**Machine learning in Python - Micheal Bowles**

CHAP1- TWO PREDICTIVE ALGOS

This book deals with one class of machine learning problems, generally referred as function approximation. Function approximation is a subset of problems called supervised learning problems

* limited to two families of algorithms
* focus on the inner workings
* decision making based on type of problems

Penalized Linear Regressions and Ensemble Methods (PLR and EM)

Variants that will solve regression and classification problems

PLR came from the need to overcome limitations of OLS regressions

The basic problem with OLS is that sometimes it overfits.

Notably, when the nb of points in the dataset is equal or less than the degree of freedoms (nb attributes) then the prediction are not very good (e.g genetics).

PLR has a tunable parameter to balance between under and overfitting

PLR provides a way to systematically reduce degrees of freedom to match the size of the data available and the complexity of the problem

EM came from the need to overcome limitations of binary decision trees

Basic idea is to build a horde of predictive models and combine their outputs (by averaging or taking majority answer) Individual models are called base learners. Most often, binary decision trees.

The trees can be trained on random samples of the training data (Bagging technique) and generate slightly different trees.

Both families allow to rank the various inputs by their importance for producing productions which is a crucial feature. One of the most time consuming step: feature engineering or feature selection. Taking some of the guesswork out of the development process and make the process more sure.

What to choose?

- PLRs faster training times than EMs

- feature engineering is a matter of trial and errors and can require several iterations

- early stage of developments, PLRs might be a better choice to get a feel of the variables useful for prediction. also for baseline performance comparison with other algos later in development

- PLRs also generate prediction much faster

- choice depends on problem complexity, predictive model complexity, and data set size to achieve the best deployable model

- EMs often give better results if they are adequate data and somewhat indirect measures of relative variable importance

Process steps:

1.extract and assemble features to be used for prediction

2.develop targets for the training

3.train a model

4.assess performance on test data

Determining performance of a trained model:

- set data aside - out of sample testing gives performance of algo

CHAP2- DATA EXPLORATION

nomenclature:

individual cases, instances, examples, observations -> data points

targets, labels, dependent variables, responses -> observed outcomes, things you want to predict

attributes, predictors, features, independent variables, inputs -> data used to make predictions

The trick with machine learning is to generate a model that will generalize to new cases (not only memorize past cases).

To achieve that, the algorithm must be forced to pay attention to more than just one row of data.

Therefore, one attribute must not be different at each observation (e.g user ID, if not indication of signup date), too specific for predictions.

different types of attributes:

- numerical type

- categorical (or factor) type - no order relationship between the various values

The type of attribute will shape your algorithm choice s and the direction you take in developing a predictive model

same dichotomy exists for label, e.g amount spent on books vs amount>$200?

label type determine whether regression or classification problem (binary or multiclass)

thinks to notice about data sets:

- shape, number of rows and columns

- number of categorical variables and number of unique value for each attribute

- missing values

- summary statistics for attributes and labels (using describe function)

- visualization of outliers

- structure of how data is stored. May need to be factored in when doing the subsequent sampling (e.g Male-Female, with all males before females, you do want to sample evenly, not introducing a bias in your sampling)

missing values:

figure out how many observations have missing values, along with total number of missing entries, to see whether you can afford to discard them. Filling them is called imputation. Several methods for imputing: average of fields, machine learning techniques (as if it was a label)…

outliers:

- visualization using quantile-quantile plot. stats.probplot(). Compare two distributions. Possibly how different from normal e.g.

- may cause trouble in model building or prediction. see how much correlated are the errors you model makes with these outliers and then maybe take step to correct them (either erase if not likely to come up again, segregate and train as separate class, or replicate poor performing examples to

force them to be more heavily represented)

- generates b-box for all the real-valued columns. sometimes might be worth it to eliminate the larger-scale attributes (with values out of scale with the others) or better, normalize all of them and show the box and whiskers plot with normalized values.

visualizing interrelationships between attributes and labels:

- visualization with parallel coordinates plots. For each example, plot the value of an attribute against its attribute name or index. Color-coding based on the labels can help you see some types of systematic relationships btw the attribute values and the labels. For classification problems you have two or a few colors but for regression problems you need shades of colors, can use logit transform function for soft range compression, better).

