Assessing Model Predictive Performance

Prof. Rafal Kustra

Public Health Sciences

October 19, 2016

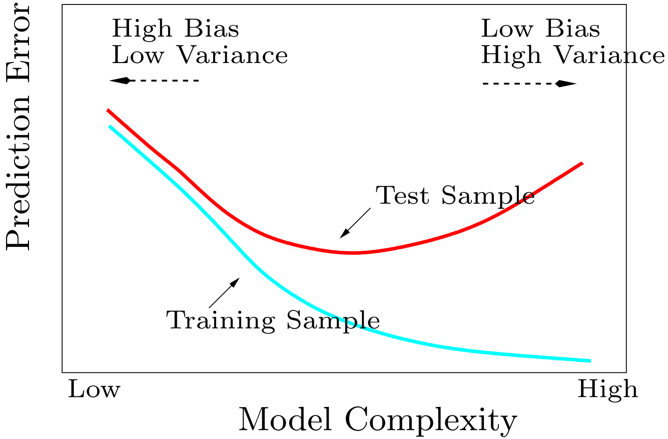
Outline

Training vs Test error

- *Training error*: how the model fits the same data on which parameters estimated
- The more complex the model the better it will fit the training data
  - In many cases have a knob tuning the complexity (roughness penalty, number of basis functions/knots, number of nearest neighbours)
  - Often can increase complexity so that training error is zero
- However when model predictions compared on *new* data they are very often worse than training error and sometimes very bad - test error

Test error and Bias-Variance tradeoff

- Typically we assume no bias in data samples
- Complex models get closer to training data - low bias
- Overly complex models will have too much variance:
    1. They pick up a lot of false “signal” from additive error
    2. They often have underlying instabilities - e.g., high-polynomial terms in model


Prediction Error
High Bias
Low Variance
Low Bias
High Variance
Test Sample
Training Sample
Low
High
Model Complexity

Why worry about PE

Outline

Stochastic Components

- Assume we have a learning method that is deterministic: applying to a given training set will always produce same results
- The result of fitting is then dependent on training set: the selection of samples there
- Of course many learning methods may have some stochastic component to them (think about random initial values, cross-validation, MCMC etc)

Loss Function

- Most measures of predictive performance will involve a *loss* function
- Assume we have data items: $\{\boldsymbol{x}_i, y_i\}$ and our method results in a function $\widehat{y} = \widehat{f}(\boldsymbol{x})$ that gives us estimates of Y (Y is a r.v.).
  - For classification problems y will be a class (discrete set of values which we can, w.l.g., code $(1, 2, \ldots, K)$
- To estimate predictive performance of a method we usually propose a loss function, $L(\cdot, \cdot)$
- For regression problems, two most often used ones:

$$L(Y, f(x)) = \begin{cases} (Y - f(x))^2 & L_2 \text{ or squared-error loss} \\ |Y - f(x)| & L_1 \text{ or absolute value loss} \end{cases}$$

- For a classifier that outputs the classes, $\widehat{f}(x) \in \{1, 2, \ldots, K\}$ we usually use 0-1 loss:

$$L(Y, f(x)) = I(Y \neq f(x))$$

- If we have estimates of posterior probabilities, $f(x) = \widehat{p}_k(x)$ available, an equivalent (to square loss in Gaussian world) choice may be a deviance (or *cross-entropy* loss):

$$L(Y, f(x)) = -2 \sum_{k=1}^{K} I(Y \neq k) \log(\widehat{p}_k(x))$$

but many times other measures are used

Population value

- Once a (deterministic) methods is selected and training set is obtained we (frequentists) can think of a estimating a true, but unknown, predictive performance in the population
- Given a loss function, we may be after:

$$PE = \mathrm{E}\left[L(Y, \widehat{f}(X))\right]$$

- This expectation goes over everything: selection of N training samples which produce $f(X)$ and the averaging over the distribution of test samples, $\{X, Y\}$, which are from the same population as training set
- Sometimes it is more reasonable to condition on the training sample, $\mathcal{T}$ observed. Then

$$PE(\mathcal{T}) = \mathrm{E}\left[L(Y, \widehat{f}_{\mathcal{T}}(X)\right) |\mathcal{T}]$$

where now the expectation is over test samples only
- Most of the rest of the lecture is devoted to how do we get the estimates of PE using data

Outline

Different distributions

- Prediction error applies to expected loss over full joint distribution of x's and y's
- However since we are evaluating a learning method - a model fitted to finite data - we often need to distinguish between training and new observations
- The overall mechanism that generates data of course does not distinguish between these two: but once we condition on training sample we need to observed the *i.i.d* rule: that (conditional on x) correlation between any new $y_0 \equiv y(x_0)$ and any existing $y_i \equiv y(x_i)$ is zero *even for same* x, ie when $x_0 = x_i$

