machine learning notes

guillem.tobias

1 TO DO LIST

- Chapter 2 exercises: 2.5,2.6,2.7,2.9(intuitive but do more mathematically). First approach done transcribe.
- Chapter 3 many exercises, most need quite a bit of lin alg. Check it out as lin alg is essential for exam.
- Chapter 10: Boosting. *Some basis expansions may be required.*

Math for ML: Chapter 7 continuous optimization.

Summary: We have algorithms that will converge to a minimum - (stochastic) gradient descent - but that minimum may be a local one. In constrained optimization we use Lagrangian and duality. The dual problem can sometimes be easier to solve than the primal. In 2 cases, quadratic and linear programming problems, we will have strong duality, that is, both (primal and dual) problems give the exact same solution and they can be solved efficiently.

Preliminaries

The most important fact is to check the dimension of what you are deriving! Most times it might be a hidden dot product.

- All derivatives over x.
- $\mathbf{a}'\mathbf{x} = \mathbf{x}'\mathbf{a} = \mathbf{a}$ This one is key!!
- $\mathbf{A}\mathbf{x} = \mathbf{A}$
- $\mathbf{x}'\mathbf{A}\mathbf{x} = (\mathbf{A} + \mathbf{A}')\mathbf{x}$

An interesting way to rewrite a constrained optimization problem is with infinite steps functions: $J(\mathbf{x}) = f(\mathbf{x}) + \sum_{i=1}^{m} \mathbf{1}(g_i(\mathbf{x}))$ and then we can view the lagragian $J(x) = \max_{\lambda \geq 0} \mathfrak{L}(\mathbf{x}, \lambda)$ the lagrangian is a lower bound of J(x) we can see this when constraints are negative and positive how max will be lambda 0 or infinity. This can be used to derive the weak duality results. Weak duality is not directly super useful in my very limited view as it is just a bound, although bounds can be very useful.

Linear programming and quadratic programming

We have a linear program: $\max_{\mathbf{x} \in \mathbb{R}^{\mathbf{d}}} \mathbf{c}'\mathbf{x}$ s.t $\mathbf{A}\mathbf{x} \leq \mathbf{b}$ In Lagrangian form we get $\mathfrak{L}(\mathbf{x}, \lambda) = \mathbf{c}'\mathbf{x} + \lambda'(\mathbf{A}\mathbf{x} - \mathbf{b})$ To get the dual we just take derivative with respect to x plug in the FOC and add the FOC and multiplier positivity as constraints. The derivation has been completed the tricky step is $\lambda' A$ is a vector which makes the whole thing a dot product!

For quadratic program we can get something very similar remember the $x'Qx$ Q is symetric positive definite so tranpose is equal. An important example is solving a system of equations via squared error loss, OLS.

$||\mathbf{A}\mathbf{x} - \mathbf{b}||^2 = \mathbf{A}\mathbf{x} - \mathbf{b}'\mathbf{A}\mathbf{x} - \mathbf{b} = \mathbf{x}'\mathbf{A}' - \mathbf{b}'\mathbf{A}\mathbf{x} - \mathbf{b} = \mathbf{x}'\mathbf{A}'\mathbf{A}\mathbf{x} - mathbfx'A'b - (b'Ax) + \mathbf{b}'\mathbf{b}$

Preprint. Under review.

Notice that all terms are scalars as they should be. Then 2 and 3rd term are the same as the transpose of a scalar is the scalar. Then take derivative. Also while A is not symmetric, A'A is symmetric which simplifies the first derivative.

Convex conjugates

Exercises

7.8 from math book: min wrt w $\mathbf{w}'\mathbf{w}s.t\mathbf{w}'\mathbf{x} \leq 1$

ESLII CHAP 2: Overview of supervised learning.

This chapter contains the basic ideas and then some

ESLII CHAP 3: Linear methods for regression.

These two chapters are very important and linear algebra heavy (for obvious reasons).

ESLII CHAP 4: Linear methods for classification.

Many important things here, to complement start with the All of Stats chapter.

- LDA for visualization seems very cool.
- MLE of a multinomial review in depth, don't want to get stuck there.
- LDA or logit sub-section is very important! Relationship between the 2
- Exercise 4.2: lda and linear regression equivalence.
- Exercise 4.6 with separating hyperplanes converging.

ESLII CHAP 5: Basis expansions

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.**

- Indicator functions and OLS over disjoint regions, ols estimates will give you sample mean over region.
- Extends to previous chapters very nicely, then coefficients interepreted as usual.
- Nonparametric logistic regression and natural splines.
- Some RKHS 5.8 also seems interesting.
- Interesting exercises for SVM from infinite dimensional to low dimension.

ESLII CHAP 6: Kernel smoothing

Have a quick look could show up somehow.

ESLII CHAP 7: Model assessment and validation

- Categorical error deviance important.
- k nearest neighbor bias var decomposition
- idem for linear regression
- understand ein picture.
- in sample vs extra sample
- AIC-BIC were mentioned in class so I should check that out.
- EXERCISES SUPER IMPORTANT DO MOST OF THEM.
- 7.3 and 5.13 on special cross validation cases.
- Bootstrap for cv learn the mathematicall pr of observartion in sample blabla

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**

7.4 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.

7.7 INTERESTING BAYES AND BIC.

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.**

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.

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.

ESLII Chap 8: Inference and bagging

- Page 285 is very interesting, do as exercise understand deeply
- Bagging good vs bad classifiers. Wisdom of crowds. Application in random forests.
- Exercises don't seem to be too good, check them out anyways, derivation in chapter can also be used as exercise.

*

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

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.

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)*

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

ESLII CHAP 9.2: Tree based methods.

