Practical Considerations when deploying Models

Giri Iyengar

Cornell University

gi43@cornell.edu

April 30, 2018

Agenda for the day

- Calibrating models
- Optimizing for best performance
- Time Series Prediction

Overview

Calibration

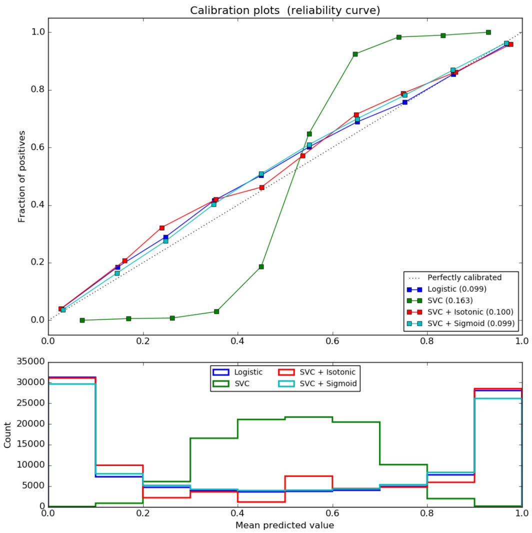
Why Calibrate - I

Often times, models we build don't give us perfect predictions. Instead we get some number which needs to be interpreted and used in downstream processing. E.g. controlling how much discount you offer someone based on model score. The more likely they are to **convert** $\rightarrow$ **higher discount**.

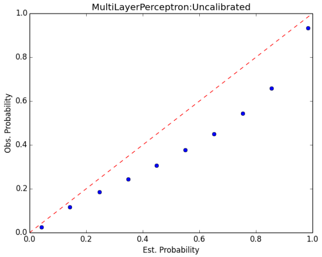
Why Calibrate - II

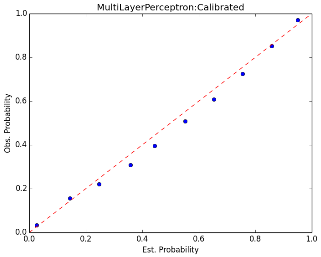
Model outputs are typically not perfectly calibrated. Several model families (e.g. SVM) don't even claim to produce calibrated outputs. So, we need to do **post-hoc** calbration

Calibration of Models



Calibration of Models





Measuring Calibration

- If we can measure it, we can improve it
- Brier Score/Loss:
$$\frac{1}{N}\sum_{t=1}^{N}(f_t - o_t)^2$$
- f_t is the **forecast** and o_t is the **actual** outcome

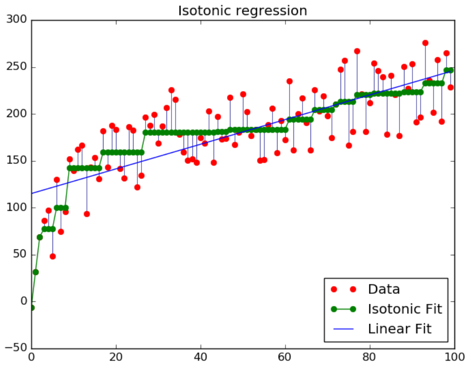
Some Calibration Algorithms

- Isotonic Regression
- Platt Scaling
- Assignment Value Algorithm

Calibration Outline

We'll take the trained model and its output on the validation set. Use the true labels and model scores on the validation set to learn a calibration function. This function is then applied to model output at test time (and in Production).

Isotonic Regression



Isotonic Regression

The isotonic regression finds a non-decreasing approximation of a function while minimizing the mean squared error on the training data.

