machine learning notes

guillem.tobias

1 TO DO LIST

- **I can start by reading the introduction to new chapter**
- Chapter 10: Boosting. *Some basis expansions may be required.*
- $$\mathcal{D}$$
  to make fancy letters
- *Optional*
- Expanding basis
- Kernel
- Understanding machine learning book. Has chapter list of basics.

2 Classifiers

3 Logistic regression

Why do we sometimes define it with K-1 alternatives and sometimes we don't care.

4 Mathematical statistics: Brief on statistical decision making.

I need to write that couple pages here. Would need a bit more familiarity with the prob theory first.

5 ESLII CHAP 5: Basis expansions

5.1 Basics for understanding boosting.

The core idea in this chapter is to augment/replace the vector of inputs X with additional variables, which are transformations of X, and then use linear models in this new space of derived input features.

Most basic splines are constants, follow the same reasoning as a tree

Knot/boundary location is *uniform over the given x_p* at first, I am sure there are algorithms to improve location and whatnot.

Traditional workflow: Take each variable and make splines out of it, can be constant or linear or cubic, whatever. Constant is clearest example. Then these splines have parameters to be estimated, the grouped mean or whatever, if it's linear it's like a local ols. **We run the same linear algorithm in the end, that is nice part.**

Preprint. Under review.

6 AoF CHAP 12: Statistical decision theory.

7 ESLII CHAP 7: Model assessment and validation

We choose a loss function, it will be mean squared loss 99 percent of the times.

$$L(Y, f(X)) = (Y - f(X))^2$$

Risk is the expected loss function, expectation over what? Well, over the random stuff, so if Y is random it will be over $f(x, y)$ if Y is a fixed unkown quantity just over $f(x)$.

Then we go to test error: $Err_t = E[L(Y, \hat{f}^{-X_t,-Y_t}(X))|test : X, Y]$ This test error is random because it comes from a random sample, because the f function is generated with the training data no? ,but it can be easily computed via sample analog, for a given sample this is a known quantity. **A bit unsure.**

By LIE we get back to original definition of risk as the expected. $E_t[Err_t]$, this definition is very useful as a more practical one because we can compute everything's sample analog with ease. Compute error over samples, inner error is known no? and then average, justifies cross validation. It is analogous. *This reasoning is general, it follows for any loss function, including for classification.*

The chapter has **2 aims**: **Model selection** and **Model evaluation**, from the previous results we know that it is very similar.

In data rich scenarios, we divide into 3 parts, training, validation for the model selection, and evaluation for the test. Because model selection and evaluation are kind of similar, you don't want to tune your models hyperparameters on the test set.

Then we get back to bias variance decomposition. **This would be good practice to derive and to fully understand what they are doing, expectations over what blabla**

7.1 7.4 part of the chapter

I think that my reasoning above was wrong as now they define it more properly. Define the training set t. Then

$$Err_{X_0,Y_0} = E_{X_0,Y_0}[L(Y^0, \hat{f}(X^0))|t]$$

This expectation is over the distribution of the test sample. Now this is a random quantity because X0, y0 are new, this is very much CV analog. For the unconditional error we take LIE over the distribution of the training samples, we get expected error or risk. Then this can be done in sample version very easily.

7.2 7.7 INTERESTING BAYES AND BIC.

7.3 7.10 Cross validation

Remember asymptotics of cross validation are tough due to correlation, read that paper and the executive summary. Cross validation estimator of pred error (I think they should say risk).

$$CV(\hat{f}) = \frac{1}{N} \sum_i = L(y_i, \hat{f}^{-k(i)}(X_i))$$

They mention this is approximately unbiased estimator, but with high correlation in the sample which is what makes it complicated I am guessing. This can be used for hyperparameter tuning, everything the same with a given alpha. For hyper parameter tuning we usually need to use brute force grid search.

The right way to do CV, *avoid data leakage*, start with CV split and then do variable selection. If we have multistep modeling, like variable selection. **The CV is done in the first step and then all the rest follows as normal but applied to the K-1 folds, never the total sample.**

7.4 Bootstrap

Z=$(z_1, ..., z_N)$ where $z_i = (x_i, y_i)$ tuple, so each tuple is an iid sample. $S(Z)$ is the statistic and we can compute the variance or confidence interval with ease.

$\hat{Var}$ = $1/(B-1)\sum_{b=1}(S(Z^b) - av(S_b))^2$

Beautiful equivalence: Note that dVar[S(Z)] can be thought of as a Monte-Carlo estimate of the variance of S(Z) under sampling from the empirical distribution function $\hat{F}$ for the data (z1, z2, . . . , zN).