These kinds of insights can help interpreting and confirming predictions made by your trained model

- visualizing pairwise relationships using scatter plots. Cross plot one attribute with the labels e.g. This works fine when problem is a regression problem but can also work for a classification problem by assigning a number to each category and plotting the attribute value against it. You'll see points on the different lines, you can spread them out and make them less opaque to get a better feel of what’s going on.

- visualizing correlation between attributes and label: plot heatmap of Pearson's correlations

CHAP 3 - PREDICTIVE MODEL BUILDING: BALANCING PERFORMANCE, COMPLEXITY AND BIG DATA

Assessing performance of predictive models, predictions close to targets:

- regression problem: where target is a real number, performance is measured in terms like the mean squared error (MSE) or the mean absolute error (MAE).

- classification problem: misclassification error (fraction that is predicted incorrectly)

Performance needs to be assessed on new data

Best practice is to hold out some of the data from the training set

Out-of-sample error should matter

Factors affecting choices and performance:

- simple vs complex problem (linear vs non linear methods): intuitively, it seems that a complex model should be fit to a complex problem

- size of the data set: but data size may dictate that a simple model fits a complex problem better than a complex model

- shape of the data set: actually size and shape matter, i.e rows vs columns. Adding a column is adding a degree of freedom (a beta coefficient to the equation). Adding another degree of freedom

is making the model more complicated. Making the model more complicated requires more data. For this reason, it is common to think in terms of the ratio of rows to columns - the aspect ratio

Bottom line: accurate predictions for complicated problems require large volumes of data. But size isn't quite a precise enough measure. Shape matters.

Linear models -> Problem simple OR more columns than rows

Non linear models -> Complex problems with way more rows than columns

Another key aspect is training time: linear techniques train much faster

Measuring the performance of predictive models:

- MSE

- misclassification error

- receiver operating curves (ROC curves)

- area under the curve (AUC)

regression pb:

- summary statistics for the errors

MSE, MAE, RMSE (root mean squared error). MAE and RMSE are of same magnitude

Good to compare std(targets) w/ std(prediction errors). if they are roughly the same the prediction algo is not performing well (could perform the same by replacing the prediction algo by the mean of the targets).

- histogram of the error or tail behavior

- degree of normality

=> yield insights into error sources and potential performance improvements

classification pb:

- misclassification error rates

- confusion matrix or contingency tables. can help set the cursor right to make the classification at the best possible threshold. Adjust the classifier.

The threshold is the prob of the classification being correct. Depending on the implications of the different types of error and their relative importance, you set it high or low.

Classifying into four categories:

TP: True positive

FP: False positive

TN: True negative

FN: False negative

Training the classifier can mean e.g converting the labels into numeric values and then using the OLS to fit a linear model

The best value for the threshold may be the one that minimizes the misclassification error. Sometimes, however, there may be more cost associated to one type of error than with another.

The choice of the threshold should also account for the real proportions of each category.

- when no costs are available: receiver operating curves (ROC curves) can be a method to assess the overall perf of the classifier = single plot that summarizes all the info of the different contingency tables (when moving the threshold):

TPR vs FPR

TPR: True positive rate = TP / (TP+FN) = prop of positive examples correctly identified as positive

FPR: False positive rate = FP / (FP+TN) = prop of negative examples incorrectly identified as positive

AUC is the area under the curve

The closer to 1 the better

Simulating performances of deployed models:

Need to do out-of-sample test. Generally 25-35% of data is held out

Another approach to hold out data is n-fold cross validation:

- several training and test passes are made through the data divided into n disjointed sets

- provide several sample of the error (help set boundaries estimates)

- can keep more data in the training set (gives generally lower generalization errors and better final perf)

- come at the expense of greater training time

Fixed holdout, probably better when:

- training time unbearable with n-fold

- enough data that holding some extra out won’t affect adversely perf

Sample should be representative of the whole data set:

- spread uniformly

- without introducing a bias

- sampling may need to be carefully controlled if the phenomenon being studied has unusual statistics; Care may have to be taken to preserve the statistical peculiarities in the test sample (e.g predicting rare events like fraud or clicks). Random sampling may under or over represent the infrequent events.

Stratified sampling divides the data into separate subsets that are separately sampled and then recombined.

After a model has been trained and tested it is good practice to recombine the training and test data into a single set and retrain the model on the larger data set. The deployed model should be trained on all the data (once you have a good idea of prediction errors).