Platt Scaling

Platt Scaling Idea

Fit a logistic curve to the output of the model

- Take the model output and label
- Learn a new logistic on the validation set
- Apply learned model to the test set

Calibration Model

$$y_{calib} = \frac{1}{1 + exp^{-a \times p_{\hat{y}} + b}}$$

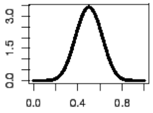
Learn a and b on validation set

Assignment Value Algorithm

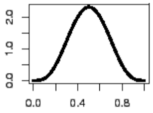
AVA Model

- Model Class 0 and Class 1 outputs as **beta** distributions
- beta distributions have a scale and a shape parameter. Defined between 0 and 1
- Calculate correction parameters. True mean for each class vs observed mean
- Apply correction factor to fix the model outputs

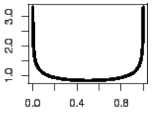
Beta Distributions



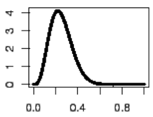
p=0.5, theta=0.05



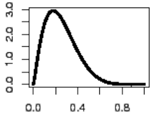
p=0.5, theta=0.1



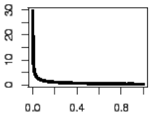
p=0.5, theta=0.4



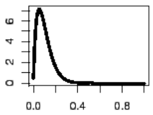
p=0.25, theta=0.05



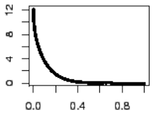
p=0.25, theta=0.1



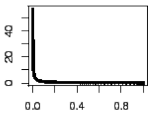
p=0.25, theta=0.4



p=0.25, theta=0.05



p=0.25, theta=0.1



p=0.25, theta=0.4

Calibration

- There is no one technique that works best in all circumstances
- 3 approaches presented here have consistently worked best on a variety of tasks
- Play with each of them and choose the best one for your task

Overview

Hyper Parameter Optimization

Fine Tuning for improving model performance

- Models have parameters that are learnt by optimizing an objective function
- In addition, models have Hyper-parameters that are usually chosen outside the objective function. That is, the objective function is optimized for a **given** setting of these hyper parameters.
- E.g. λ, α in ALS, L_1 and L_2 regularization weights in Logistic Regression, C in SVM etc

Outer Loop of Evaluation

Typical approach:

1. Choose a hyper parameter setting
2. Build and evaluate the model (expensive)
3. Choose next hyper parameter setting and repeat

Hyper Parameter Optimization

Approaches for Tuning Hyper Parameters

- Manual
- Exhaustive
- Random
- Modeling the Model

Hyper Parameter Optimization

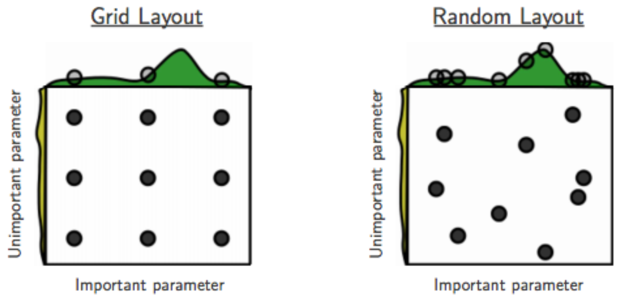
Exhaustive Search

- Combinatorial Explosion
- Expensive
- May miss important regions of optimization
- Typically some parameters are more important and some less – it may not know which is which
- Manual search often works better than exhaustive search

Hyper Parameter Optimization

Random Search

- Random search often times works better than Exhaustive Search



Can we do better? This is an active area of research.

Hyper Parameter Optimization

Sketch of Modeling the Model

- Replace the model with a **less** expensive approximation
- Choose the next point to sample more intelligently
- Spend more time in regions of importance and less time elsewhere

- Parzen Density Estimation. Construct a non-parametric model for $p(y|\mathbf{x})$
- Gaussian Process. Model $p(y|\mathbf{x}) \sim N(m(\mathbf{x}), k(\mathbf{x}))$

