Error Decomposition and Boosting

Wei Wang @ CSE, UNSW

April 11, 2020

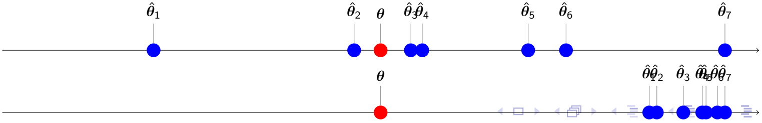
Overview

- Error decomposition
  - Bias-variance (+noise) decomposition
  - Estimation-approximation decomposition
- Adaboost:
  - We derive the Adaboosting algorithm (for classification) from the Gradient Boosting perspective.

Parameter Estimation

- By now, you should realize that we are essentially doing parameter estimation in Machine Learning. The two inputs are:
    - Parameterized function family: $f(\mathbf{x}; \boldsymbol{\theta})$
    - Training data: $\mathbf{X}$

  The output is $\hat{\boldsymbol{\theta}}$.
- Examples:
    - MLE
    - MAP
- How to evaluate the quality of the estimator $\hat{\boldsymbol{\theta}}$?
    - $\hat{\boldsymbol{\theta}}$ is a random variable wrt. $\mathbf{X}$!
    - Bias and Variance, i.e., $\mathbf{E}_{\mathbf{X}}\left[\!\left[\hat{\boldsymbol{\theta}}\right]\!\right]$ and $\mathbf{Var}_{\mathbf{X}}\left[\!\left[\hat{\boldsymbol{\theta}}\right]\!\right]$



Bias-Variance Tradeoff (on the Parameter)

- Estimated difference between a specific estimate and the true parameter:
  - Change of notation: D for $\mathbf{X}$, $\hat{\boldsymbol{\theta}} \stackrel{\text{def}}{=} h(D)$
  - Blue symbols are random variables.

$$
\begin{aligned}
&\mathbf{E}_D \left[\!\left[(h(D)-\theta))^2\right]\!\right] = \mathbf{E}_D \left[\!\left[(h(D)-\mu+\mu-\theta)^2\right]\!\right] \\
=&\mathbf{E}_D \left[\!\left[(h(D)-\mu)^2\right]\!\right] + \mathbf{E}_D \left[\!\left[(\mu-\theta)^2\right]\!\right] + 2\,\underbrace{\mathbf{E}_D \left[\!\left[(h(D)-\mu)\cdot(\mu-\theta)\right]\!\right]}_{\textit{cross term}} \\
=&\underbrace{\mathbf{E}_D \left[\!\left[(h(D)-\mathbf{E}_D\left[\!\left[(h(D)\right]\!\right])^2\right]\!\right]}_{\textit{variance}} + \underbrace{\mathbf{E}_D \left[\!\left[(\mathbf{E}_D\left[\!\left[(h(D)\right]\!\right]-\theta)^2\right]\!\right]}_{\textit{bias}}
\end{aligned}
$$

- Elimination of the cross term:

$$
\begin{aligned}
&\mathbf{E}_D \left[\!\left[(h(D)-\mu)\cdot(\mu-\theta)\right]\!\right] = (\mu-\theta)\cdot\mathbf{E}_D \left[\!\left[(h(D)-\mu)\right]\!\right] \\
=&(\mu-\theta)\cdot(\mathbf{E}_D \left[\!\left[(h(D)\right]\!\right]-\mu)
\end{aligned}
$$

  - Choose $\mu \stackrel{\text{def}}{=} \mathbf{E}_D \left[\!\left[(h(D)\right]\!\right]$ (not a r.v.), then this term vanishes

Understanding

$$\mathbf{E}_D \left[\!\left[(h(D) - \theta))^2 \right]\!\right] = \underbrace{\mathbf{E}_D \left[\!\left[(h(D) - \mu)^2 \right]\!\right]}_{\textit{variance}} + \underbrace{\mathbf{E}_D \left[\!\left[(\mu - \theta)^2 \right]\!\right]}_{\textit{bias}}$$

- μ is the ("long-term") expected parameter inferenced from training data
- Can you find approximately where μs are in the previous 1D examples?

