T
- Output: Classifier $H(x)$
- Initialize sample weights: $w^{(n)} = \frac{1}{N}$ for $n = 1, \ldots, N$
- For $t = 1, \ldots, T$
  - Fit a classifier to weighted data ($h_t \leftarrow \text{WeakLearn}(\mathcal{D}_N, \mathbf{w})$), e.g.,

$$h_t \leftarrow \underset{h \in \mathcal{H}}{\text{argmin}} \sum_{n=1}^N w^{(n)} \mathbb{I}\{h(\mathbf{x}^{(n)}) \neq t^{(n)}\}$$

  - Compute weighted error $\text{err}_t = \frac{\sum_{n=1}^N w^{(n)} \mathbb{I}\{h_t(\mathbf{x}^{(n)}) \neq t^{(n)}\}}{\sum_{n=1}^N w^{(n)}}$
  - Compute classifier coefficient $\alpha_t = \frac{1}{2} \log \frac{1 - \text{err}_t}{\text{err}_t}$ $(\in (0, \infty))$
  - Update data weights

$$w^{(n)} \leftarrow w^{(n)} \exp\left(-\alpha_t t^{(n)} h_t(\mathbf{x}^{(n)})\right) \left[\equiv w^{(n)} \exp\left(2\alpha_t \mathbb{I}\{h_t(\mathbf{x}^{(n)}) \neq t^{(n)}\}\right)\right]$$

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

注：最终的模型为 H(x)。一开始，我们给数据中 N 个 feature 分配的权重 w 都一样 ($\frac{1}{N}$)。最后一共训练了 t 个 weak classifiers。

$h_t \leftarrow \underset{h \in \mathcal{H}}{\text{argmin}} \sum_{n=1}^N w^{(n)} \mathbb{I}\left\{h\left(\mathbf{x}^{(n)}\right) \neq t^{(n)}\right\}$ 代表用数据拟合出模型 h_t，argmin 代表使其 error 最小化。

$\text{err}_t = \frac{\sum_{n=1}^N w^{(n)} \mathbb{I}\left\{h_t\left(\mathbf{x}^{(n)}\right) \neq t^{(n)}\right\}}{\sum_{n=1}^N w^{(n)}}$ 中，上面是错误数据的权重和，下面是所有权重的和 (即 1)。

α_t 是当前模型在最终模型中占的权重，即模型的系数。

在更新数据权重的时候 $w^{(n)} \leftarrow w^{(n)} \exp\left(2\alpha_t \mathbb{I}\left\{h_t\left(\mathbf{x}^{(n)}\right) \neq t^{(n)}\right\}\right)$:

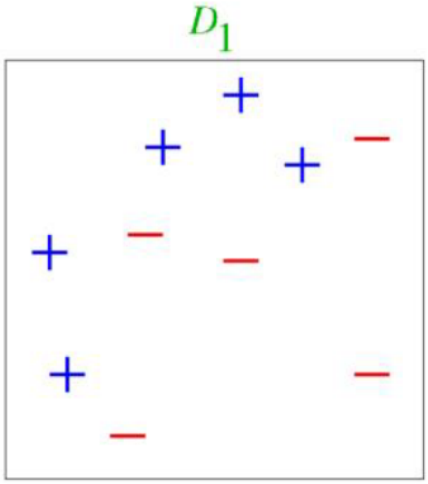
- 如果误差接近 0 ($err_t \approx 0$)，该模型的系数大，错误数据更受到重视