Hyper Parameter Optimization

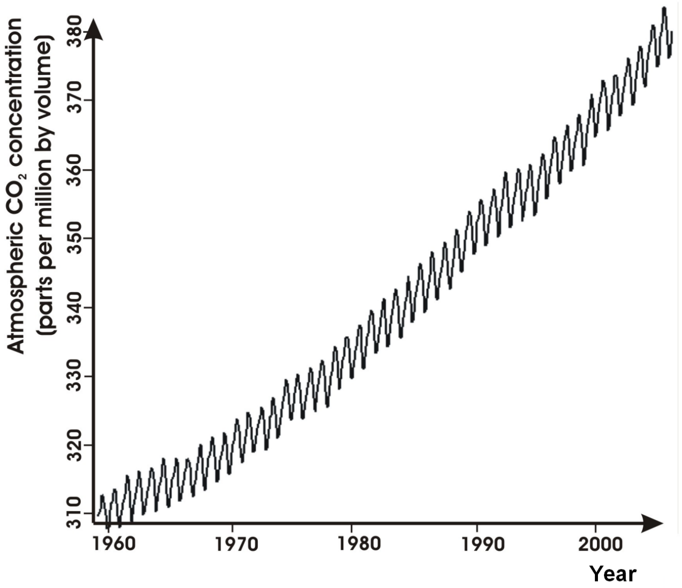
- Definitely perform HPO
- At least choose several Manual settings
- Perform Random Search – relatively easy to code up
- GP or Parzen estimators if you really want to tune the model

Model vs HPO

Balance budget between inner loop model iterations and outer loop HPO iterations

Overview

Time Series Prediction



Time Series Prediction

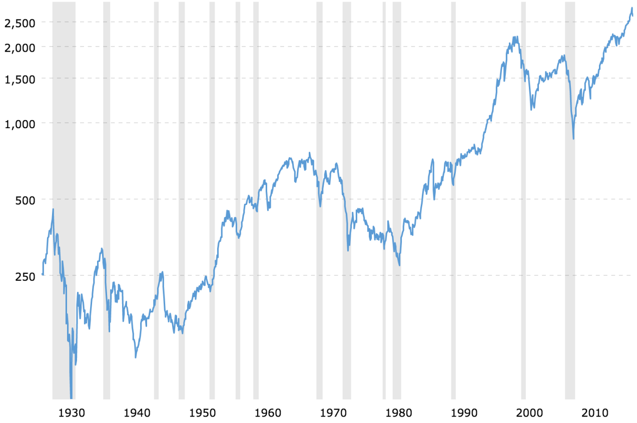


Figure: S&P 500 Price History

Classical Approach

ARIMA

In statistics and econometrics, and in particular in time series analysis, an autoregressive integrated moving average (ARIMA) model is a generalization of an autoregressive moving average (ARMA) model. Both of these models are fitted to time series data either to better understand the data or to predict future points in the series (forecasting). ARIMA models are applied in some cases where data show evidence of non-stationarity, where an initial differencing step (corresponding to the "integrated" part of the model) can be applied one or more times to eliminate the non-stationarity.

Auto-Regressive Model

AR Model

AR(p) is parametrized by lag size **p** and given as

$$X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + \epsilon_t$$

where c is a constant, ϕ_i are coefficients to be learned and ϵ_t is the residual error, usually zero mean.

Moving Average Model

MA Model

MA(q) is parametrized by order **q** and given as

$$X_t = \mu + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + \epsilon_t$$

where μ is the expectation of X_t, θ_i are coefficients to be learned and ϵ_t is the residual error.

ARMA Model

ARMA Model

ARMA(p,q) is given as

$$X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} + \epsilon_t$$

These models **assume** that the underlying time series is (weakly) stationary. Mean and Auto-covariance don't change with time. ARIMA is used when model shows evidence of non-stationarity. It tries to eliminate the non-stationarity by doing an initial differentiation operation.

Differencing Operation

$$X_t' = X_t - X_{t-1}$$

$$\begin{aligned} X_t^* &= X_t' - X_{t-1}' \\ &= X_t - X_{t-1} - (X_{t-1} - X_{t-2}) \\ &= X_t - 2X_{t-1} + X_{t-2} \end{aligned}$$

Deep Learning for Time Series Prediction

- Use RNNs – Unreasonably effective
- Hybrid models. RNNs and then fit ARMA / ARIMA

thank you
danke
謝謝
ngiyabonga
teşekkür ederim
gracias
merci
obrigado
dziękuję
bedankt
спасибо
go raibh maith agat
mochchakkeram
sukriya
kop khun krap
grazie
arigatō
dakujem
мерси
terima kasih
감사합니다
ευχαριστώ
tapadh leat
dank je
hvala
mauruuru
sağolun
takk
taiku