Bias-Variance Decomposition for Regresssion

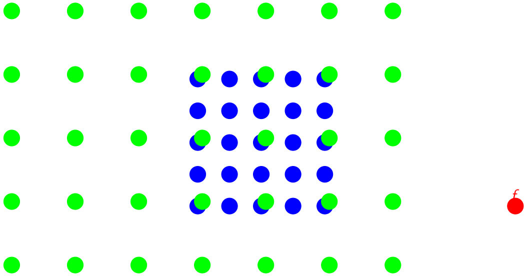
- Consider the regression with ℓ_2 loss: $\ell = (h - f)^2$.
  - Note that f and h are both functions, but we hide the input D.
  - Also note that summation is not needed, as the input is D.
- Consider its expectation.

$$
\begin{aligned}
\mathbf{E}\left[\!\left[\ell\right]\!\right] &= \mathbf{E}\left[\!\left[(h-f)^2\right]\!\right] = \mathbf{E}\left[\!\left[((h-\bar{h}) + (\bar{h}-f))^2\right]\!\right] \\
&= \mathbf{E}\left[\!\left[(h-\bar{h})^2\right]\!\right] + \mathbf{E}\left[\!\left[(\bar{h}-f)^2\right]\!\right] + \mathbf{E}\left[\!\left[2(h-\bar{h})\cdot(\bar{h}-f)\right]\!\right] \\
&= \underbrace{\mathbf{E}\left[\!\left[(h-\bar{h})^2\right]\!\right]}_{\textit{variance}} + \underbrace{\mathbf{E}\left[\!\left[(\bar{h}-f)^2\right]\!\right]}_{\textit{bias}}
\end{aligned}
$$

- Bias measures how much the prediction (averaged over all training data sets) differs from the desired regression function.
- Variance measures how much the predictions for individual training data sets vary around their average.

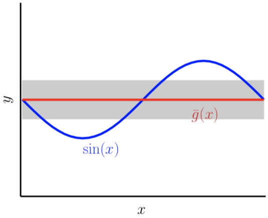
Understanding

- There is a trade-off between bias and variance. As we increase model complexity (from blue function family to green family)
  - bias decreases (more likely to fit the training data better) and
  - variance increases (the choice of the learned function varies more with training data)
- No such clearn decomposition for classification, but the trade-off still holds.

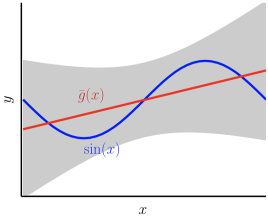


Example /1

$$\mathcal{H}_0 : y = c \text{ and } \mathcal{H}_1 : y = a \cdot x + b$$

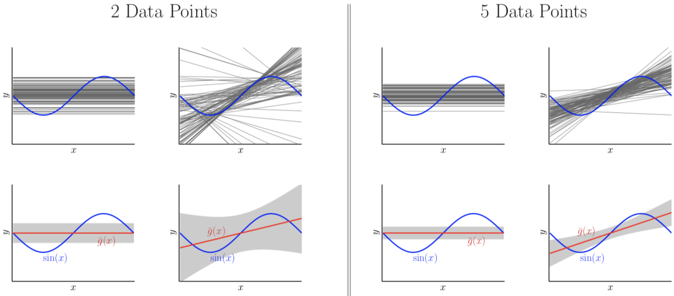


$\mathcal{H}_0$
bias $= 0.50$
var $= 0.25$
$E_{\text{out}} = 0.75$ ✓



$\mathcal{H}_1$
bias $= 0.21$
var $= 1.69$
$E_{\text{out}} = 1.90$

Example /2



$\mathcal{H}_0$	$\mathcal{H}_1$	$\mathcal{H}_0$	$\mathcal{H}_1$
bias $= 0.50$;	bias $= 0.21$;	bias $= 0.50$;	bias $= 0.21$;
var $= 0.25$.	var $= 1.69$.	var $= 0.1$.	var $= 0.21$.
$E_{\text{out}} = 0.75$ ✓	$E_{\text{out}} = 1.90$	$E_{\text{out}} = 0.6$	$E_{\text{out}} = 0.42$ ✓

The General Version

- Considering the noise, i.e.,

$$\hat{f}(\mathbf{x}) = f(\mathbf{x}) + \underbrace{\epsilon}_{noise}$$

then the decomposition can be decomposed as

$$\mathbf{E}\left[\!\left[(\hat{f} - f)^2\right]\!\right] = \mathbf{Var}\left[\!\left[\hat{f}\right]\!\right] + (\mathbf{E}\left[\!\left[\hat{f}\right]\!\right] - f)^2 + \underbrace{\mathbf{Var}\left[\!\left[\epsilon\right]\!\right]}_{noise}$$

- The additional noise component represents *irreducible error*.

Estimation-Approximation Error Decomposition

- Use linear regression as an example.

$$\mathbf{E}\left[\!\left[(\hat{f}-f)^2\right]\!\right] = \mathbf{E}\left[\!\left[((\hat{f}-\hat{f^*})+(\hat{f^*}-f))^2\right]\!\right]$$

$$= \mathbf{E}\left[\!\left[((\hat{f}-\hat{f^*})^2)\right]\!\right] + \mathbf{E}\left[\!\left[(\hat{f^*}-f)^2\right]\!\right] + 2\,\underbrace{\mathbf{E}\left[\!\left[(\hat{f}-\hat{f^*})\cdot(\hat{f^*}-f)\right]\!\right]}_{\textit{cross term}}$$

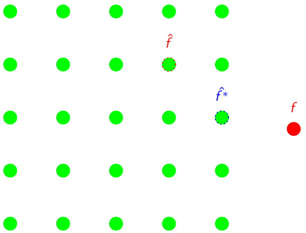
$$= \underbrace{\mathbf{E}\left[\!\left[(\hat{f}-\hat{f^*})^2\right]\!\right]}_{\text{estimation error}} + \underbrace{\mathbf{E}\left[\!\left[(\hat{f^*}-f)^2\right]\!\right]}_{\text{approximation error}}$$

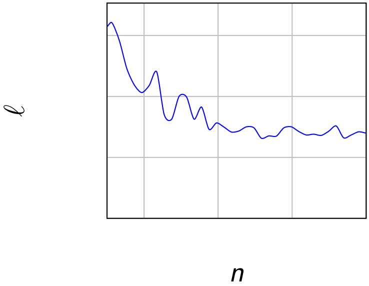
  - The expectation is wrt. the distribution from which training and test data are sampled.
  - The cross term must be 0 as the error of the best possible approximation $(\hat{f^*}-f)$ is independent of the error of any linear function of the data (hence $\hat{f}-\hat{f^*}$).

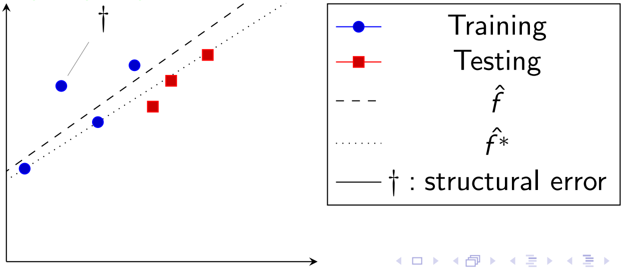
- General case

$$\ell_D^{0-1}[h_S] = \underbrace{\left(\ell_D^{0-1}[h_S] - \inf_{h\in\mathcal{H}} \ell_D^{0-1}[h]\right)}_{\text{Estimation Error in } \mathcal{H}} + \underbrace{\left(\inf_{h\in\mathcal{H}} \ell_D^{0-1}[h] - \ell_D^{0-1,*}\right)}_{\text{Approximation error of } \mathcal{H}} + \underbrace{\ell_D^{0-1,*}}_{\text{Irreducible Error}}$$

Example


$\hat{f}$
$\hat{f}^*$
f


ℓ
n


†
Training
Testing
$\hat{f}$
$\hat{f}^*$
† : structural error

Additive Model

- $H_t(\mathbf{x}) = \sum_{k=1}^{t} \alpha_k h_k(\mathbf{x})$
- Each h_i belong to a function family $\mathcal{H}$ (typically a weak functional class)
- Intuitively, we judiciously increase the complexity of the function class one step at a time (rather than going towards a very complex function class that will easily get overfitted).

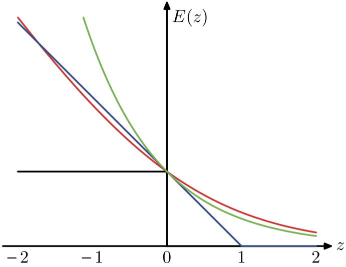
Naïve (Regression) Method

- Training data: $(\mathbf{x}_i, y_i)$
- Current model's prediction: $(\mathbf{x}_i, H_t(\mathbf{x}_i))$
- Learn a function $\alpha_{t+1} h_{t+1}()$ such that it approximates $y_i - H_t(\mathbf{x}_i)$ well.
    - A new regression problem with the effective training data: $(\mathbf{x}_i, y_i - H_t(\mathbf{x}_i))$
- What's the problem with this method?