Achieving harmony between model and data

Using OLS to illustrate several points

First: overfitting = significant discrepancy btw errors on the training data and the test data

Second: methods to overcome overfit problem

OLS good example, supervised algo with a training procedure and a deployment procedure. Like other modern func approx algo it can be overfit in some circumstances.

OLS is missing an important feature though, there's no means to throttle it back when it overfits. The source of overfitting is having too many columns of attributes.

However, getting rid of some involves deciding how many to eliminate and which ones.

Methods exist to adjust the throttle on OLS:

- The brute-force method is called best subset selection. Impose constraints on the number of columns, try all the subsets and see which ones perform better for subsets of 1, 2, ... n columns. Pick the one with the least out of sample errors. Problem is that it requires too much calculation even for a modest nb of attributes (say 10, implies 2\*\*10= 1000 subsets)

- forward stepwise regression. idea is to start with one column subsets and then, given the best single column, to find the best second column to append instead of evaluating all possible two-column subsets. Rank attributes by their predicting quality in the process. The algo produces a sequence different model with a different complexity parameter (number of attributes)

- ridge regression - also has a sequence of models but instead of different number of attributes, ridge regression models have different values of alpha, the parameter that determines the severity of the penalty on the betas (alpha is the complexity tuning parameter).

CHAP 4 - PENALIZED LINEAR REGRESSION

Ridge regression is a specific example of a penalized linear regression algo.

Why PLR methods are so useful:

Several properties make PLR methods outstandingly useful, including the following:

- extremely fast model training

- variable importance information

- reliable perf on a wide variety of pbs - particularly on attributes matrices that are not very tall compared to their width or that are sparse. Sparse solutions (that is more parsimonious model)

- may require linear model

extremely fast coefficient estimation:

matters because process is iterative. trial and error. makes development process faster.

also if conditions change rapidly, it is important to be able to retrain your models, sometimes speed is needed.

variable importance info:

ranking for each attribute you chose to base your model on. Helps weed out variables during the feature engineering process.

also knowing what variables are driving the predictions helps explaining your model to others

These algos help you quickly get your arms around the problems and decide which features are going to be useful

extremely fast evaluation when deployed:

simple computation of the sumproduct of factors and attributes

reliable perf:

will generate reasonable answers to problems of all different shapes and sizes. sometimes outperform contenders

sparse coefficients:

means many of coefs are zero. easier to interpret. faster to compute.

problem may require linear model:

requirements in some fields like insurance or drug testing where calculations would need to be written in plain English as part of a contract. Nearly impossible for EMs

When to use Ensemble Methods:

Prime reason is you might get better perf: e.g complicated problems with plenty of data to resolve the problem's complexities

also, EMs give more info about variable importance and the relationship btw attributes and predictions (higher order info)

PLR: regulating Linear regression for optimum perf

Let’s denote:

Y the column of outcomes

X the matrix of real-valued attributes

B the vector model coefficients

B0 a scalar

Training linear models: minimizing errors and more:

In many problems, the size of the problem or the interrelationships between the variables makes guessing the B’s impossible (impossible to find by hand); So the approach taken is solving a minimization problem => find values for B’s that minimize the MSE (but doesn’t make it zero, otherwise overfits).

The square of the error is chosen because it’s positive regardless of whether the error is positive or negative and because the square function facilitates some of the math.

The formulation of the OLS regression problem is to find B0\*,B\* that satisfy:

*B0\*,B\* = argminB0,B (1/n \* sum1>n[( yi – (xi\*B + B0))²] )*

Adding a Coefficient Penalty to the OLS formulation:

E.g ridge regression adds a penalty term to the basic OLS problem.

The penalty term is:

*BT\*B\*lambda/2= (B1²+B2²+…+Bn²)\*lambda/2*

It adds to the right hand side of the equation of the OLS. The minimization is then forced to balance the conflicting goals of:

* minimizing the squared prediction error
* and minimizing the squared value of the coefficients

It’s easy to minimize the sum of the squared value of the coefficients. Just make them all zero. But that results in large prediction error. Similarly, the OLS solution minimizes the prediction errors by themselves but may result in a large coefficient penalty depending on how large lambda is.