Apparent error rate

- A naive estimate of PE would be to average our loss on the training sample. This is actually useful, but not directly:

$$\overline{\mathrm{err}} = N^{-1} \sum_i L(y_i, f(x)_i)$$

- As an estimate of the PE this will be biased ??wards for three reasons (assuming additive model):
  1. We have directly minimized loss on training set to produce $\widehat{f}(x)$
  2. typically covariate values in training set do not exhaust all possibilities: new cases can appear with different $\boldsymbol{x}$
  3. There is an additive error term hiding in each y_i, so next realization of y_i (with same $\boldsymbol{x}$) will have different value

In-sample prediction error

- Sometimes convenient to focus on observed covariate points
  - This will clearly underestimate the full PE since it is easier to get the true function right at points in X-space where we observed our training samples: no interpolation (and sometimes extrapolation)
  - It can however help us understand *optimism* of the training error
  - It can also be useful in relative sense, for example for selecting smoothing parameter or comparing different methods
- In expectation terms, we are restricting ourselves to x_i's in a training set:

$$\begin{aligned} PE_{is} &= \mathrm{E}\left[L(Y, f(X)) | X \in \mathcal{T}\right] \\ &= 1/N \sum_i \mathrm{E}_Y \mathrm{E}_{Y^{new}} L(Y_i^{new}, \widehat{f}(x_i)) \end{aligned}$$

Optimism of Learning rule

- The apparent error rate will be often too low compared to real (test) error
- We call the difference *optimism*

$$\mathrm{op} = \mathrm{PE} - \mathrm{E}_Y\overline{\mathrm{err}}$$

- If we could estimate the optimism ($\widehat{\mathrm{op}}$) we could use it to get the estimate of PE:

$$\widehat{\mathrm{PE}} = \mathrm{E}_Y\overline{\mathrm{err}} + \widehat{\mathrm{op}}$$

- There is a remarkable results for *In-sample* optimism:

$$\mathrm{op}_{is} = \mathrm{PE}_{is} - \mathrm{E}_Y\overline{\mathrm{err}} = \frac{2}{N}\sum_i \mathrm{Cov}(\widehat{f}(x_i), y_i)$$

  it holds for squared-error, 0-1 loss and many other loss functions
- Intuitively, the more you (over)fit, the greatest correlation will be between random (true) value of y_i and it's fitted value, and this drives optimism

Examples with regression

- Lets focus on a single *test* point, (x_0, y_0) and a regression case where a true model is
$$Y = f(X) + \epsilon$$
and using squared-error loss
- We showed already that
$$\mathrm{PE}(x_0) = \sigma_\epsilon^2 + \left[\mathrm{E}\widehat{f}(x_0) - f(x_0)\right]^2 + \mathrm{E}\left[\widehat{f}(x_0) - \mathrm{E}\widehat{f}(x_0)\right]^2$$
- For k-NN regression, this has a simple form
$$\mathrm{PE}_k(x_o) = \sigma_\epsilon^2 + \left[f(x_0) - 1/k \sum_{\nu=1}^{k} f(x_{(\nu)})\right]^2 + \sigma_\epsilon^2/k$$
which incidently shows how hyper-parameter k trades bias and variance

- With OLS model we get something like:

$$\mathrm{PE}(x_0) = \sigma_\epsilon^2 + [f(x_o) - \mathrm{E}\widehat{f}(x_0)]^2 + ||\boldsymbol{h}(x_0)||^2\sigma_\epsilon^2$$

  where predicted value is:

$$\widehat{f}(x_0) = x_0'(X'X)^{-1}X'\boldsymbol{y} = \boldsymbol{h}'\boldsymbol{y}$$

- The error at x_0 will in general depend on x_0, but if we want to get an *in-sample* average error it is easy to show that $\mathrm{Ave}_i||\boldsymbol{h}(x_i)||^2 = p/N$ and then we have:

$$PE_{\text{is}} = \sigma_\epsilon^2 + [f(x_o) - \mathrm{E}\widehat{f}(x_0)]^2 + p/N\sigma_\epsilon^2$$

Akaiki Information Criterion and Mallows C_p

- In case linear regression model one can show that:
$$\mathrm{op}_{is} = 2 \cdot \frac{p}{N} \sigma_\epsilon^2$$
where p is number of parameters fitted (number of columns in design matrix)
- Mallow's C_p uses a plug-in estimate apparent error and usual estimate for σ_ϵ^2 to arrive at estimate of in-sample PE:
$$C_p = \overline{\mathrm{err}} + 2\frac{p}{N}\widehat{\sigma}_\epsilon^2$$
- AIC generalizes is to log-likelihood loss function:
$$-2\mathrm{E}[\log \mathrm{P}_{\widehat{\theta}}(Y)] \approx -\frac{2}{N}\mathrm{loglik} + 2 \cdot \frac{p}{N}$$
where the term on the left is (-twice) an expected log density at Y using a parameter vector fitted under maximum likelihood ($\widehat{\theta}$), and loglik is a maximized log-likelihood for N training points.