Exponential Loss

- $0-1$ loss is hard to optimize directly.
- Exponential loss upper bounds the 0-1 loss:

$$\ell(H_t) = \sum_{k=1}^{n} \exp(-y_i H_t(\mathbf{x}_i))$$



- Black: 0-1 Loss
- Blue: Hinge Loss
- Red: Logistic Loss
- Green: Exponential Loss

From $H_t(\mathbf{x})$ to $H_{t+1}(\mathbf{x})$

$$\begin{aligned}
\ell_{t+1} &= \sum_{i=1}^{n} \exp(-y_i H_{t+1}(\mathbf{x}_i)) \\
&= \sum_{i=1}^{n} \exp(-y_i \left[H_t(\mathbf{x}_i) + \alpha_{t+1} h_{t+1}(\mathbf{x}_i)\right]) \\
&= \sum_{i=1}^{n} \exp(-y_i H_t(\mathbf{x}_i)) \cdot \exp(-y_i \alpha_{t+1} h_{t+1}(\mathbf{x}_i))) \\
&\overset{\text{def}}{=} \sum_{i=1}^{n} w_t(i) \cdot \exp(-y_i \alpha_{t+1} h_{t+1}(\mathbf{x}_i)))
\end{aligned}$$

- To minimize the loss ℓ_{t+1}, we have two “variables” to tune: $h_{t+1}()$ and α_{t+1} (note that $w_t(i)$ is a constant by now)

Choose $h_{t+1}()$ to minimize ℓ_{t+1}

We will omit the subscripts on α and $h()$

$$\begin{aligned}
\ell_{t+1} &= \sum_{i=1}^{n} w_t(i) \cdot \exp(-y_i \alpha h(\mathbf{x}_i)) \\
&= \sum_{i \mid y_i = h(\mathbf{x}_i)} w_t(i) \cdot e^{-\alpha} + \sum_{i \mid y_i \neq h(\mathbf{x}_i)} w_t(i) \cdot e^{\alpha} \\
&= \sum_{i \mid y_i = h(\mathbf{x}_i)} w_t(i) \cdot e^{-\alpha} + \sum_{i \mid y_i \neq h(\mathbf{x}_i)} w_t(i) \cdot e^{-\alpha} - \sum_{i \mid y_i \neq h(\mathbf{x}_i)} w_t(i) \cdot e^{-\alpha} + \sum_{i \mid y_i \neq h(\mathbf{x}_i)} w_t(i) \cdot e^{\alpha} \\
&= e^{-\alpha} \cdot \left(\sum_{i \mid y_i = h(\mathbf{x}_i)} w_t(i) + \sum_{i \mid y_i \neq h(\mathbf{x}_i)} w_t(i) \right) + (e^{\alpha} - e^{-\alpha}) \cdot \sum_{i \mid y_i \neq h(\mathbf{x}_i)} w_t(i) \\
&= e^{-\alpha} \cdot \sum_{i=1}^{n} w_t(i) + (e^{\alpha} - e^{-\alpha}) \cdot \sum_{i=1}^{n} w_t(i) \mathbf{1}[\![y_i \neq h(\mathbf{x}_i)]\!] \\
&= e^{-\alpha} + (e^{\alpha} - e^{-\alpha}) \cdot \sum_{i=1}^{n} w_t(i) \mathbf{1}[\![y_i \neq h(\mathbf{x}_i)]\!] \qquad (\dagger)
\end{aligned}$$

Minimization /1

$$\ell_{t+1} = e^{-\alpha} + (e^{\alpha} - e^{-\alpha}) \cdot \sum_{i=1}^{n} w_t(i) \mathbf{1}[\![y_i \neq h(\mathbf{x}_i)]\!] \qquad (\dagger)$$

- Choose h_{t+1} (within the function family) to minimize $\epsilon_t \overset{\text{def}}{=} \sum_{i=1}^{n} w_t(i) \mathbf{1}[\![y_i \neq h(\mathbf{x}_i)]\!]$.
  - $h_{t+1}()$ minimizes the weighted error.
- After $h_{t+1}()$ is learned (i.e., fixed), need to determine the best α_{t+1}:

$$\frac{\partial \ell_{t+1}}{\partial \alpha} = e^{\alpha} \epsilon_t + e^{-\alpha} \epsilon_t - e^{-\alpha}$$

