Measuring Performance in Supervised Learning

-Applied Analytics-

APM Chapter 5, ISLR Chapter 2.2

Lecturer: Darren Homrighausen, PhD

Preamble:

- Outline the notation for supervised learning
- Define a loss function
- Give the specifics of loss functions for regression

Notation recap

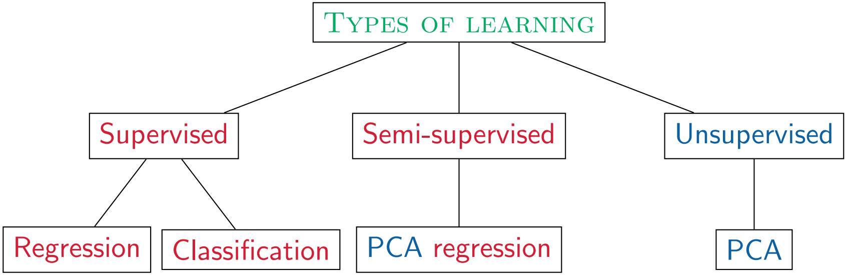
- We have data $\mathcal{D} = \{(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)\}$
  (The training data)
- $X \in \mathbb{R}^p$ is a vector of measurements for each subject
  (Example: $X_i = [1, \mathrm{income}_i, \mathrm{education}_i]^\top$)
- $x \in \mathbb{R}^n$ is a vector of subjects for each measurement
  (Example: $x_j = [\mathrm{income}_1, \mathrm{income}_2, \ldots, \mathrm{income}_n]^\top$)
- X_{ij} is the j^{th} measurement on the i^{th} subject
  (Example: $X_{ij} = \mathrm{income}_i$)

Notation recap

We will concatenate the features into the design or feature matrix $\mathbb{X}$, and the supervisors into the supervisor vector $\mathbb{Y}$

$$\mathbb{X} = \begin{bmatrix} | & & | \\ x_1 & \cdots & x_p \\ | & & | \end{bmatrix} = \begin{bmatrix} -X_1- \\ -X_2- \\ \vdots \\ -X_n- \end{bmatrix} \quad \text{and} \quad \mathbb{Y} = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix}$$

This makes $\mathbb{X}$ an $n \times p$ matrix and $\mathbb{Y}$ a length n vector.



Some comments:

Comparing predictions to Y gives a natural notion of prediction accuracy

Much more heuristic, unclear what a good solution would be

The set-up

We use the training data $\mathcal{D}$ to train a model, producing $\hat{f}$
(This means that there are model parameters inside f that get estimated)

Goal: Given a new $X \in \mathbb{R}^p$ but not the associated Y, we want to form predictions

$$\hat{f}(X) = \hat{Y}$$

Such that $\hat{Y}$ is a good prediction of Y, the unobserved supervisor

We need to define what it means to make good predictions

Example: We want to know how much a house is likely to sell for. We have trained a model $\hat{f}$ on $\mathcal{D}$ and put the features of the house, X, into the trained model. We get out prediction $\hat{Y} = \hat{f}(X)$

Loss functions

If we want a $\hat{f}(X)$ which is a good prediction, what does good mean?

That means there is some true Y that we haven't observed that $\hat{Y}$ is close to

(That would be the actual sales price for the house, which hasn't been observed, yet)

Define a loss function which

- Inputs both
  - $\underbrace{\hat{f}(X)}_{\text{Our prediction}}$
  - $\underbrace{Y}_{\text{Unknown, true value}}$
- Outputs a number $\ell(\hat{f}(X), Y)$ between 0 and ∞...

...such that smaller $\ell(\hat{f}(X), Y)$ indicate better performance

The loss function for regression

When the supervisor is quantitative, then we are doing regression

(Note that regression doesn't mean multiple linear regression, that is a special case)

Regression loss function is the squared error function

$$\ell(\hat{f}(X), Y) = (\hat{f}(X) - Y)^2$$

The details of this loss function will depend on what data are used to compute it..

The apparent error

We are now in a better position to define the apparent or training error

The apparent error is the loss function computed on the training data

Again in the case of regression and squared error loss, the apparent error can be written

$$\mathrm{SSE} = \sum_{i=1}^{n} (Y_i - \hat{f}(X_i))^2$$

(Sometimes this is divided by the number of terms to produce a mean)

Hence, the issue with the apparent error can be restated for regression as: we don't want to use the SSE for computing model parameters and setting model flexibility

Back to data splitting

Defining a good model

We can compute the loss on any of the subsets generated by data splitting

1. Training: Used to fit (or train) the considered models, producing some candidates $\hat{f}$

   (So, the use SSE as a criterion for estimating model parameters)

2. Validation: Used to choose each candidate model's tuning parameter(s)

   (Use the loss computed on the validation subset to choose model flexibility)

3. Testing: Used to choose amongst estimated/tuned models

Ultimately, we want to estimate/choose models that generalize well, which we measure by computing the loss on the testing data

$$\text{test error} = \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} (Y_i - \hat{f}(X_i))^2$$

Hence, we will define a good model as one that has small test error

Bias and variance trade-off

Imagine there is some unknown model f_* (some choice of model, model parameters, and tuning parameters) that would minimize the test error

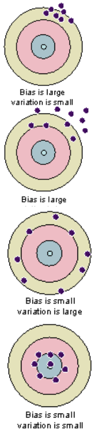
(If we had an infinitely large test set, we could hypothetically find this model)

For large enough test sets

(that is, as n_{test} goes to infinity)

$$\text{test error} = \text{bias}^2 + \text{variance} + \sigma^2$$

- The bias is how far, on average, we would be from f_*
- The variance is how sensitive our model is to small changes in the data
- σ^2 is the irreducible error, and is the test error of the best possible model f_*



Bias and variance trade-off

There is a natural conservation between these quantities

Low bias $\rightarrow$ complex model $\rightarrow$ many parameters $\rightarrow$ high variance

The opposite also holds
(Think: $\hat{f} \equiv 0$.)