If lambda = 0, no penalty

If lambda = +inf, the penalty is so severe that it forces all the B’s to zero (notice, however, that B0 is not included in the penalty so the prediction becomes a constant independent of x’s)

Ridge regression can have a similar effect to leaving some attributes out. The process is to generate a whole family of solutions to the penalized version of the minimization problem (solving it with different values of lambda). Each of these solutions is then tested on out-of-sample data and the solution that minimize out of sample error is used for making real-world predictions.

Other useful coefs penalties – Manhattan and ElasticNet:

Any metric of vector length would work. You can gauge the length of a vector in a number of ways. Using different measures of length change important properties of the solution.

Ridge regression: Euclidean geometry (sum of squared B’s)

Lasso regression: Manhattan length or L1 norm (sum of absolute B’s) – metric of taxicab geometry

**||***B***||***\*lambda = (***|***B1***|***+* **|***B2***|***+…+* **|***Bn***|***)\*lambda*

The double vertical bars are called norm bars. They denote magnitude for things like vectors or operators. These different coefficient penalty functions cause some important and useful changes in the solutions. One of the main differences is that the Lasso coefficient vector B\* is sparse for large to moderate values of lambda, meaning that many of the coefs are zero. By contrast, the ridge regression of B\* is completely populated.

The ridge penalty is the squared length of a straight line between zero and the vector space point B (distance as the crow flies).

The Lasso penalty is like the distance that a taxicab would have to drive in a city where the streets constrain it to move north-south or east-west only.

Why Lasso Penalty Leads to Sparse Coefficient Vectors:

Sparsity property stems from the form of the coef penalty function.

The shape of the curves of constant penalty is determined by the nature of the distance being used.

For ridge, the curves representing the coef penalty are circles centered at the origin. The set of points where the sum of B1 and B2 squared are constant defines a circle.

For Lasso, the curves are diamonds.

Increasing lambda increases the penalty associated and affect the value associated to each penalty curve.

OLS prediction errors are ellipses, these ellispes represent constant sum squared error. The minimum represent the unpenalized minimum and lies at the center of the ellipses.

Minimizing the sum of these two functions require a compromise somewhere in between the minimum for the prediction error and the coefficient penalty. Larger values of lambda will put the compromise closer to the minimum for the penalty (all B’s zero) and smaller values of lambda closer to the unconstrained minimum prediction error.

The overall minimum will always be a point where the curve of constant penalty is tangent to the curve of square prediction error. That’s where the distinction btw sum of squared coefs penaltiesand sum of abs value penalties becomes important.

For ridge, solutions will tend to minimize all coefs (not on either of the coordinate axis)

For lasso, solutions will completely ignore some of the variables depending on lambda values. A sparse coefficient vector is the algo’s way of telling you that you can completely ignore some of the variables. Only when lambda is small enough will the attributes be non zero.

ElasticNet Penalty Includes Both Lasso and Ridge:

Adjustable blend of the ridge and Lasso penalty. Introduces another parameter, alpha, that parameterizes the fraction of the total penalty that is the ridge penalty and the fraction that is the Lasso penalty. Alpha = 1: Lasso. Alpha=0: Ridge.

Both lambda and alpha must be specified to solve for the coefs for the linear model. Usually, the approach is to pick a value for alpha and solve for a range of lambda’s.

Solving the PLR problem

It amounts to solving an optimization problem. A number of purpose numeric optimization algo can solve the optimization problems stated above. This section goes though the mechanics of two algos least angle regression (or LARS) and Glmnet.

Understanding LARS and its relationship to fwd stepwise regression:

LARS algorithm can be understood as a refinement of the forward stepwise algo.

forward stepwise algo:

* initialize all B’s at zero

at each step:

* find residuals (errors) after using variables already chosen
* determine which unused variable best explains residuals and add it to the mix

LARS is very similar, main diff is instead of unreservedly incorporating each new attribute, it only partially incorporates them:

* initialize all B’s at zero

at each step:

* determine which attribute has the largest correlation with the residuals
* increment that variable’s coefficient by a small amount if the correlation is positive or decrement by a small amount if negative

LARS solves a slightly different pb, however, yields solutions usually the same as Lasso.