BIC and AIC

- For square-error loss and Gaussian error AIC and C_p are the same (up to a constant, and assuming known σ_ϵ), but AIC holds more generally for other MLE methods (like GLMs)
- Bayesian Information Criterion by Schwartz (1979) was derived from different point of view but looks very similar:

$$\mathrm{BIC} = -2 \cdot \mathrm{loglik} + (\log N) \cdot p$$

- For squared-error and Gaussian distribution we have:

$$\mathrm{BIC} = \frac{N}{\sigma_\epsilon^2} \left[\overline{\mathrm{err}} + (\log N) \frac{p}{N} \right]$$

  which makes it proportional to C_p (and AIC) with $2 \cdot d/N$ replaced by $(\log N) \cdot d/N$
- BIC will put heavier penalty on larger p (more complex models)

Generalized Cross-Validation

- A leave-one-out cross-validation attempts to mimic the new observations by fitting N models, each without (x_i, y_i), $\widehat{f}_{(-i)}$ and using it to predict (y_i)
- Average squared errors produces an estimate to PE:

$$\mathrm{PE}_{\mathsf{looCV}} = 1/N \sum_i (y_i - \widehat{f}_{(-i)}(x_i))^2$$

- For many linear models $\widehat{\boldsymbol{y}} = S\boldsymbol{y}$ this update formula holds:

$$\mathrm{PE}_{\mathsf{looCV}} = 1/N \sum_i \left[\frac{y_i - \widehat{f}(x_i)}{1 - S_{ii}} \right]^2$$

  where $\widehat{f}()$ is a function fitted on the *whole* data
- replacing S_{ii} with its average, $\mathrm{trace}(S)/N$ we get:

$$GCV = 1/N \sum_i \left[\frac{y_i - \widehat{f}(x_i)}{1 - \mathrm{trace}(S)/N} \right]^2$$

AIC, BIC and GCV for regularized models

- Of course we have already seen trace(S) - as an effective degrees of freedom (EDF).
- Both AIC and BIC can be used with non-linear or penalized models using EDF in place of p
- Typically one will have a family of regularized models parameterized by roughness (or smoothness) parameter, λ
- For example in smoothing splines, the fit is:

$$\begin{aligned} \widehat{\boldsymbol{y}}_\lambda &= \Phi(X)(\Phi(X)'\Phi(X) + \lambda\Omega)^{-1}\Phi(X)\boldsymbol{y} \\ &= S(\lambda)\boldsymbol{y} \end{aligned}$$

  and one can use $p(\lambda) = \text{trace} S(\lambda)$ to calculate either of the three error estimates
- This can allow to choose λ and hence model complexity which asymptotically and approximately minimizes the prediction error

Split-sample validation set

- If we had a large independent test-set, $\mathcal{T}_0$ we could easily estimate the PE using plug-in estimator:

$$\widehat{\mathrm{PE}} = |\mathcal{T}_0|^{-1} \sum_{i \in \mathcal{T}_0} L(y_i, \widehat{f}(x_i))$$

- Typically no such test set if forthcoming: and if it were we would like to include it to improve our estimate of $\widehat{f}(x)$
- Of course we can always split our available dataset into two parts: training and test
  - What relative sizes to use?
  - Splitting action is random - how to take this into account?
  - Can the efficiency be improved?

V-fold Cross-Validation

- V-fold CV attempts to utilize the data more efficiently

V-fold CV algorithm

1. Randomly divide the samples in the dataset into V parts of roughly the same size
2. Loop over $v \in \{1, 2, \ldots, V\}$
3. In each iteration the v^{th} part (fold) becomes a validation set, and the rest of the samples constitute the training set
4. Build a model, $\mathcal{M}$ on the training set and estimate the prediction error, $\text{PE}^{(v)}$ on the validation set
5. At the end, report the average PE.