- 如果误差接近 0 ($err_t \approx 0.5$，这是二分类，0.5 就是啥都没干)，该模型的系数小，错误数据不受重视

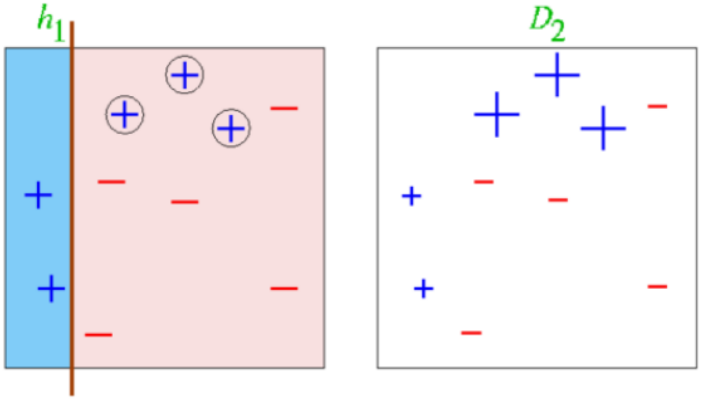
最后得到的最终模型，是所有加权 weak learners 的相加。而误差最小的 learner 会得到最高的权重。

Example

- Training data



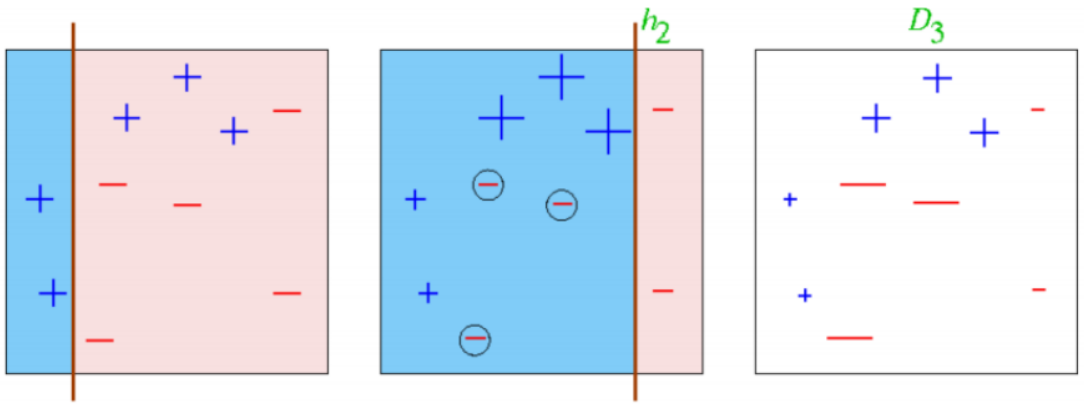
- Round 1



$$\mathbf{w} = \left(\frac{1}{10}, \ldots, \frac{1}{10}\right) \Rightarrow \text{Train a classifier (using } \mathbf{w}) \Rightarrow \text{err}_1 = \frac{\sum_{i=1}^{10} w_i \mathbb{I}\{h_1(\mathbf{x}^{(i)}) \neq t^{(i)}\}}{\sum_{i=1}^{N} w_i} = \frac{3}{10}$$

$$\Rightarrow \alpha_1 = \frac{1}{2} \log \frac{1 - \text{err}_1}{\text{err}_1} = \frac{1}{2} \log(\frac{1}{0.3} - 1) \approx 0.42 \Rightarrow H(\mathbf{x}) = \text{sign}\left(\alpha_1 h_1(\mathbf{x})\right)$$