We'd like to 'balance' these quantities to get the best possible predictions

Bias-variance tradeoff

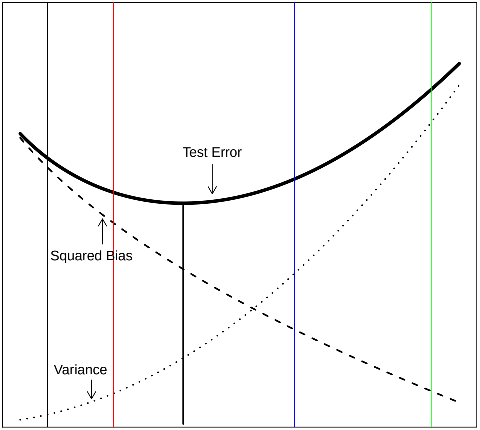


Figure: Model Complexity $\nearrow$

Example

Example

Let's look at a simple simulation to see the issue with using the apparent error to set model flexibility

Let's suppose $\mathcal{D}$ is drawn as

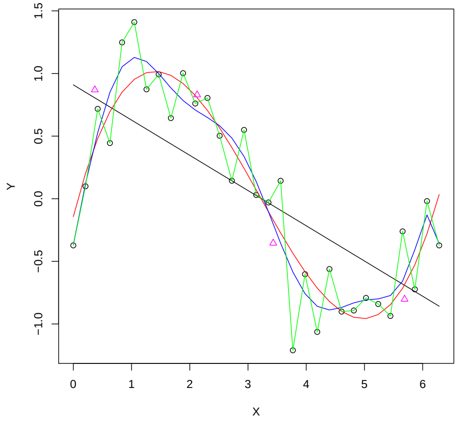
```
n = 30
X = (0:n)/n*2*pi
Y = sin(X) + rnorm(n,0,.25)
```

We will discuss this more soon, but let's imagine we want to fit a polynomial model to $\mathcal{D}$. These have the form:

$$f(X_i) = \beta_0 + \sum_{j=1}^{J} \beta_j X_i^j$$

If we try to estimate the model parameters (β_j) and set the model flexibility (J) with apparent error, we would see the following..

Example



The apparent errors are:

10.98

2.86

2.28

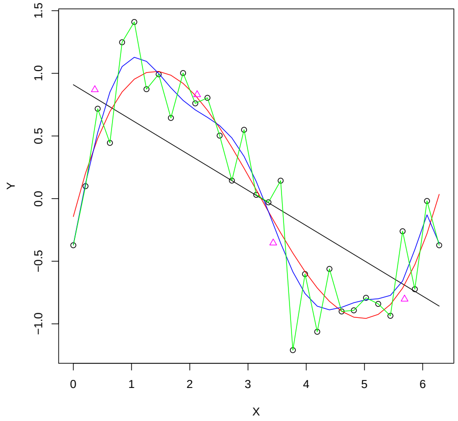
0

So, we would set the flexibility J to the maximum value (which is 29 in this case)

We would overfit!

What about predicting new observations (△)?

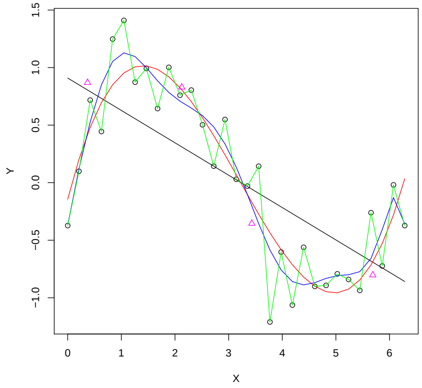
Example

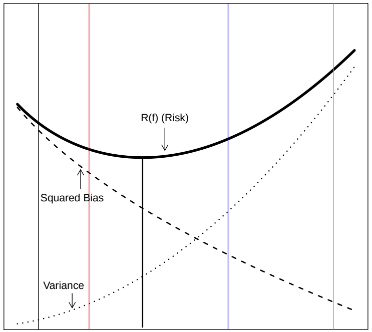


- Black model has low variance, high bias
- Green model has low bias, but high variance
- Red model and Blue model have intermediate bias and variance.

We want to balance these two quantities.

Bias vs. Variance





Model Complexity $\nearrow$

Coefficient of determination

Coefficient of determination, R^2

A very commonly used measure of model performance is the coefficient of determination

It takes the form of a proportion: the proportion of variance in Y explained by the model

$$R^2 = \frac{TSS - SSE}{TSS} = 1 - \frac{SSE}{TSS}$$

where $TSS = \sum_{i=1}^{n}(Y_i - \overline{Y})^2$ is the total sums of squares

Important: Note that the apparent error appears in R^2. Hence, it is subject to the same limitations

(In particular, do not use R^2 to set model flexibility)

Coefficient of determination, R^2

Some facts:

- R^2 is between 0 and 1
  (with larger values considered 'better', but remember the apparent error connection)
  - A value near 0 could mean a poor model fit or that the irreducible error is very large
  - The model is poor fit
- If there is a single feature X and supervisor Y, then R^2 is the square of the correlation of X and Y
  (However, the expression in terms of SSE and TSS is more general and applies to any model)

Postamble:

- Outline the notation for supervised learning
  (We train a model f using training data to produce $\hat{f}$ and make predictions at a new feature value X: $\hat{f}(X)$)
- Define a loss function
  (A loss function takes in two arguments: the prediction and what the true supervisor value is/would be)
- Give the specifics of loss functions for regression
  (We generally use square error loss for regression)