Interesting thought, so the bootstrap will converge to the variance of the estimator in the sample? And we think that sample will be representative.

Not a good estimator for expected error/risk. Can be more or less fixed.

7.5 Expected error or conditional error

Review the simulation they do because it is confusing. **Main takeaway:** Estimating conditional is more difficult. CV /boot give unbiased estimates of the error rate.

8 Highly imbalanced classes

https://stats.stackexchange.com/questions/235808/binary-classification-with-strongly-unbalanced-classes

https://stackoverflow.com/questions/59409967/proper-way-to-handle-highly-imbalanced-data-binary-classification

9 ESLII CHAP 9.2: Tree based methods.

Trees are usually done with this recursive binary splitting, so we split the whole regressor space into 2, then we split the subsequent regions and so on and so on. If you don't stop the tree will be overfitting a lot.

Key equation that describes a tree:

$$\hat{f}(X) = \sum_{m=1}^{N_r} c_m \mathbf{I}(X \in R_m)$$

This seems complicated but it just means that for all observations in a given region we will predict with a constant number.

If we then use a squared loss criterion the c_m tha minimizes the loss will be the conditional mean for that given subset.

It is easy start from $\sum_i^n (y_i - f(x))^2$ then plug in our f(x) and just decompose this using the indicator, so partition the n sum into different smaller sums for each subgroup. *Do the derivation as an exercise.*

Then for any general split: We consider the following regions: $R_1(j, s) = \{X|X_j \leq s\}$ and $R_2(j, s) = \{X|X_j > s\}$

Our optimization problem is then to minimize the squared loss in both regions :

$$\min_{j,s}[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_1(j,s)} (y_i - c_2)^2]$$

Where we know the inner problems are just an average within a group which is super easy to compute.

But then we can have many j and s how is this problem easy to solve then? It is discrete, infinite combinations? Or if we just take points in the sample then we have still a lot of points to compare! **ASK Milan.**

9.1 Pruning

More or less intuitive but I don't think I fully understand. Weakest link prune means the part that decreases the loss by less is condensed into a single node. So you want to start this thing from the bottom. **PRUNE THE LEAVES LMAO.**

9.2 Classification trees

We will normally classify to a given label via majority vote within the region. The important definition is for node m, which represents region R_m and observations N_m we define:

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{i \in R_m} (\mathbf{1}(y_i = k))$$

These are the true labels, no prediction here. We then classify according to this rule:

$$k(m) := \text{argmax}_k(\hat{p}_{mk})$$

Where we loop over the k for given m region and classify as the k that has larger presence in the region.

When picking best regions we can't use the squared error loss within, so we will then be using either Gini or cross-entropy/deviance. They generalize to K categories without issue. *Missclassification error should be avoided.*

9.3 Surrogate splits

If we have missing data we can make surrogate splits, for each branch we think of the multiple cuts we could make, get the top 5 and compute them with the non missing variables. Then if the data has missing values for the first optimal cut, we go to second and so on so on. Then thinking about the regions gets a bit complicated though no? What happens then? **Complete. READ also missing data 9.6 chapter.**

```
$i \leftarrow 10$
if $i \geq 5$ then
    $i \leftarrow i - 1$
else
    if $i \leq 3$ then
        $i \leftarrow i + 2$
```

Intuitively this makes sense if the data is truly missing at random If we have categorical data we can add a "missing" category.

10 ESLII CHAP 15: Random forests

A finely tuned random forest can be a very powerful model for Task 1 of the project.

10.1 ESLII CHAP 8.7: Bagging

TO DO READ CHAP 8 ON BOOTSTRAP *In Section 8.4 we investigated the relationship between the bootstrap and Bayes approaches, and found that the bootstrap mean is approximately a posterior average.*

Bootstrap aggregation can be applied to many models, it works especially well with high variance low bias models like trees.

General procedure: We have training data **Z** = $\{(x_1, y_1), ..., (x_N, y_N)\}$ We fit a model to the original data and we get: $\hat{f}(x)$. OG model.