- Round 2

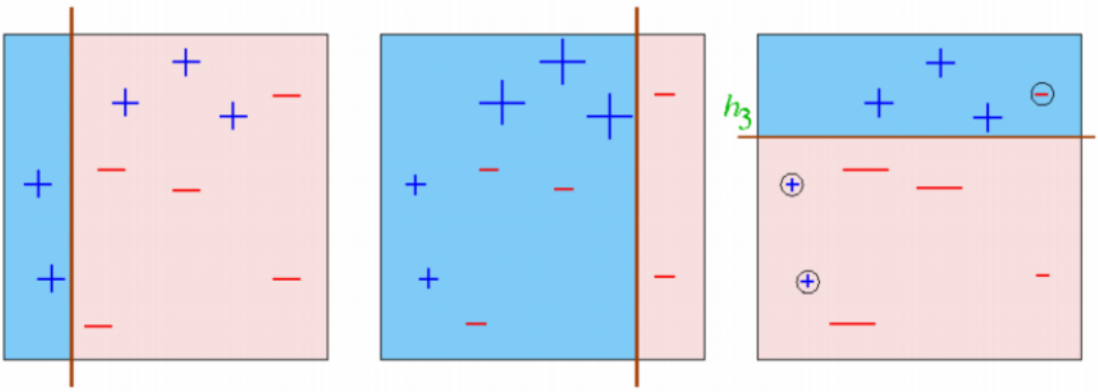


ε_2=0.21
α_2=0.65

$$\mathbf{w} = \text{updated weights} \Rightarrow \text{Train a classifier (using } \mathbf{w}) \Rightarrow \text{err}_2 = \frac{\sum_{i=1}^{10} w_i \mathbb{I}\{h_2(\mathbf{x}^{(i)}) \neq t^{(i)}\}}{\sum_{i=1}^{N} w_i} = 0.21$$

$$\Rightarrow \alpha_2 = \frac{1}{2} \log \frac{1 - \text{err}_3}{\text{err}_3} = \frac{1}{2} \log(\frac{1}{0.21} - 1) \approx 0.66 \Rightarrow H(\mathbf{x}) = \text{sign}\left(\alpha_1 h_1(\mathbf{x}) + \alpha_2 h_2(\mathbf{x})\right)$$

- Round 3

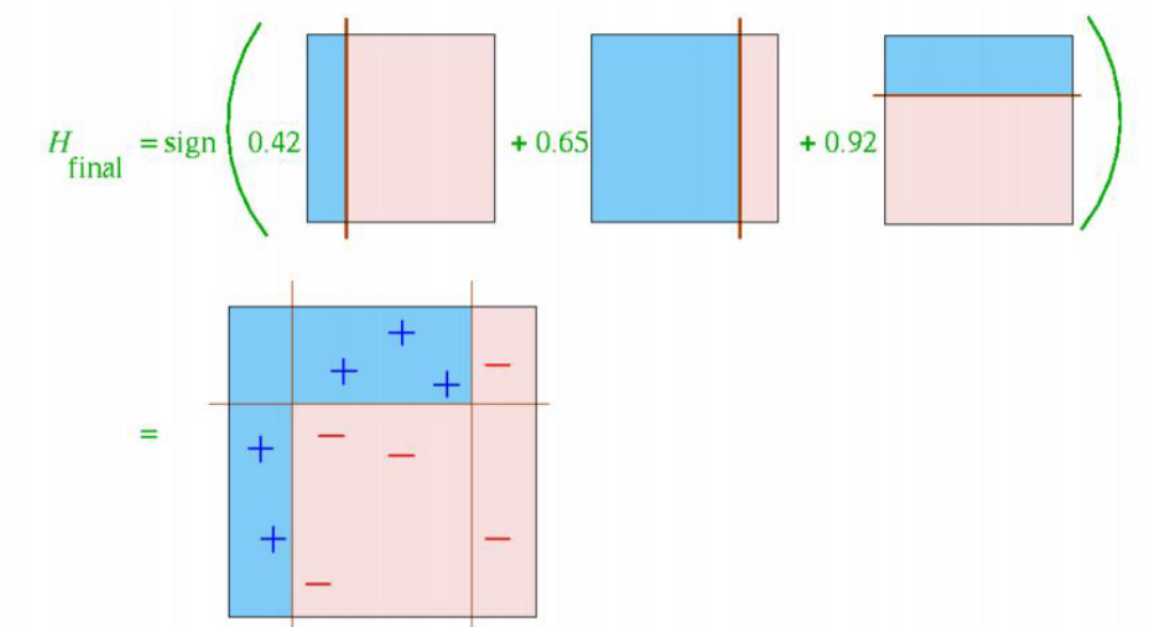


ε_3=0.14
α_3=0.92

$$\mathbf{w} = \text{updated weights} \Rightarrow \text{Train a classifier (using } \mathbf{w}) \Rightarrow \text{err}_3 = \frac{\sum_{i=1}^{10} w_i \mathbb{I}\{h_3(\mathbf{x}^{(i)}) \neq t^{(i)}\}}{\sum_{i=1}^{N} w_i} = 0.14$$