Very important section! 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.**

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.**

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.*

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.

ESLII CHAP 10: Boosting.

- Exponential loss and AdaBoost go through full derivation 343.
- Then derivaation 10.16 -¿ It is an exercise as well.

- Discussion on different loss functions is also a very interesting one.
- Boosting trees
- Gradient boosting also very important. **Not understood by me!**
- Read up on partial dependence plots! Also very interesting!

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
- 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.

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:

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

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 $m = 1 : M$

- $(\beta_m, \gamma_m) = \text{argmin} \sum_{i=1}^{N} \mathbb{L}(y_i, f_{m-1}(x_i) = \beta b(x_i, \gamma))$ 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.

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.

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!

Numerical optimization via Gradient boosting

THIS IS WHAT I NEED!

ESLII CHAP 12: SVM

- One of the most math intensive methods: need strong lagrangian + duality.
- In chap 4 there is some discussion on optimal separating hyperplanes for separable problems.
- 12.2.1 Go through these derivations.
- 12.3.2 SVM as a penalization model seems very interesting.
- 12.39 equations of HUber very interesting -¿ also exercise I think.
- Go in depth into this chapter.
- Good treatment and practice in math book: optmization chapter and svm

ESLII CHAP 13: nearest stuff

- Something interesting at 13.2-13.4

ESLII CHAP 13: Unsupervised learning.

ESLII CHAP 15: Random forests

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

- A lot of things to know in the random forests, specially compared to bagging and boosting.
- Some important exercises here
- Analysis of random forests 15.4, check it out. Can understand derivation as exercise.
- Almost all these exercises are super interesting to learn.

2 All of stats: Chap 12 Statistical decision theory

We have a wide variety of estimators $\hat{\theta}$, how do we choose amgonst them, knowing only that the truth $\theta \in \Theta$, where the big theta is the parameter space. We will be trying to measure the difference between our estimator and the truth using loss functions. They map $\Theta * \Theta \rightarrow \mathcal{R}$ There are 5 key ones (more in ML): squared error loss, L_p losses, zero-one and the kullback leibler.

IMPORTANT: The estimator $\hat{\theta}$ is a function of the data. And only the data. Yes the parameter will drive the data, but for given data the estimator is the same for different truths. That is my best intuition for what comes later.

Risk

Definition of risk:

$$R(\theta, \hat{\theta}) = \mathbb{E}_\theta(L(\theta, \hat{\theta})) = \int L(\theta, \hat{\theta}(x)) f(\theta; x)\, dx$$

Intuition:
This is the expected value (over data generated from truth) of the loss function.

I think there they try to avoid Bayesian thinking here. I usually interpret as the expectation given the true underlying data distribution...I think it is a bit confusing. In the end the correct

intuition is we want to compare risks of different estimators at all possible truths. There is only one truth but we don't know ex ante which one it is. **Exercise:** Do variance squared bias decompostion for the case of squared error loss.

Example 12.2-12.3: In 12.2 the θ is not random, it is just unknown to the researcher. We take our expectation over that random part which is X.

In 12.3 we use the decomposition from squared error. In the first term we get something in terms of truth and sample size, so this is how we see that for unbiased estimators we tend to go to 0 risk as the variance diminishes. *We can also get risk estimators that don't depend on true parameter!!!*

Bayes risk and minimax:

It is difficult for one risk function to uniformly dominate another, so we need other ways to compare risk functions and the estimators that create them.

Definitions: The max risk is just picking the truth that could yield to worse perfomance, aka, maximize the risk function. $\sup_\theta R(\theta, \hat{\theta})$ This is a bit pessimistic. Bayes risk is defined as:

$$\int R(\theta, \hat{\theta}) f(\theta)\, d\theta$$

Intuition: It is now where in Bayes terms we can think of the possible truths, not as Bayes thinking but as a researcher that doesn't know which truth it is ex-ante. This can be defined with the uninformative prior.

Example 12.5 This shows how we can pick one or another estimator depending on our criterion. One number summaries are imperfect though, we should exercise caution.

Bayes and minimax estimators.

Instead of getting random estimators and getting the bayes risk, we can derive estimators such that they minimize the bayes risk. These are called Bayes rules. We have an analog for minimax. Pick the estimator that minimizes the maximum.

2.0.1 Deriving Bayes estimators:

Preliminaries: Bayes theorem We are just using the definition of conditional expecation and applying marginals whatever. Continuous Bayes theorem.

$$f(\theta|x) = \frac{f(x|\theta)f(\theta)}{m(x)} = \frac{f(x|\theta)f(\theta)}{\int f(x|\theta)f(\theta)\, d\theta}$$

We define the posterior risk as (just the conditional risk given the data, this is definition of conditional expectation):

$$r(\hat{\theta}|x) = \int L(\theta, \hat{\theta}) f(\theta|x)\, d\theta$$

Now it states that Bayes risk can be rewritten as a function of posterior risk:

$$r(f, \hat{\theta}) = \int r(\hat{\theta}|x) m(x) dx$$

There is a long proof showing this equality is true. **And now if we minimize pointwise for each x, we get the Bayes estimator**.

IMPORTANT: For squared error loss, the Bayes estimator is the conditional expectation. There are a couple of other important rules.

The proof in itself is trivial, start from posterior risk and take a derivative with respect to the estimator. FOC and you get a conditional expectation. *The important part is that we have done this so that the x are fixed, if not it is impossible. Try those exercises from ML*

Try example 12.9, not easy. Well expected value with a prior and we get posterior mean.

Rest is not too useful, do the exercises at the end of the chapter for ML.

3 All of stats: Chap 22 Classification