Ensemble methods

Kristian Wichmann

October 20, 2017

Ensemble learning in machine learning is broadly speaking the practice of making a collection - an *ensemble* - of models, and combine them into a single model with more desired properties.

1 Bagging and boosting

There are two broad categories of such meta-algorithms:

1.1 Bagging

Bagging is short for *bootstrap aggregation*. This is because the ensembles are made through bootstrapping, i.e. random resampling from the training set. Properties of these meta-algorithms:

- Ensembles are build independently in *parallel*.
- Tends to reduce variance.

1.2 Boosting

Properties of these meta-algorithms:

- Ensembles are build *sequentially*, each building on the ones before.
- Tends to reduce bias.

2 Simple, unweighted boosting

As noted above, the general idea behind boosting is to combine an ensemble of weak learners into one strong learner. Here, the term *weak learner* means one that is slightly better than random guessing.

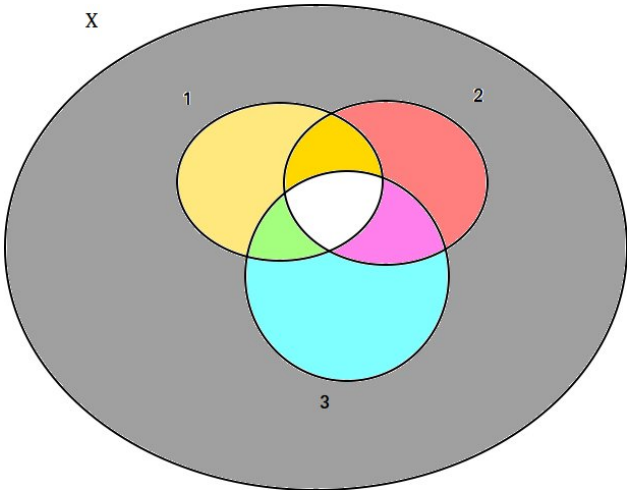

Figure 1: Regions of error for the binary classification with three learners.

2.1 Wisdom of the crowd

The most basic type of boosting is known from the quiz show "Who wants to be a Millionaire?". One of the lifelines is asking the audience. Here, a simple majority vote determines what is usually the best answer. This phenomenon is also known as the *wisdom of the crowd*.

The procedure above describes a classification problem. For regression type problems, the mean of the individual predictions would be used instead of majority vote.

2.2 Example: Binary classification with three learners

As a simple example consider three binary classifiers h_1, h_2, h_3 on a training set X. The two classes corresponds to values of ± 1. The true labels are denoted $f(x)$. Then the simple boosting classifier as described above can be written like:

$$H(x) = \operatorname{sgn}\left[h_1(x) + h_2(x) + h_3(x)\right] \tag{2.1}$$

The situation is visualized in figure 1. Each of the three initial classifiers has a region in which its prediction is wrong. These are shown as the regions 1, 2, and 3 respectively. The boosted classifier is only wrong when two or all of the three initial classifiers are wrong. In other words, the error region for H consists of the union of the four regions in orange, green, purple, and white.

Now, initially this looks like an improvement, but in fact in needs not be! Imagine that there's one data point in each of the four regions mentioned

above, and none in the yellow, red and blue ones. Then each of h_1, h_2, and h_3 misclassify three points, while H actually misclassifies four!

3 AdaBoost

As the example above shows, we need to be more clever to ensure a good boosting strategy. *AdaBoost*, which is short for *adaptive boosting* is one such widely used algorithm.

3.1 Weighted binary classification

Consider equation 2.1, but for T learner instead of three. We can generalize this by including *weights* for each classifier:

$$H(x) = \text{sgn}\left[\sum_{t=1}^{T} \alpha_t h_t(x)\right] \tag{3.1}$$

This introduces more flexibility - more *capacity* - to our boosted model. The question now of course is both how to create the classifiers h_t and how to pick the weights α_t appropriately.

3.2 The base algorithm

To build our weak learners, we must use some *base algorithm*, which we will denote $\mathcal{L}$ for short. The only requirement is, that for a given probability distribution D on the training set X, we can build a classifier:

$$\mathcal{L}[X, D] \tag{3.2}$$

3.3 Distributions

We describe the distributions D through a series of weights for each sample in the training set of size N. These should sum to one:

$$\sum_{i=1}^{N} w_i = 1 \tag{3.3}$$

The starting point - at what we will call time step 1 - will be a distribution where the weights are evenly distributed:

$$w_i^{(1)} = \frac{1}{N} \tag{3.4}$$

We will denote this $D^{(1)}$ for short, and generally $D^{(t)}$ for the t'th time step.