$$\Rightarrow \alpha_3 = \frac{1}{2} \log \frac{1 - \text{err}_3}{\text{err}_3} = \frac{1}{2} \log(\frac{1}{0.14} - 1) \approx 0.91 \Rightarrow H(\mathbf{x}) = \text{sign}\left(\alpha_1 h_1(\mathbf{x}) + \alpha_2 h_2(\mathbf{x}) + \alpha_3 h_3(\mathbf{x})\right)$$

- Final classifier



Minimizes the Training Error

Theorem

Assume that at each iteration of AdaBoost the WeakLearn returns a hypothesis with error $\text{err}_t \leq \frac{1}{2} - \gamma$ for all $t = 1, \ldots, T$ with $\gamma > 0$. The training error of the output hypothesis $H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$ is at most

$$L_N(H) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\{H(\mathbf{x}^{(i)}) \neq t^{(i)})\} \leq \exp\left(-2\gamma^2 T\right).$$

- This is under the simplifying assumption that each weak learner is γ-better than a random predictor.
- This is called geometric convergence. It is fast!

注：如果 classifier 的 error 是 $\frac{1}{2}$，那么它就啥都没做。所以 error 应该小于 $\frac{1}{2}$，而 γ 就是小了多少。

Additive Models (公式的推导，没用)

现在用另一种方式解释 AdaBoost。

- Consider a hypothesis class $\mathcal{H}$ with each $h_i : \mathbf{x} \mapsto \{-1, +1\}$ within $\mathcal{H}$, i.e., $h_i \in \mathcal{H}$. These are the "weak learners", and in this context they're also called **bases**.
- An **additive model** with m terms is given by

$$H_m(x) = \sum_{i=1}^{m} \alpha_i h_i(\mathbf{x}),$$

  where $(\alpha_1, \cdots, \alpha_m) \in \mathbb{R}^m$.
- Observe that we're taking a linear combination of base classifiers $h_i(\mathbf{x})$, just like in boosting.

注：这个实际上就是 boosting。

A greedy approach to fitting additive models, known as **stagewise training**:

1. Initialize $H_0(x) = 0$
2. For $m = 1$ to T:
   - Compute the m-th hypothesis $H_m = H_{m-1} + \alpha_m h_m$, i.e. h_m and α_m, assuming previous additive model H_{m-1} is fixed:

$$(h_m, \alpha_m) \leftarrow \underset{h \in \mathcal{H}, \alpha}{\operatorname{argmin}} \sum_{i=1}^{N} \mathcal{L}\left(H_{m-1}(\mathbf{x}^{(i)}) + \alpha h(\mathbf{x}^{(i)}),\ t^{(i)}\right)$$

   - Add it to the additive model

$$H_m = H_{m-1} + \alpha_m h_m$$

注：m 之前的 H 都已经固定了，这里通过求最小 loss 找到 h_m 和 α_m。

Consider the exponential loss

$$\mathcal{L}_{\mathrm{E}}(z,t) = \exp(-tz).$$

We want to see how the stagewise training of additive models can be done.

$$\begin{aligned}(h_m, \alpha_m) &\leftarrow \operatorname*{argmin}_{h\in\mathcal{H},\alpha} \sum_{i=1}^{N} \exp\left(-\left[H_{m-1}(\mathbf{x}^{(i)}) + \alpha h(\mathbf{x}^{(i)})\right] t^{(i)}\right) \\ &= \sum_{i=1}^{N} \exp\left(-H_{m-1}(\mathbf{x}^{(i)})t^{(i)}\right) \exp\left(-\alpha h(\mathbf{x}^{(i)})t^{(i)}\right) \\ &= \sum_{i=1}^{N} w_i^{(m)} \exp\left(-\alpha h(\mathbf{x}^{(i)})t^{(i)}\right).\end{aligned}$$

Here we defined $w_i^{(m)} \triangleq \exp\left(-H_{m-1}(\mathbf{x}^{(i)})t^{(i)}\right)$ (doesn't depend on h, α).

注：exponential loss 中 t 和 z 的值为 -1 或 1。因此，当预测准确时，结果是 -1，反之是 -1。e^{-1} 接近 0，而 e^{1} 很大。之后把 loss 带进去算，这里的 w 还是数据的权重。

- Now that we obtained h_m, we can plug it into our exponential loss objective (1) and solve for α_m.
- The derivation is a bit laborious and doesn't provide additional insight, so we skip it.
- We arrive at:

  $$\alpha_m = \frac{1}{2}\log\left(\frac{1-\mathrm{err}_m}{\mathrm{err}_m}\right),$$

  where err_m is the weighted classification error:

  $$\mathrm{err}_m = \frac{\sum_{i=1}^{N} w_i^{(m)} \mathbb{I}\{h_m(\mathbf{x}^{(i)}) \neq t^{(i)}\}}{\sum_{i=1}^{N} w_i^{(m)}}.$$

We can now find the updated weights for the next iteration:

$$\begin{aligned}w_i^{(m+1)} &= \exp\left(-H_m(\mathbf{x}^{(i)})t^{(i)}\right) \\ &= \exp\left(-\left[H_{m-1}(\mathbf{x}^{(i)}) + \alpha_m h_m(\mathbf{x}^{(i)})\right] t^{(i)}\right) \\ &= \exp\left(-H_{m-1}(\mathbf{x}^{(i)})t^{(i)}\right) \exp\left(-\alpha_m h_m(\mathbf{x}^{(i)})t^{(i)}\right) \\ &= w_i^{(m)} \exp\left(-\alpha_m h_m(\mathbf{x}^{(i)})t^{(i)}\right)\end{aligned}$$

To summarize, we obtain the additive model $H_m(x) = \sum_{i=1}^{m} \alpha_i h_i(\mathbf{x})$ with

$$h_m \leftarrow \operatorname*{argmin}_{h \in \mathcal{H}} \sum_{i=1}^{N} w_i^{(m)} \mathbb{I}\{h(\mathbf{x}^{(i)}) \neq t^{(i)}\},$$

$$\alpha = \frac{1}{2} \log \left(\frac{1 - \text{err}_m}{\text{err}_m} \right), \qquad \text{where } \text{err}_m = \frac{\sum_{i=1}^{N} w_i^{(m)} \mathbb{I}\{h_m(\mathbf{x}^{(i)}) \neq t^{(i)}\}}{\sum_{i=1}^{N} w_i^{(m)}},$$

$$w_i^{(m+1)} = w_i^{(m)} \exp \left(-\alpha_m h_m(\mathbf{x}^{(i)}) t^{(i)} \right).$$

We derived the AdaBoost algorithm!

Boosting summary

- Boosting reduces bias by generating an ensemble of weak classifiers.
- Each classifier is trained to reduce errors of previous ensemble.
- It is quite resilient to overfitting, though it can overfit.

一般的模型越复杂，越容易过拟合。但 boosting 中加入更多 weak classifiers，却不会过拟合。

Summary

- Ensembles combine classifiers to improve performance
- Boosting
  - Reduces bias
  - Increases variance (large ensemble can cause overfitting)
  - Sequential
  - High dependency between ensemble elements
- Bagging
  - Reduces variance (large ensemble can't cause overfitting)
  - Bias is not changed (much)
  - Parallel
  - Want to minimize correlation between ensemble elements.