More on CV

- In V-fold CV, each sample will be in a test set exactly once (and will be in the training set V-1 times). Hence can form a vector of all N predicted values, $\widehat{f}(x_i)$ produced by fits that excluded i^{th} observation
- V is typically between 5-10: the larger it is the smaller the validation dataset, and more correlated the training sets
- Leave-One-Out cross-validation is for $V = N$: this typically biases the PE estimate downwards (too optimistic)
- On the other hand too low V has its own problems:
  - The actual V-fold splitting is random, and for small V (2, 3 fold) this randomness can lead to very variable estimates of PE
  - PE of a given method depends on training-set size N: typically the larger the N the smaller the PE. For small V in V-fold CV, the training set sizes will be about $N - N/V$

LOO bootstrap

- Bootstrap is a general method to mimic the sampling of a training set
- Bootstrap set, $\mathcal{T}^*$ is a sample of size N where each observation is selected with equal probability (1/N) *and with replacement* from original set
- One can show that on average a bootstrap sample will contain 63.2% of observations from the original sample (some of them repeated more than once, of course)
- That leaves about 37% observations *not* selected to a particular bootstrap training set
- In LOO bootstrap once repeatedly samples bootstrap sets, $\mathcal{T}^*$ fits the learning algorithm, and predicts the average of 37% of cases left out to produce PE*.
- PE* are then averaged to obtain the PE estimate

Outline

Model Selection

- In typically data-mining project there will be a various potential modelling approaches to try:
    - For example in regression project, one may try basis-expansion and ridge-regression; boosting model with certain base-learner (like a regression tree); neural-network model, support-vector regression
    - As part of the modelling approach one may need to decide which covariates to drop, merge, expand into basis; how and weather to normalize covariates or transform a response, etc
- Each such choice constitutes a different modelling approach (learning method) and we may want to know which one is better

Resampling for model selection

- V-fold CV or Bootstrap will often be used to choose between different models
- In each fold, a full learning method is repeated on a training set and applied to the validation set
- Then resulting PE estimates are compared to choose the method with lowest estimate

IMPORTANT!

It is absolutely crucial that all parts of a learning method are repeated in each fold. For example, if as a first step we use univariate models - or correlations - to drop some covariates, this has to be validated as well

Model optimization

- For many approaches (learning methods) there is typically one or more hyperparamaters that usually regulate model complexity
  - For regularized methods, such as ridge regression, there will be a smoothing parameter, λ
  - Simple linear regression can be extended to all-subsets regression, and the size of the subset will be a regurizing parameter
  - k-NN
  - SVMs have a regularizer and many kernels within them has a regularizing paramater
  - Decision trees have various ways to control depth (complexity) of a tree
  - Neural Networks have ways to semi-automatically control number of hidden layers and many use ridge-like penalty (weight-decay)
- These hyper-parameters are often optimized to improve PE

Model Selection and Optimization

- If you have decided on a single modeling approach, you can easily do V-CV or Boostrap to optimize hyper-parameters
- If you are choosing among various methods, you may do double-resampling:
  1. In outer loop (CV or Boostrap) the different learning methods are tested: fitted on a current training set and applied to the current validation set
  2. In inner loop, an (outer) training set if further split (multiple times) to optimize hyperparameters
- Double resampling may be too computationally intensive and/or the data set may be too small or irregular to support it
- An alternative is to use asymptotic methods (AIC, BIC, GCV) for optimizing hyper-parameters, and resampling for model selection

Outline

Honest estimates of PE

- It is surprisingly easy to (inadvertantly) cheat when estimating PE
- We often spent a lot of time looking and “playing” with data before embarking on a more formal modelling
- This can easily lead to overfitting
- It is also easy to cheat when constracting features:
  - For example, when we constract features that combine historical performance up to date: with future data left for validating it is easy to inadvertantly borrow from the future
  - Sometimes we will be using information contained in previous responses to constract features: again easy to overlook it when applying a model to a test set
- Also sometimes observations can be “bundled”: in geographical area, temporally, administratively: make sure you know what you think new cases are
- When normalizing covariates, statistics from whole dataset (means, variances etc) are used. These normalizing estimates should NOT contain information from the test set AND the test-set normalization SHOULD apply estimates from the training set

Model Validation: Overfitting the test set

- Like many things in modelling too much of a good thing can be detrimental
- If you fit hundred different models, and use resampling to choose the best one, you may need to validate this process
  - If the projects expects certain predictive performance from your final model, the minimum resampling PE you get from 100 models will very likely be overoptimistic as a final PE estimate of the model
- In some cases you may need to either:
  - Employ another outer validation loop to get the honest estimate of min-PE selected model
  - Split the data set *at the beginning* and leave a true testing set for the final validation