Setting it to 0 give us:

$$\alpha_{t+1} = \frac{1}{2} \ln \frac{1 - \epsilon_t}{\epsilon_t}$$

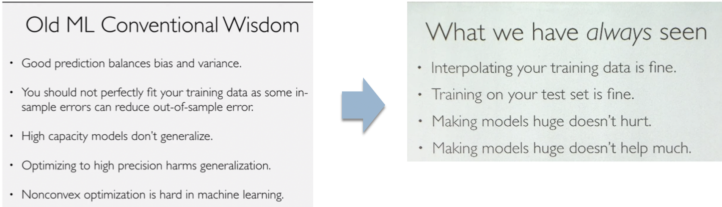
Renormlizing the Weight Distribution

- We need to ensure $w_{t+1}(i)$ is a distribution (that sums up to 1)

$$\begin{aligned}
w_{t+1}(i) &= \frac{\exp(-y_i H_{t+1}(\mathbf{x}_i))}{Z_{t+1}} && \text{(by definition)} \\
&= \frac{1}{Z_{t+1}} \cdot \exp(-y_i \left[H_t(\mathbf{x}_i) + \alpha_{t+1} h_{t+1}(\mathbf{x}_i)\right]) \\
&= \frac{1}{Z_{t+1}} \cdot (\exp(-y_i H_t(\mathbf{x}_i)) \;\cdot\; \exp(-y_i \alpha_{t+1} h_{t+1}(\mathbf{x}_i)))) \\
&= \frac{1}{Z_{t+1}} \cdot (w_t(i) \cdot \exp(-y_i \alpha_{t+1} h_{t+1}(\mathbf{x}_i)))
\end{aligned}$$

Comments

- This is just a theoretical explanation of the Adaboost algorithm. Exponential loss is not a good loss function anyway.
- Continuing to add weak classifiers after training error reaches 0 usually still increases the performance (due to the increase of the margin).
    - Similar phenomenon in Deep Learning.



Old ML Conventional Wisdom

- Good prediction balances bias and variance.
- You should not perfectly fit your training data as some in-sample errors can reduce out-of-sample error.
- High capacity models don't generalize.
- Optimizing to high precision harms generalization.
- Nonconvex optimization is hard in machine learning.

What we have *always* seen

- Interpolating your training data is fine.
- Training on your test set is fine.
- Making models huge doesn't hurt.
- Making models huge doesn't help much.

Source: Ben Recht

Adaboost

Algorithm 1: Adaboost($(\mathbf{X}, \mathbf{y})$, T)

```
$w_0(i) = \frac{1}{n}, \forall i \in [n]$;
for $t = 1$ to $T$ do
    $h_t \leftarrow$ the weak classifier (from $\mathcal{H}$) that minimizes weighted error
      (see $\epsilon_t$ below);                    /* $h_t(\mathbf{x}) \in \{-1, 1\}$ */;
    $\epsilon_t \leftarrow \frac{1}{n} \cdot \sum_{i=1}^{n} w_{t-1}(i) \cdot \mathbf{1}[\![H(\mathbf{x}_i) \neq y_i]\!]$;
    $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$;
    for $i = 1$ to $n$ do
        $w_t(i) \leftarrow w_{t-1}(i) \cdot e^{-\alpha_t [y_i h_t(\mathbf{x}_i)]}$;
    Renormalize $w_{t+1}(i)$ to be a distribution;
return $H_T(\mathbf{x}) \overset{\text{def}}{=} \operatorname{sgn}\left(\sum_{t=1}^{T} \alpha_t \cdot h_t(\mathbf{x})\right)$;
```

References

- Scott Fortmann-Roe. Understanding the Bias-Variance Tradeoff. http://scott.fortmann-roe.com/docs/BiasVariance.html
- M. Magdon-Ismail. Approximation Versus Generalization. http://www.cs.rpi.edu/~magdon/courses/LFD-Slides/SlidesLect07.pdf
- Boris Babenko. Note: A Derivation of Discrete AdaBoost. http://vision.ucsd.edu/~bbabenko/data/boosting_note.pdf.