3.4 The algorithm

We now build the T classifiers as follows:

- Start at time step $t = 1$, i.e. with a distribution $D^{(1)}$ as described through equation 3.4.
- Repeat:
  - Build the classifier $h_t = \mathcal{L}[X, D^{(t)}]$.
  - Calculate the error rate for h_t:

$$\epsilon_t = \sum_{h_t(x_i) \neq f(x_i)} w_i \tag{3.5}$$

  - If h_t is not a weak classifier, i.e. if $\epsilon_t > 0.5$, then break.
  - Set the weight for the classifier to:

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right) \tag{3.6}$$

  - Set the weights for the distribution $D^{(t+1)}$ of the next time step:

$$w_i^{(t+1)} = \frac{w_i^{(t)}}{Z_t} \exp \left[-\alpha_t h_t(x_i) f(x_i) \right] \tag{3.7}$$

    Here $Z_t = \sum_{i=1}^{N} w_i^{(t)} \exp \left[-\alpha_t h_t(x_i) f(x_i) \right]$ is a normalization constant.
  - Go to the next time step, unless $t = T$.
- Construct the boosted classifier as:

$$H(x) = \operatorname{sgn} \left[\sum_{t=1}^{T} \alpha_t h_t(x) \right] \tag{3.8}$$

3.5 Mathematical justification

AdaBoost is a *greedy* algorithm, in that it chooses the best available option in each step: Each time we add another classifier to the sum in equation 3.1, we wish to do in a way that minimizes a reasonable *loss function*.

Let's start by naming the partial sum combinations from equation 3.1:

$$C_m(x) = \sum_{t=1}^{m} \alpha_t h_t(x) \tag{3.9}$$

Now assuming we have trained the $m-1$ classifiers and obtained appropriate weights, we're faced with finding h_m and α_m. The classifier we're looking to optimize is the sign of:

$$h = C_{m-1} + \alpha_m h_m \tag{3.10}$$

Consider the following *exponential loss function*:

$$\ell(h|D) = \mathbb{E}_D \left[\exp(-h(x)f(x))\right] \tag{3.11}$$

Here h is the classifier (without the sign function), and D a distribution described by a set of weights w_i. When a data point is correctly classified, we get a negative inside the exponential. Similarly, when a data point is incorrectly classified, we get a positive. So it makes sense to minimize this loss function.

We can rewrite the exponential loss function through the weights:

$$\ell(h|D) = \sum_{i=1}^{N} w_i \exp[-h(x_i)f(x_i)] = \tag{3.12}$$

$$\sum_{i=1}^{N} w_i \exp[-(C_{m-1}(x_i) + \alpha_m h_m(x_i))f(x_i)] = \tag{3.13}$$

$$\sum_{i=1}^{N} \underbrace{w_i \exp[-C_{m-1}(x_i)f(x_i)]}_{w_i'} \exp[-\alpha_m h_m(x_i)f(x_i)] = \tag{3.14}$$

$$\sum_{i=1}^{N} w_i' \exp[-\alpha_m h_m(x_i)f(x_i)] \tag{3.15}$$

Here, modified weights w_i' have been introduced.

Now we can minimize ℓ with respect to α_m by differentiating:

$$\frac{\partial \ell}{\partial \alpha_m} = \sum_{i=1}^{N} w_i' \exp[-\alpha_m h_m(x_i)f(x_i)]h_m(x_i)f(x_i) \tag{3.16}$$

We can now split the sum into two parts, according to whether h_m gets the classification right or wrong. When it is right, $h_m(x_i)f(x_i) = 1$, and when it is wrong $h_m(x_i)f(x_i) = -1$. So:

$$\frac{\partial \ell}{\partial \alpha_m} = \sum_{h_m(x_i)=f(x_i)} w_i' e^{-\alpha_m} - \sum_{h_m(x_i)\neq f(x_i)} w_i' e^{\alpha_m} \tag{3.17}$$