Easy to outline and relatively compact to code

PLR packages generally normalizes the attributes to get them to the same scale

E.g each step in the LARS algo increments one of the B’s by a fixed amount. If the attributes have different scales, this fixed increment means different things to different attributes

Because the algo is using normalized attributes, there is no need for the intercept B0\*.

How LARS Generates Hundreds of Models of Varying Complexity:

For ridge and lasso, B’s depend on the parameter lambda. The LARS algo doesn’t deal explicitly deal with lambda values, but it has the same effect. Each set of coefficients optimizes the ridge or lasso equation for some value of lambda

Notice that coefficients move off zero one by one and for a while there is only one or two – sparsity property. The order in which coefficients move off zero can be used as indication of the rank order of importance of the variables.

Choosing the Best Model form Hundreds LARS Generates:

Out of sample performance of the different set of coeffs

e.g Ten-fold cross-validation

Comparison of the Glmnet and LARS algo:

Extensions to Linear Regression with Numeric Input:

Solving Classification pbs with penalized regression:

Binary: turn the labels into 0 and 1 and solve the regression pb. Good alternative although there are more sophisticated approach. Substituting numeric is a simple coding approach that trains faster.

Another approach is to formulate the pb in terms of the likelihoods of the two outcomes -> logistic regression (glmnet is part of that framework).

Working with classification problems having more than two outcomes:

Several alternatives that are not ordered – multiclass classification pb

You can always handle a multiclass classification pb as a binary, technique is called one vs all or one versus the rest. Basically, you pose the pb as several binary pbs

Understanding Basis Expansion: Using Linear Methods on Nonlinear Problems:

By their nature, linear methods assume classification and regression predictions can be expressed as a linear combination of the attributes that are available to the designer. What if you have reasons to suspect that a linear model isn’t enough? You can get a linear model to work with strong nonlinearities by using basis expansion. The basic idea behind basis expansion is that the nonlinearities in your problem can be approximated as polynomials of the attributes (or sum of other nonlinear functions of the attributes); then you can add attributes that are powers of the original attributes and let a linear method determine the best set of coefficients for the polynomial.

Incorporating Non-Numeric Attributes into Linear Methods:

The standard method for converting categorical values to numeric is to code them into several new columns of attribute data. If an attribute has N possible values, it gets coded into N-1 new columns of data as follows. Identify N-1 columns of data with N-1 of the N attributes. In each row enter 1 in the ith column if the row takes the ith possible value of the categorical variable. Put zeros in the other columns. If the row takes the Nth value of the categorical variable, all the entries will be zero.

CHAP 5 – BUILDING PREDICTIVE MODELS USING PENALIZED LINEAR METHODS

Python Packages for PLRs:

Scikit-learn has packages implementing Lasso, LARS and ElasticNet regression. Faster and more reliable. sklearn.linear\_model. All models come in two flavors,e.g ElasticNet and ElasticNetCV, first one is used to calculate coefficient curves and second one to do the Cross Validation run to produce out-of-sample estimates of performance.

In some cases you won’t be able to use the CV version because you’ll need very specific control of the contents of the training and test sets for each fold:

* if pb as a categorical attribute that takes its value very infrequently; you may need to control sampling so that the attribute is evenly represented across the folds
* you may also need a different error measure from the MSE that the CV packages deliver
* when you use linear regression to solve classification pb, needs misclassification error or AUC

Several things to keep in mind when using these packages

* some of them (but not all) normalize the attributes before fitting a model (although documentation states an intention to bring everything into conformance by including normalization in all of them)
* the scikit-learn packages names variables differently than Chap4 and Friedman’s papers, it uses alpha instead of lambda, and l1\_ratio instead of alpha

Multivariable Regression: Predicting Wine Taste:

Since only integer scores: possible to treat it as a multiclass classification pb. Regression is a more natural way to pose the problem because it preserves order relationship. Error measuring will also be different (MSE vs misclassification error). Regression seems more natural in this case but there is no way to know if it will give superior performance (besides trying both).

Discussion on whether X and Y should be normalized

Bottom line: X should always be, Y doesn’t make a material difference

Didn’t get the explanation p.171, check again later

The only downside of normalizing the labels is that it makes the MSE calculation less meaningful relative to a regression problem

Binary Classification: using PLR to detect Unexploded Mines:

Multiclass Classification: Classifying Crime Scene glass Samples:

CHAP 6 – ENSEMBLE METHODS