Lecturer: Prof. Homrighausen | Scribe: Ryan Hicks

This document will pick up from slide 29 of Boosting 1 and continue through slide 31 of Boosting 2.

§Functional Gradient Descent

We have previously seen that if we have a squared error loss function l and want to minimize the risk $R(f)$ over f, the minimizer is $\mathbb{P}(Y|X)$. We can generalize this result to any loss function and to be able to work with actual data by using

$$\hat{g}(X_i) = \frac{\partial l(f(X_i), Y_i)}{\partial f}$$

This procedure simply walks in the direction of the negative gradient of the risk. However, it overfits and is only defined at the observed X_i. We can prevent the overfitting by restricting the subspace of functions we are interested in. Here is an improved algorithm:

For $b = 1, ..., B$, do:

1. $R_i \leftarrow -\hat{g}(X_i)\Big|_{f=\hat{f}_{b-1}} = \frac{\partial \ell(f(X_i), Y_i)}{\partial f}\Big|_{f=\hat{f}_{b-1}}$
2. $\hat{f} \leftarrow \arg\min_{f \in \mathcal{F}} ||R - f||_2^2$
   This is the projection step, allowing for $\hat{f}$ to be defined at new X. This is crucial, as it allows us to reduce overfitting and interpolate new X_i.
3. Update: $\hat{f}_b \leftarrow \hat{f}_{b-1} + \lambda \hat{f}$

If we compare this algorithm to boosting:

1. Fit $\hat{f}_b$ with $M + 1$ regions to $\tilde{\mathcal{D}} = \{(X_1, R_1), \ldots, (X_n, R_n)\}$
2. Update: $\hat{f} \leftarrow \hat{f} + \lambda \hat{f}_b$
3. Update: $R \leftarrow R - \hat{f}$

..., we notice that they are the same. Boosting is simply an algorithmic form of fitting a general additive model.

§Additive Models

Despite the example previously with the squared error loss function, it does not make sense to use with classification. This is because the quadratic term will penalize classifications that are "too" correct (AdaBoost, introduced below, uses an exponential loss function that does not have this problem). The most common approach is additive logistic regression.

Suppose $Y \in \{-1, 1\}$

$$\log\left(\frac{\mathbb{P}(Y=1|X)}{\mathbb{P}(Y=-1|X)}\right) = \sum_{j=1}^{p} h_j(x_j) = h(X)$$

We can invert this equation to get a probability estimate

$$\pi(X) = \mathbb{P}(Y=1|X) = \frac{e^{h(X)}}{1+e^{h(X)}}$$

Models of this type can be fit by maximizing the binomial likelihood (numerically, not analytically). In R, additive logistic regression models can be fit with the package gam.

§AdaBoost

AdaBoost is a powerful and popular boosted classifier. There are two primary forms, Discrete and Real AdaBoost, though they tend to give similar results. Both can be thought of as a stage-wise estimation procedure for fitting additive logistic regression models. In general, the method trains the classifier as usual, then at each step increases the weight of misclassifications and decreases the weight of correct classifications. In detail, here is the algorithm for Real AdaBoost:

1. Initialize $w_i \equiv 1/n$
2. For $b = 1, \ldots, B$
   (a) Fit the classifier on $\mathcal{D}$, weighted by w_i and produce $p_b(X) = \hat{P}_w(Y=1|X)$
   (b) Set $h_b(X) \leftarrow \{\frac{1}{2}\log(p_b(X)/(1-p_b(X)))$
   (c) Set $w_i \leftarrow w_i \exp\{-Y_i h_b(X_i)\}$
3. Output: $g(X) = \text{sgn}\left(\sum_{b=1}^{B} h_b(X)\right)$

B is a minor tuning parameter that needs to be chosen, but it is relatively insensitive. However, if increased too much, it can make the approximation worse. Real AdaBoost uses the class probability estimates to determine the contribution of the b^{th} classifier. This is in contrast to the estimated label of Discrete AdaBoost. However, as mentioned before, they tend to yield similar results.