Lesson 5: Ensemble Learning: Boosting & Bagging

Spam/whitelist rules ("from: spouse", "Nigeria", "has pr0n", etc.) apply to many emails, but not all.

How do we generate rules? How do we combine rules? Similar to decision trees. If you already know the rules, the combining step may be like neural networks.

Ensemble Learning & Bagging

Combine simple rules into a complex rules.

- Learn over a **subset** of data to generate a **simple rule**.
- Learn over another **subset** of data to generate another **simple rule**.
- ...
- ...
- Combine all rules.

Subsetting

- *Uniformly randomly* pick a subset of data. Apply any learning algorithm on it.

Combine

- Regression: Take mean.
- Classification: equal-weighted vote.

Help with overfitting * For example, ensemble using several 3-order polynomial will outperform a single 4-order polynomial on testing dataset.

Bagging/bootstrap aggregation – the method of taking random subsets, learning, and taking mean of the individual machines. This is the simplest form of ensemble method, where we imporve lots of things.

Boosting

Subsetting: Instead of picking subsets randomly, take advantage of what we are learning as we go along. Especially, **pick examples we are bad at**. Take "hardest" subsets."

Combine: weighted mean/voting. To avoid just giving the average of everything. And also avoid the subsetting bounce back and forth.

About Error

Previously we defined error in classification as the number of mismatches, which implicitly treats every example equally important.

Now we define error as

$$Pr_D(h(x) \neq c(x))$$

where D is the distribution from which x is acquired, h is the hypothesis we pick, and cv is the underlying true concept.

Note here the data points x are not equally likely but drawed from the distribution D, and so the error rate will be affected by the D. Boosting is going to manipulate D, and this allows for harder (more probably) examples to produce a greater error contribution than already-learnt (less likely) example. We want to be good at the common points.

Weak learner

Does better than chance always: for all distribution D, $Pr_D(h(x) \neq c(x)) = 1/2 - \epsilon$. This means the learner always perform greater than a half.

This is actually a very strong and important condition. (*Personal understanding*: For example, if the distribution can be arbitrary, there must be one hypothesis n the hypothesis space which get all data points labeled correcly.) In some problems, there may exist distributions such that there is no weak learner. This may require an expansion of the hypothesis space (relieve the restriction bias).

Boosting in code

Given D = $\{(x_i, y_i)$ where y_i are -1 or +1$\}$

For t = 1 to T

construct D_t (a distribution)

find weak classifier h_t with small error $\epsilon_t = P_{Dt}(h_t(x_i) \neq y_i)$

Output H_{final}

Note that $h(x_i) = 1, -1$ just means it is a binary classification problem.

How to construct D_t? (AdaBoosting)

Base case (uniform initialization):

$$D_1(i) = 1/n$$

Induction on D:

$$D_{t+1}(i) = \frac{D_t(i) * e^{-\alpha_t y_i h_t(x_i)}}{z_t}$$

where α is going to be always positive:

$$a_t = 1/2 * ln(\frac{1-\epsilon_t}{\epsilon_t})$$

And Z_t is the nomalization factor.

When the hypothesis agrees with the truth, it will raise e to some negative number (->small probability). Otherwise, it will raise e to some positive number (-> large probability)

So generally puts more probability on incorrect instances and less probability on correct instances. But note we have an issue if error is exactly half or exactly 0.

How to combine smaller classifiers into H_{final}?

Take a weighted average of all weak learner combined with α_t. Since α_t carries the information of how well the weak learners perform.

$$H_{final}(x) = sgn(\sum_t \alpha_t * h_t(x))$$

Hypotheses/ensemble methods

Compare to decision trees.

Compare to locally weighted linear regression (as a modification of k-NN) where you stitch simpler models into a complex one. This characterizes all ensemble methods.

Why is boosting good?

Every iteration of boosting only "gains" more information by better classifying badly classified examples.

Ensemble learning

- Ensembles are good
- Bagging (simple ensemble) is good
- Combines simple into complex
- Boosting is really good – agnostic to learner
- Weak learners
- Error based on distributions
- Cool points: boosting does not seem to overfit over arbitrary iterations in practice (or does it?)