We do many boostrap samples and rerun the model at each iteration, $\hat{f}^{b_1}(x), ..., \hat{f}^{b_M}(x)$

Our bagging model will be defined as

$$\hat{f}_{\text{bag}}(x) = \frac{1}{M} \sum_{b=1}^{M} \hat{f}^{b_i}(x)$$

And we will of course use it on the original data. *For classification we use majority vote on the predictions.*

This has very nice interpretation, if we define $\hat{E}$ as the empirical distribution. The true bagging is defined as $\mathbf{E}_{\hat{E}}(\hat{f}^*(x))$ So our estimator is a Monte Carlo approximation to it. And if we have a large enough sample the empirical distribution converges to the true distribution. *Maybe we get something like exp of f(x), IDK, check statistical decision theory then jajaja.*

EXAMPLE 8.7.1 Good, check it out, but how do they compute this Bayes error so easily check out the first weeks of theory.

With predictors we have some semi guarantees it will improve, with classifiers there are no such guarantees.

10.2 Focus on random forests.

Random forests are not just bagged trees, they are bagged decorrelated trees!!

Boosting and random forests have similar performance on a lot of problems, but are super easy to train.

Same procedure as boosting but the idea is to decrease the correlation amongst the trees to reduce the variance of the model. Then what we do is for every bootstrap sample, at each split decision, we only consider a subset of m ¡¡ p variables to split from. This decorrelates the trees quite a bit.

The expected value of the average will be the same as for one tree but we will be decreasing the variance. *This argument is imprecise as the bootsrtrap samples are not iid but ok we approximate some empirical distribution blablabla* **Law of large numbers reasoning.** Variance of i.d not independent observations will be:

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

Deriving this reminds me of time series but I should be able to do it no?! **TRY EXERCISES 15.1 AND 15.4** *In 15.1 the key is recognizing the number of covariance terms we will have, for that it is useful to draw a box! b(b-1)*

10.3 Out of bag error

For each observation $z_i = (x_i, y_i)$, construct its random forest predictor by averaging only those trees corresponding to boot- strap samples in which z_i did not appear.

This can be an interesting procedure for our

11 ESLII CHAP 10: Boosting and additive trees.

Objetives: Understand boosting, especially the gradient based tree methods.

11.1 AdaBoost algorithm

$w_i = 1/N \quad \forall \quad i$ For m in 1:M:

- a. Fit $G_m(x)$ usually a tree A-B
- $\text{err}_m = \frac{\sum_{n=1}^{N} w_i \mathbb{I}(y_i \neq G_m(x))}{\sum_{n=1}^{N} w_i}$ This is a weighted error metric.
- $w_i = w_i \text{exp}(\alpha_m \mathbb{I}(y_i \neq G_m(x))) \quad \forall i$ Notice only the weights of the missclassified points are being updated.

Final ensemble predictor is now:

$$G(x) = \text{sign}[\sum_{m=1}^{M} \alpha_m G_m(X)]$$

Boosting is a way of fitting an additive expansion over a set of basis functions. This sounds like a mouthful but you can see it from the last estimator.

11.2 Forward stagewise additive fitting

Generally you would have a class of models $f(x) = \sum_{m=1}^{M} \beta_m b(x; \gamma_m)$ where b depends on data and is parametrized by the gamma. Then you would love to minimize the following:

$$\min_{\{\beta_m, \gamma_m\}^M} \sum_{i=1}^{N} \mathbb{L}(y_i, \sum_{m=1}^{M} \beta_m b(x; \gamma_m))$$

But this is super complicated so we can simplify it to the following algorithm, Forward stagewise additive fitting (FSAF):

Initialize $f_0(x) = 0$

For $m = 1 : M$

- $(\beta_m, \gamma_m) =$ argmin $\sum_{i=1}^{N} \mathbb{L}(y_i, f_{m-1}(x_i) = \beta b(x_i, \gamma))$ See the prediction will be based on a weighted average of the present model and all the previous models. But the params of all previously fitted models will not be modified.
- Update previous models to be the ensemble with the last iteration: $f_m(x) = f_{m_1}(x) = \beta_m b(x; \gamma_m)$

Once you have a final ensemble you run the classification decision.

Boosting is a special case of this.

11.3 Loss functions and deriving what boosting is doing 10.4-10.6

AdaBoost is forward stagewise additive fitting when the loss is exponential. For any base ”weak” classifier.

11.4 Boosting trees

Forward stagewise additive fitting but the general function are now trees. These are complex models, given that we have to find the region and then the optimal parts within the region (easy part). An iteration of FSAF would look like:

$$\hat{\Theta} = \text{argmin}_{\Theta_m} \sum_{i=1}^{N} \mathbb{L}(y_i, f_{m-1}(x_i) + T(x_i, \Theta_m))$$

Special cases could show up in the exam!

11.5 Numerical optimization via Gradient boosting

THIS IS WHAT I NEED!

12 Understanding machine learning book: